# Unit Descriptors for the Pearson BTEC Higher Nationals Computing Suite

## Issue 1

For use with:

**Pearson BTEC Higher National Certificate and Diploma in Computing for England**

**Pearson BTEC Higher National Certificate and Diploma in Computing**

**About Pearson**

We are the world's leading learning company operating in countries all around the world. We provide content, assessment and digital services to students, educational institutions, employers, governments and other partners globally. We are committed to helping equip students with the skills they need to enhance their employability prospects and to succeed in the changing world of work. We believe that wherever learning flourishes so do people.

*References to third party materials made in this document are made in good faith. Pearson does not endorse, approve or accept responsibility for the content of materials, which may be subject to change, or any opinions expressed therein. (Materials may include textbooks, journals, magazines and other publications and websites.)*

*All information in this document is correct at time of publication.*

All the material in this publication is copyright
© Pearson Education Limited 2026

# Contents


**1.0 Introduction** **1**

1.1 Qualifications indicated ‘for England’ 1

1.2 Qualifications not indicated ‘for England’ 1

**2.0 Programme structures** **2**

**3.0 The unit descriptor** **7**

**4.0 Unit descriptors** **9**

Unit 401: Fundamentals of Computer Science 10

Unit 402: Designing a Computing Project (Pearson-set) 23

Unit 403: Professional Practice 36

Unit 404: Digital Systems Analysis 50

Unit 405: Core Software Design and Development 62

Unit 406: Core Software Testing 73

Unit 407: Applied Software Testing 85

Unit 408: Applied Software Development 98

Unit 409: Core Cybersecurity 108

Unit 410: Database Engineering 121

Unit 411: Core Data Analytics 133

Unit 412: Applied Data Analytics 145

Unit 413: Core Networking 158

Unit 414: Applied Networking 170

Unit 415: Applied Cybersecurity 183

Unit 416: Core Cloud Computing 195

Unit 417: Cloud DevOps Automation 208

Unit 418: Cloud Networking 221

Unit 419: Cloud Security 235

Unit 420: Core AI 249

Unit 421: AI Automation and Digital Productivity 260

Unit 422: Ethics and AI 274

Unit 423: Agentic AI 288

Unit 424: Creative AI Coding 299

Unit 425: AI Programming 312

Unit 426: Digital Project Management 325
Unit 427: UI/UX Programming and Development 338
Unit 428: Maths for Computing 349
Unit 429: Responsive Website Development 358
Unit 430: Emerging Technologies 369
Unit 431: Operating Systems 378
Unit 432: System Architecture 389
Unit 433: Game Development 399
Unit 434: Green Computing 412
Unit 435: Client-Server Computer Systems 423
Unit 501: Computing Research Project (Pearson-set) 436
Unit 502: Applied Computing Project (Pearson-set) 449
Unit 503: Risk Assessment and Change Management 461
Unit 504: Security Testing 474
Unit 505: Application Development 485
Unit 506: Advanced Software Design 496
Unit 507: Full Stack Development 507
Unit 508: Discrete Mathematics 519
Unit 509: Enterprise Database Management Systems 530
Unit 510: Advanced Data Engineering 541
Unit 511: Big Data Analytics and Visualisation 552
Unit 512: Advanced Data Analytics 564
Unit 513: AI for Data Analysis 576
Unit 514: Quality of Service (QoS) 587
Unit 515: Secure System Design 598
Unit 516: Internet of Things 610
Unit 517: Applied Data Forensics 622
Unit 518: Cloud Data Centre Operations 634
Unit 519: Cloud System Architecture 647
Unit 520: Applied AI 660
Unit 521: Generative and Conversational AI 673
Unit 522: Machine Learning Concepts 686
Unit 523: Natural Language Processing 697
Unit 524: Advanced AI Concepts 708
Unit 525: Computer Vision 720

Unit 526: Deep Learning 731
Unit 527: System Analysis and Design 744
Unit 528: Robotics 756
Unit 529: Game Design and Scripting 767
Unit 530: Information Security Management 779
Unit 531: E-Commerce and Strategy 792
Unit 532: Prototyping with AI 800
Unit 533: Transport Layer Engineering 813
Unit 534: Digital Sustainability 825
Unit 535: Virtual and Augmented Reality 838
Unit 536: Work-Based Learning 850
Unit 537: Fundamentals of Blockchain Technology 863
Unit 538: Applied Cloud Cryptography 876
Unit 539: Cloud Infrastructure Protection 888
Unit 540: Improving Organisational Performance 901
Unit 541: FinOps 909
Unit 542: API Design and Development 918
Unit 543: Data Structures and Algorithms 930

# 1.0 Introduction

The unit descriptors included in this document are for use with the following qualifications:

- Pearson BTEC Level 4 Higher National Certificate in Computing: **610/8025/1**
- Pearson BTEC Level 5 Higher National Diploma in Computing: **610/8026/3**
- Pearson BTEC Level 4 Higher National Certificate in Computing for England: **610/8027/5**
- Pearson BTEC Level 5 Higher National Diploma in Computing for England: **610/8028/7**.

## 1.1 Qualifications indicated 'for England'

Qualifications that are indicated as 'for England' are designed to align to the requirements of specific occupational standards that meet Department for Education (DfE) current occupation criteria.

Meeting the requirements of the occupational standards relates to:

- qualifications that are 'quality marked' as Higher Technical Qualifications (HTQs)
- the knowledge, skills and behaviours for identified occupations that are associated with relevant occupational standards.

## 1.2 Qualifications not indicated 'for England'

Qualifications that are **not** indicated as 'for England' can be delivered at any centre, in the UK or overseas, subject to approvals from Pearson. These qualifications are not 'quality marked' as HTQs by DfE.

# 2.0 Programme structures

Programme structures define the unit combinations required for a given qualification. These are defined in *Section 6.0 Programme structures* within the relevant programme specification for the qualification.

 © Pearson Education Limited 2026

## 2.1 Programme structures overview

The infographic provides an 'at a glance' view of the pathway structures available in these qualifications.

*Table 1: HNC/HND pathways showing core mandatory, specialist mandatory and space for optional or specialist units*

### Level 4

| CREDITS | UNIT TYPE | UNIT NUMBER | L4 GENERAL PATHWAY | UNIT TYPE | UNIT NUMBER | L4 QUALITY ENGINEERING | UNIT TYPE | UNIT NUMBER | L4 SOFTWARE ENGINEERING | UNIT TYPE | UNIT NUMBER | L4 DATA ENGINEERING |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **15** | L4 Core Mandatory | **401** | Fundamentals of Computer Science | L4 Core Mandatory | **401** | Fundamentals of Computer Science | L4 Core Mandatory | **401** | Fundamentals of Computer Science | L4 Core Mandatory | **401** | Fundamentals of Computer Science |
| **15** | L4 Core Mandatory | **402** | Designing a Computing Project | L4 Core Mandatory | **402** | Designing a Computing Project | L4 Core Mandatory | **402** | Designing a Computing Project | L4 Core Mandatory | **402** | Designing a Computing Project |
| **15** | L4 Core Mandatory | **403** | Professional Practice | L4 Core Mandatory | **403** | Professional Practice | L4 Core Mandatory | **403** | Professional Practice | L4 Core Mandatory | **403** | Professional Practice |
| **15** | L4 Specialist Mandatory | **404** | Digital Systems Analysis | L4 Specialist Mandatory | **405** | Core Software Design and Development | L4 Specialist Mandatory | **405** | Core Software Design and Development | L4 Specialist Mandatory | **404** | Digital Systems Analysis |
| **15** | L4 Optional | | | L4 Specialist Mandatory | **406** | Core Software Testing | L4 Specialist Mandatory | **408** | Applied Software Development | L4 Specialist Mandatory | **405** | Core Software Design and Development |
| **15** | L4 Optional | | | L4 Specialist Mandatory | **407** | Applied Software Testing | L4 Specialist Mandatory | **409** | Core Cybersecurity | L4 Specialist Mandatory | **410** | Database Engineering |
| **15** | L4 Optional | | | L4 Optional | | | L4 Optional | | | L4 Specialist Mandatory | **411** | Core Data Analytics |
| **15** | L4 Optional | | | L4 Optional | | | L4 Optional | | | L4 Optional | | |

### Level 5

| CREDITS | UNIT TYPE | UNIT NUMBER | L5 GENERAL PATHWAY | UNIT TYPE | UNIT NUMBER | L5 QUALITY ENGINEERING | UNIT TYPE | UNIT NUMBER | L5 SOFTWARE ENGINEERING | UNIT TYPE | UNIT NUMBER | L5 DATA ENGINEERING |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **30** | L5 Core Mandatory | **501 or 502** | Computing Research Project **OR** Applied Computing Project | L5 Core Mandatory | **501 or 502** | Computing Research Project **OR** Applied Computing Project | L5 Core Mandatory | **501 or 502** | Computing Research Project **OR** Applied Computing Project | L5 Core Mandatory | **501 or 502** | Computing Research Project **OR** Applied Computing Project |
| **15** | L5 Specialist Mandatory | **503** | Risk Assessment and Change Management | L5 Specialist Mandatory | **503** | Risk Assessment and Change Management | L5 Specialist Mandatory | **506** | Advanced Software Design | L5 Specialist Mandatory | **509** | Enterprise Database Management Systems |
| **15** | L5 Optional | | | L5 Specialist Mandatory | **504** | Security Testing | L5 Specialist Mandatory | **507** | Full Stack Development | L5 Specialist Mandatory | **510** | Advanced Data Engineering |
| **15** | L5 Optional | | | L5 Specialist Mandatory | **505** | Application Development | L5 Specialist Mandatory | **508** | Discrete Mathematics | L5 Specialist Mandatory | **511** | Big Data Analytics and Visualisation |
| **15** | L5 Optional | | | L5 Optional | | | L5 Optional | | | L5 Optional | | |
| **15** | L5 Optional | | | L5 Optional | | | L5 Optional | | | L5 Optional | | |
| **15** | L5 Optional | | | L5 Optional | | | L5 Optional | | | L5 Optional | | |

## Level 4

| CREDITS | UNIT TYPE | UNIT NUMBER | L4 DATA ANALYTICS | UNIT TYPE | UNIT NUMBER | L4 NETWORK ENGINEERING | UNIT TYPE | UNIT NUMBER | L4 CYBERSECURITY | UNIT TYPE | UNIT NUMBER | L4 CLOUD COMPUTING |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **15** | L4 Core Mandatory | **401** | Fundamentals of Computer Science | L4 Core Mandatory | **401** | Fundamentals of Computer Science | L4 Core Mandatory | **401** | Fundamentals of Computer Science | L4 Core Mandatory | **401** | Fundamentals of Computer Science |
| **15** | L4 Core Mandatory | **402** | Designing a Computing Project | L4 Core Mandatory | **402** | Designing a Computing Project | L4 Core Mandatory | **402** | Designing a Computing Project | L4 Core Mandatory | **402** | Designing a Computing Project |
| **15** | L4 Core Mandatory | **403** | Professional Practice | L4 Core Mandatory | **403** | Professional Practice | L4 Core Mandatory | **403** | Professional Practice | L4 Core Mandatory | **403** | Professional Practice |
| **15** | L4 Specialist Mandatory | **411** | Core Data Analytics | L4 Specialist Mandatory | **413** | Core Networking | L4 Specialist Mandatory | **409** | Core Cybersecurity | L4 Specialist Mandatory | **416** | Core Cloud Computing |
| **15** | L4 Specialist Mandatory | **412** | Applied Data Analytics | L4 Specialist Mandatory | **414** | Applied Networking | L4 Specialist Mandatory | **413** | Core Networking | L4 Specialist Mandatory | **417** | Cloud DevOps Automation |
| **15** | L4 Optional | | | L4 Specialist Mandatory | **416** | Core Cloud Computing | L4 Specialist Mandatory | **415** | Applied Cybersecurity | L4 Specialist Mandatory | **418** | Cloud Networking |
| **15** | L4 Optional | | | L4 Optional | | | L4 Optional | | | L4 Specialist Mandatory | **419** | Cloud Security |
| **15** | L4 Optional | | | L4 Optional | | | L4 Optional | | | L4 Optional | | |

## Level 5

| CREDITS | UNIT TYPE | UNIT NUMBER | L5 DATA ANALYTICS | UNIT TYPE | UNIT NUMBER | L5 NETWORK ENGINEERING | UNIT TYPE | UNIT NUMBER | L5 CYBERSECURITY | UNIT TYPE | UNIT NUMBER | L5 CLOUD COMPUTING |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **30** | L5 Core Mandatory | **501 or 502** | Computing Research Project **OR** Applied Computing Project | L5 Core Mandatory | **501 or 502** | Computing Research Project **OR** Applied Computing Project | L5 Core Mandatory | **501 or 502** | Computing Research Project **OR** Applied Computing Project | L5 Core Mandatory | **501 or 502** | Computing Research Project **OR** Applied Computing Project |
| **15** | L5 Specialist Mandatory | **511** | Big Data Analytics and Visualisation | L5 Specialist Mandatory | **514** | Quality of Service (QoS) | L5 Specialist Mandatory | **515** | Secure System Design | L5 Specialist Mandatory | **518** | Cloud Data Centre Operations |
| **15** | L5 Specialist Mandatory | **512** | Advanced Data Analytics | L5 Specialist Mandatory | **515** | Secure System Design | L5 Specialist Mandatory | **517** | Applied Data Forensics | L5 Specialist Mandatory | **519** | Cloud System Architecture |
| **15** | L5 Specialist Mandatory | **513** | AI for Data Analysis | L5 Specialist Mandatory | **516** | Internet of Things | L5 Optional | | | L5 Optional | | |
| **15** | L5 Optional | | | L5 Optional | | | L5 Optional | | | L5 Optional | | |
| **15** | L5 Optional | | | L5 Optional | | | L5 Optional | | | L5 Optional | | |
| **15** | L5 Optional | | | L5 Optional | | | L5 Optional | | | L5 Optional | | |

## Level 4

| CREDITS | UNIT TYPE | UNIT NUMBER | L4 APPLIED ARTIFICIAL INTELLIGENCE |
|---|---|---|---|
| **15** | L4 Core Mandatory | **401** | Fundamentals of Computer Science |
| **15** | L4 Core Mandatory | **402** | Designing a Computing Project |
| **15** | L4 Core Mandatory | **403** | Professional Practice |
| **15** | L4 Specialist Mandatory | **420** | Core AI |
| **15** | L4 Specialist Mandatory | **421** | AI Automation and Digital Productivity |
| **15** | L4 Specialist Mandatory | **422** | Ethics and AI |
| **15** | L4 Optional | | |
| **15** | L4 Optional | | |

## Level 5

| CREDITS | UNIT TYPE | UNIT NUMBER | L5 APPLIED ARTIFICIAL INTELLIGENCE |
|---|---|---|---|
| **30** | L5 Core Mandatory | **501 or 502** | Computing Research Project **OR** Applied Computing Project |
| **15** | L5 Specialist Mandatory | **520** | Applied AI |
| **15** | L5 Specialist Mandatory | **521** | Generative and Conversational AI |
| **15** | L5 Optional | | |
| **15** | L5 Optional | | |
| **15** | L5 Optional | | |
| **15** | L5 Optional | | |

*Table 2: Level 4 and Level 5 Optional Units*

| CREDITS | UNIT NUMBER | L4 OPTIONAL UNITS |
|---|---|---|
| **15** | **423** | Agentic AI |
| **15** | **424** | Creative AI Coding |
| **15** | **425** | AI Programming |
| **15** | **426** | Digital Project Management |
| **15** | **427** | UI/UX Programming and Development |
| **15** | **428** | Maths for Computing |
| **15** | **429** | Responsive Website Development |
| **15** | **430** | Emerging Technologies |
| **15** | **431** | Operating Systems |
| **15** | **432** | System Architecture |
| **15** | **433** | Game Development |
| **15** | **434** | Green Computing |
| **15** | **435** | Client-Server Computer Systems |

| CREDITS | UNIT NUMBER | L5 OPTIONAL UNITS |
|---|---|---|
| **15** | **522** | Machine Learning Concepts |
| **15** | **523** | Natural Language Processing |
| **15** | **524** | Advanced AI Concepts |
| **15** | **525** | Computer Vision |
| **15** | **526** | Deep Learning |
| **15** | **527** | System Analysis and Design |
| **15** | **528** | Robotics |
| **15** | **529** | Game Design and Scripting |
| **15** | **530** | Information Security Management |
| **15** | **531** | E-Commerce and Strategy |
| **15** | **532** | Prototyping with AI |
| **15** | **533** | Transport Layer Engineering |
| **15** | **534** | Digital Sustainability |
| **15** | **535** | Virtual and Augmented Reality |
| **15** | **536** | Work-Based Learning |
| **15** | **537** | Fundamentals of Blockchain Technology |
| **15** | **538** | Applied Cloud Cryptography |
| **15** | **539** | Cloud Infrastructure Protection |
| **15** | **540** | Improving Organisational Performance |
| **15** | **541** | FinOps |
| **15** | **542** | API Design and Development |
| **15** | **543** | Data Structures and Algorithms |

# 3.0 The unit descriptor

The unit descriptor is how we define the individual units of study that make up a Higher National qualification. Students will complete the units included in the programme you offer at your centre.

You can use any of the unit descriptors listed in Section 4 Unit descriptors. We have described each part of the unit as follows.

*Table 3: Description of each part of the unit*

| | |
|---|---|
| **Unit title** | A general statement of what the unit will cover. |
| **Unit code** | The Ofqual unit reference number. |
| **Unit type** | There are three unit types:<br>• core (mandatory to all pathways)<br>• specialist (mandatory to specific pathways)<br>• optional (available to most pathways). |
| **Unit level** | All our Pearson BTEC Higher National units are at Levels 4 or 5. |
| **Credit value** | The credit value relates to the total qualification time (TQT) and unit learning hours (ULH). It is easy to calculate:<br>• 1 credit = 10 ULH<br>• 15 credits = 150 ULH.<br>To complete a Higher National Certificate or Higher National Diploma, students must achieve all of the credits required. Refer to Section *7.5 Calculating the final qualification grade* in the programme specification. |
| **Introduction** | Some general notes on the unit:<br>• setting the scene<br>• stating the purpose and aim, and<br>• outlining the topics to be learned and skills gained through the unit. |
| **Learning Outcomes** | These clearly explain what students will be able to do after completing the unit. There are usually four Learning Outcomes for each unit. |
| **Essential Content** | This section covers the content that students can expect to study as they work towards achieving their Learning Outcomes. |
| **Learning Outcomes and Assessment Criteria** | Tutors can refer to this table when grading assignments. The table connects the unit's Learning Outcomes with the student's work. Assignments can be graded at 'Pass' (P), 'Merit' (M) and 'Distinction' (D), depending on the quality of the student's work. |
| **Recommended Resources** | This section lists the resources that students should use to support their study for the unit. It includes books, journals and online material. The programme tutor may also suggest resources, particularly for local information. It may also contain delivery requirements, e.g., specific equipment, case study material and learning resources, depending on the subject. |

**Web resources – referencing**

Some units have web resources as part of their recommended resources list. Hyperlinking to these resources directly can cause problems, as their locations and addresses may change. To avoid this problem we only link to the main page of the website and signpost students and tutors to the section where the resource can be found. Therefore we have referenced web resources as follows:

[1] A link to the main page of the website

[2] The title of the site

[3] The section of the website where the resource can be found

[4] The type of resource it is, for example:

- research
- general reference
- tutorial
- training
- e-book
- report
- wiki
- article
- data set
- development tool
- discussion forum.

## Example

[1] learn.microsoft.com

[2] Microsoft Learn

[3] Design Principles for Technical Architectures

[4] (General reference)

Students and tutors must use a referencing system to cite and reference resources in an academic format.

 © Pearson Education Limited 2026

# 4.0 Unit descriptors

# Unit 401: Fundamentals of Computer Science

**Unit code:** **Y/652/3253**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

Modern organisations rely heavily on computer systems to support communication, decision-making and digital services across every sector. From business operations and healthcare systems to creative industries and public services, an understanding of how software, data, networks and intelligent systems interact is essential. This unit introduces students to the core principles that underpin computer science, providing a foundation for understanding how digital systems are designed, built and operated in real-world contexts.

The aim of this unit is to develop a clear and practical understanding of the fundamental concepts that shape modern computing systems. Students are introduced to key programming constructs, data structures and system principles that support effective problem-solving and technical reasoning. The unit is designed to build confidence with core computing ideas that underpin further study and professional practice within the technology sector.

The unit content covers a balanced range of foundational areas within computer science. Students begin by exploring programming fundamentals, focusing on logic, control structures and code quality. The unit then examines relational database concepts, including data modelling and basic SQL operations used to store and manage structured information. Core networking principles are introduced to explain how devices communicate securely and reliably, and the unit concludes with an introduction to artificial intelligence, outlining key AI concepts, data-driven models and ethical considerations.

On successful completion of this unit, students will have developed essential knowledge and skills that support progression to further Level 4 and Level 5 study, as well as entry-level roles within the digital sector. Students will be equipped to pursue roles such as Junior Software Developer, IT Support Technician, Database Assistant, Network Support Technician, Digital Technology Apprentice or to progress into specialist pathways including software development, cybersecurity, data analytics or artificial intelligence.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Apply fundamental programming constructs to develop simple applications

LO2 Explore the fundamental concepts of relational database design and development

LO3 Discuss the core principles of networking theory

LO4 Construct a basic AI process to ethically support simple decision-making tasks for an identified scenario.

## Essential Content

LO1 **Apply fundamental programming constructs to develop simple applications**

*Programming fundamentals:*

Data representation: variables e.g. value storage, memory reference; data types e.g. numeric types, text types, logical types; expressions e.g. arithmetic expressions, logical expressions

Operations and logic: operators e.g. arithmetic operators, comparison operators; input formats e.g. keyboard input, file input; output formats e.g. console output, screen display

Code quality: naming conventions e.g. meaningful identifiers, consistent casing; code readability e.g. indentation, commenting

Example data types: integers e.g. whole numbers, counters; strings e.g. text values, user input; Booleans e.g. true values, false values.

*Control structures and modularity:*

Program flow: sequence e.g. step order, execution flow; selection e.g. decision-making, condition checking; iteration e.g. loops, repetition

Program structure: functions e.g. reusable blocks, task routines; parameters e.g. input values, passed data

Scope management: local scope e.g. block variables, function variables; global scope e.g. shared variables, program variables

Selection constructs: if-else e.g. condition branches, true-false paths; switch e.g. case selection, option matching.

*Program design considerations:*

Requirements analysis: functional requirements e.g. required features, expected behaviour; constraints e.g. time limits, system limits

System context: platform characteristics e.g. operating systems, hardware types; resource limitations e.g. memory limits, processing limits

User focus: usability considerations e.g. user interfaces, accessibility

Development support: documentation basics e.g. comments, user guides; version control fundamentals e.g. file versions, change tracking.

*Testing and debugging basics:*

Test design: test cases e.g. planned tests, input scenarios; unit tests e.g. component tests, function tests; expected output tables e.g. result tables, comparison tables

Test data types: normal data e.g. typical values, valid input; boundary data e.g. edge values, limit cases; erroneous data e.g. invalid input, unexpected values

Debugging tools: assertions e.g. condition checks, validation checks; breakpoints e.g. pause points, inspection points; watch expressions e.g. variable tracking, value monitoring.

*Test-driven development:*

Test-driven development e.g. test-first approach, defined expected outcomes; test levels e.g. unit, integration, regression; automated testing e.g. repeatable execution, consistent results; refactoring e.g. code restructuring, maintainability improvement; incremental changes e.g. small updates; behaviour preservation e.g. maintaining functionality during change; architectural evolution e.g. gradual system improvement; version control alignment e.g. merge readiness.

LO2 **Explore the fundamental concepts of relational database design and development**

*Relational model foundations:*

Core relational components: entities e.g. customer records, product tables; attributes e.g. field names, data fields; relationships e.g. table links, data associations

Key identification: primary keys e.g. unique identifiers, record IDs; foreign keys e.g. reference fields, linking fields; cardinality e.g. single association, multiple association

Integrity and relationships: integrity constraints e.g. entity integrity, referential integrity; one-to-one e.g. user profiles, account details; one-to-many e.g. customers and orders, authors and books.

*Logical design and normalisation:*

Normal forms: first normal form e.g. atomic values, no repeating groups; second normal form e.g. partial dependency removal, composite key refinement; third normal form e.g. transitive dependency removal, dependency isolation

Design refinement: denormalisation considerations e.g. performance optimisation, query efficiency; schema refinement e.g. structure adjustment, table reorganisation

Key strategies: surrogate keys e.g. system-generated IDs, artificial identifiers; natural keys e.g. real-world identifiers, meaningful attributes.

*Structural database design:*

Modelling tools: entity-relationship diagrams e.g. visual schemas, relationship models; data dictionaries e.g. field definitions, metadata repositories; naming conventions e.g. consistent labels, standard identifiers

Data handling rules: NULL handling e.g. missing values, undefined fields; data validation rules e.g. value constraints, input restrictions.

*SQL operations and operational considerations:*

Data manipulation commands: SELECT e.g. data retrieval, record listing; INSERT e.g. new records, data entry; UPDATE e.g. record modification, value changes; DELETE e.g. record removal, data deletion

Query construction and joins: predicates e.g. WHERE conditions, filtering logic; joins e.g. table combination, relational linking; inner join e.g. matched records, shared keys; left join e.g. full left table, optional matches; subqueries e.g. nested queries, internal queries

Aggregation mechanisms: COUNT e.g. record totals, row counts; SUM e.g. numeric totals, accumulated values; aggregation functions e.g. grouped calculations, summary values

Performance and reliability: primary indexes e.g. index structures, search optimisation; ACID properties e.g. transaction consistency, data reliability.

LO3 **Discuss the core principles of networking theory**

*Reference models and topologies:*

Reference models: Open Systems Interconnection (OSI) model e.g. conceptual layers, data flow; Transmission Control Protocol/Internet Protocol (TCP/IP) suite e.g. protocol stack, internet communication

Network topologies: star topology e.g. central node, isolated faults

Advanced topologies: mesh topology e.g. multiple paths, resilience; hybrid topology e.g. combined structures, scalability.

*Addressing and naming:*

Internet Protocol (IP) addressing: IPv4 e.g. dotted notation, widespread use; IPv6 e.g. expanded address space, modern networks;

Address management: public addressing e.g. internet routing, global uniqueness; private addressing ranges e.g. internal networks, non-routable addresses;

Name resolution services: Domain Name System (DNS) hierarchy e.g. domain levels, name delegation; name resolution e.g. hostname translation, IP lookup

Network configuration services: Dynamic Host Configuration Protocol (DHCP) e.g. automatic configuration, IP assignment; Network Address Translation (NAT) e.g. address translation, address conservation.

*Core protocols and services:*

Transmission protocols: TCP e.g. reliable delivery, error checking; User Datagram Protocol (UDP) e.g. low overhead, fast transmission

Web protocols: Hypertext Transfer Protocol (HTTP) e.g. request-response, web communication; HTTPS e.g. encrypted web traffic, secure access

Network access technologies: Ethernet e.g. wired networking, local area networks; IEEE 802.11 wireless e.g. Wi-Fi standards, radio transmission

Traffic control mechanisms: routing fundamentals e.g. path selection, packet forwarding; switching fundamentals e.g. frame forwarding, Media Access Control (MAC) addressing

Security services: access control lists e.g. traffic filtering, permission rules; Transport Layer Security (TLS) e.g. encrypted communication, authentication.

*Network design considerations:*

Performance factors: throughput e.g. data transfer rate, capacity; latency e.g. transmission delay, response time

Scalability and availability: scalability e.g. growth support, user expansion; availability e.g. uptime, service continuity; redundancy concepts e.g. backup links, duplicate components

Operational monitoring: monitoring fundamentals e.g. network observation, status checks; performance monitoring e.g. metric tracking, usage analysis.

*Core network threats and countermeasures:*

Threat categories: internal threats e.g. human error, misuse of access; external threats e.g. unauthorised access, network attacks

Infrastructure risks: hardware threats e.g. device failure, physical damage; software threats e.g. vulnerabilities, unpatched systems

Preventive controls: antivirus e.g. malware detection, threat removal; firewalls e.g. traffic filtering, network boundaries

Organisational controls: policies and procedures e.g. usage rules, security guidance; staff training e.g. awareness building, safe behaviour.

LO4 **Construct a basic AI process to ethically support simple decision-making tasks for an identified scenario**

*AI paradigms and tasks:*

Learning approaches: supervised learning e.g. labelled data, training examples; unsupervised learning e.g. pattern groups, data clusters; reinforcement learning e.g. feedback signals, behaviour learning; examples: Gmail spam filter, Amazon product recommendations, DeepMind Atari agents

Application types: classification tasks e.g. category labels, grouped outputs; recommendation systems e.g. suggested items, ranked options; anomaly detection e.g. unusual behaviour, outlier cases

AI scope: narrow AI e.g. task-specific systems, defined outputs; applied AI tasks e.g. spam detection, image tagging.

*Data, models and evaluation:*

Data elements: datasets e.g. collections, sample records; features e.g. attributes, input values; labels e.g. target categories, expected outputs

Data splits: training sets e.g. learning examples, labelled samples; test sets e.g. unseen samples, evaluation data

Outcome measures: accuracy e.g. correct results, matching outputs; precision e.g. relevant positives, correct positives; recall e.g. captured cases, found positives; examples: MLflow, IBM Watson Studio, Weights & Biases.

*Introductory AI processes:*

Data preparation: data collection e.g. data gathering, sample records; data preparation e.g. cleaned values, organised inputs

Model process (high-level): training stage e.g. learning patterns, example exposure; evaluation stage e.g. performance checks, result review; examples: TensorFlow, Google Colab, PyTorch.

*AI prompts:*

Prompt construction: prompt structure e.g. task instructions, context setting; input definition e.g. clear queries, structured inputs; output guidance e.g. format instructions, response constraints

Iterative refinement: prompt testing e.g. trial inputs, response checking; prompt adjustment e.g. wording changes, parameter tuning; output evaluation e.g. relevance checks, accuracy review, human oversight.

*Responsible use of AI:*

Ethical considerations: bias e.g. unfair patterns, skewed data; fairness e.g. equal outcomes, balanced treatment; responsible use e.g. appropriate application, safe operation; examples: IBM AI Fairness 360, Microsoft Responsible AI Tools, Google What-If Tool

Governance principles: legislation e.g. General Data Protection Regulation (GDPR), Equality Act (2010); risk management; knowledge currency, emerging developments, evolving practice.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Apply fundamental programming constructs to develop simple applications | | |
| **P1** Design a simple application using a selection of design techniques to meet identified requirements.<br>**P2** Develop a simple application using a selection of identified programming constructs to meet identified requirements. | **M1** Evaluate, using the results of application testing and debugging, the effectiveness of the developed program in meeting identified requirements. | **D1** Justify the decisions made in the design and development of the application in supporting its ability to effectively meet identified requirements. |
| **LO2** Explore the fundamental concepts of relational database design and development | | |
| **P3** Design a simple relational database that demonstrates basic design concepts and meets identified requirements.<br>**P4** Develop a simple relational database that demonstrates basic relational database concepts and meets identified requirements. | **M2** Evaluate, using the results of database testing, the effectiveness of the developed relational database in meeting identified requirements. | **D2** Justify the decisions made in the design and development of the relational database in supporting its ability to effectively meet identified requirements. |
| **LO3** Discuss the core principles of networking theory | | |
| **P5** Discuss, using a selection of real-world examples, the core principles of networking theory.<br>**P6** Design a simple, secure network and associated usage policies for an identified scenario that meets organisational requirements. | **M3** Analyse the network design to identify strengths and areas for improvement in addressing the requirements of an identified scenario. | **D3** Evaluate the effectiveness of the network design in comprehensively addressing the requirements of an identified scenario. |

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Construct a basic AI process to ethically support simple decision-making tasks for an identified scenario | | |
| **P7** Explain, using a selection of real-world examples, the core paradigms used in AI process development.<br><br>**P8** Construct a basic AI process to ethically support simple decision-making tasks for an identified scenario. | **M4** Assess the effectiveness of the developed AI process in meeting the requirements of an identified scenario. | **D4** Evaluate the effectiveness of the developed AI process compared to traditional programming methodologies in meeting the requirements of an identified scenario. |

## Recommended Resources

### Textbooks

Brookshear, J.G. and Brylow, D. (2024) *Computer Science: An Overview*. 13th edn. Harlow: Pearson.

Coronel, C. and Morris, S. (2019) *Database Systems: Design, Implementation, and Management*. 13th edn. Boston, MA: Cengage Learning.

Kurose, J.F. and Ross, K.W. (2023) *Computer Networking: A Top-Down Approach*. 9th edn. Harlow: Pearson.

Russell, S. and Norvig, P. (2021) *Artificial Intelligence: A Modern Approach*. 4th edn. Harlow: Pearson.

Sommerville, I. and Tilley, S. (2024) *Software Engineering*. 11th edn. Harlow: Pearson.

Stallings, W. (2021) *Data and Computer Communications*. 10th edn. Harlow: Pearson.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/computer-science-fundamentals/programming-tutorial/ | GeeksforGeeks<br>Programming Tutorial \| Introduction, Basic Concepts, Getting started, Problems<br>(Tutorial) |
| www.tutorialspoint.com/computer_programming/index.htm | TutorialsPoint<br>Computer Programming Tutorial<br>(Tutorial) |
| www.geeksforgeeks.org/dbms/introduction-of-database-normalization/ | GeeksforGeeks<br>Introduction to Database Normalization<br>(General reference) |
| www.khanacademy.org/computing/computer-science/sql | Khan Academy<br>Intro to SQL: Querying and managing data<br>(Tutorial) |
| www.postgresql.org/docs/current/ddl-basics.html | PostgreSQL Documentation<br>Chapter 5 Data Definition<br>(General reference) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.cloudflare.com/learning/network-layer/what-is-the-osi-model/ | Cloudflare Learning<br>What is the OSI Model?<br>(General reference) |
| www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13769-5.html | Cisco<br>TCP/IP and Routing Fundamentals<br>(General reference) |
| www.ibm.com/think/topics/artificial-intelligence | IBM<br>What is artificial intelligence (AI)?<br>(Article) |
| learn.microsoft.com/en-us/training/modules/get-started-ai-fundamentals/ | Microsoft Learn<br>Introduction to AI Concepts<br>(General reference) |

**Journals and articles**

Miedema, D. (2023) 'Toward a Fundamental Understanding of SQL Education.' *In ICER '23 Proceedings*, pp. 64–68. Available at: doi.org/10.1145/3568812.3603454.

Radanliev, P. (2024) 'Artificial intelligence: reflecting on the past and looking towards the next paradigm shift'. *Journal of Experimental & Theoretical Artificial Intelligence*. 37(7), pp.1045-1062. Available at: doi.org/10.1080/0952813X.2024.2323042.

Rak, J. (2024) 'Fundamentals of Resilience of Communication Networks and Networked Systems'. *In Resilient Routing in Communication Networks* (pp. 3–20). Springer. Available at: doi.org/10.1007/978-3-031-64657-7_1.

Sun, C., Yang, S., & Becker, B. (2024) 'Debugging in Computational Thinking: A meta-analysis on the effects of interventions on debugging skills.' *Journal of Educational Computing Research*, 62(4), 1087–1121. Available at: doi.org/10.1177/07356331241227793.

**Links**

This unit links to the following related units:

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 405: Core Software Design and Development*

*Unit 406: Core Software Testing*

*Unit 409: Core Cybersecurity*

*Unit 410: Database Engineering*

*Unit 411: Core Data Analytics*

*Unit 413: Core Networking*

*Unit 416: Core Cloud Computing*

*Unit 420: Core AI.*

 © Pearson Education Limited 2026

# Unit 402: Designing a Computing Project (Pearson-set)

**Unit code:** **A/652/3254**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

This unit is assessed through a **Pearson-set assignment**. The project brief will be set by the centre, based on a theme provided by Pearson (this will change annually). The theme and chosen project within the theme will enable students to explore and examine a relevant and topical aspect of computing in the context of a professional environment.

The purpose of the unit is to develop students' confidence in approaching a computing project in a structured and professional way. Students are required to interpret a project brief, research relevant technical and organisational factors, and begin forming reasoned design ideas. By working through this process, students develop core project-related behaviours that are expected of entry-level computing practitioners.

The aim of the unit is to give students the opportunity to apply research, analysis and design skills to a computing scenario, using evidence to inform decisions. Students will explore different design approaches and techniques, justify their choices, and communicate their ideas clearly to others. The unit also develops students' ability to reflect on feedback and refine their work in line with stakeholder expectations.

On successful completion of the unit, students will be able to investigate a computing problem, propose an appropriate design solution, and present their ideas in a professional manner. They will develop transferable skills in problem-solving, communication and decision-making that support progression into computing-related roles such as Junior Software Developer, Data Analyst, Network Technician, Cybersecurity Technician and Business Analyst Trainee.

***Please refer to the accompanying Pearson-set Assignment Guide and the Theme Release document for further support and guidance on the delivery of the Pearson-set unit.**

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Conduct small-scale research in relation to a given computing project brief

LO2 Develop a design based on the requirements of a given computing project brief

LO3 Present the design of a given computing project brief to an intended audience

LO4 Refine the design of a given computing project brief based on stakeholder feedback.

## Essential Content

LO1 **Conduct small-scale research in relation to a given computing project brief**

*Project brief interpretation:*

Purpose and aims: project purpose e.g. business need, technical objective; aims e.g. functional goal, quality target; expected outcomes e.g. working prototype, documented design

Context and users: end-user needs e.g. accessibility needs, usability needs; domain context e.g. organisational setting, technical environment; data e.g. datasets, data sources; network e.g. local network, cloud network; software e.g. applications, platforms

Scope and constraints: project boundaries e.g. feature limits, time limits; constraints e.g. budget limits, resource limits; acceptance considerations e.g. test criteria, approval conditions

Legal and ethical constraints: legal considerations e.g. General Data Protection Regulation (GDPR) legislation, intellectual property rights; ethical considerations e.g. responsible data use, fairness and bias; compliance requirements e.g. organisational policies, professional standards; user impact considerations e.g. privacy implications, accessibility obligations.

*Research scope definition:*

Research focus: key questions e.g. problem cause, user requirements; focus areas e.g. performance issues, security needs; relevance e.g. project alignment, stakeholder value

Assumptions and unknowns: assumptions e.g. user behaviour, system availability; unknowns e.g. data quality, scalability limits; solution spaces e.g. software solutions, infrastructure solutions

Problem alignment: computing problems e.g. performance bottlenecks, security risks; solution spaces e.g. architectural options, technology choices; relevance e.g. industry practice, project fitness.

*Information sources:*

Primary sources: primary sources e.g. stakeholders, users; stakeholder inputs e.g. requirements feedback, expectations; interviews e.g. user interviews, expert interviews; user descriptions e.g. personas, usage scenarios

Secondary sources: secondary literature e.g. published materials, documented research; academic literature e.g. journals, conference papers; standards documentation e.g. International Organization for Standardization (ISO) standards, industry standards

Industry materials: vendor specifications e.g. software manuals, hardware datasheets; industry reports e.g. market analysis, trend reports; white papers e.g. technical briefs, solution papers; vendor-neutral resources e.g. open standards, protocols, open source documentation, API specifications; training and certification frameworks e.g. vendor-neutral courses, professional certifications.

*Data and evidence types:*

System evidence: system behaviour information e.g. response behaviour, fault behaviour; performance information e.g. latency metrics, throughput metrics

Technical descriptions: infrastructure descriptions e.g. server layouts, network topology; existing solution examples e.g. deployed systems, reference solutions

Data and users: dataset characteristics e.g. size, structure; user profiles e.g. roles, access levels; workflow information e.g. process flows, task sequences.

*Research quality factors:*

Source quality: credibility e.g. authoritative authors, trusted sources; accuracy e.g. correct data, verified facts; recency e.g. current standards, recent publications

Evaluation criteria: objectivity e.g. neutral perspective, balanced viewpoint; relevance e.g. project relevance, domain relevance; triangulation across multiple sources e.g. cross-checking, source comparison.

LO2 **Develop a design based on the requirements of a given computing project brief**

*Requirements understanding:*

Requirements definition: functional requirements e.g. system behaviour, required features; non-functional requirements e.g. scalability, performance; stakeholder expectations e.g. user needs, business priorities

Constraints and limitations: technical constraints e.g. platform limits, compatibility issues; compliance e.g. security requirements, regulatory constraints; reliability e.g. availability needs, fault tolerance.

*Design principles:*

Structural principles: modularity e.g. component separation, reusable modules; structure e.g. layered design, logical grouping; abstraction e.g. simplified models, interface hiding

Quality principles: simplicity e.g. reduced complexity, clear logic; interoperability e.g. system integration, standard interfaces; maintainability e.g. ease of updates, code clarity.

*Design representation methods:*

Conceptual views: conceptual models e.g. high-level diagrams, system overviews; logical architectures e.g. layered architecture, component layout; data structures e.g. tables, data entities

Visual artefacts: system diagrams e.g. architecture diagrams, flow diagrams; structural schemas e.g. data schemas, system blueprints; interface maps e.g. screen maps, navigation flows; modelling approaches e.g. whiteboarding, sketch diagrams; collaborative modelling e.g. shared diagrams, team consensus building; evolving representations e.g. iterative diagram updates, design refinement

Process views: process flows e.g. workflow diagrams, task sequences; component diagrams e.g. module interaction, dependency mapping; interface definitions e.g. input formats, output formats

Structured problem-solving methods e.g. affinity mapping, impact mapping; iterative improvement approaches e.g. plan-do-check-act cycles, incremental refinement; logical reasoning e.g. hypothesis formulation, solution testing; issue identification e.g. root cause exploration, scenario analysis

Supporting documentation: annotated design notes e.g. design explanations, rationale notes; interface maps e.g. interaction mapping, data exchange mapping; structural schemas e.g. component structure, logical layout.

*Design components:*

System elements: inputs e.g. user input, data input; outputs e.g. reports, system responses; data pathways e.g. data flow, processing sequence

System boundaries: control flows e.g. decision logic, process control; system boundaries e.g. internal systems, external systems; user interaction points e.g. user interfaces, access points.

*Design justification factors:*

Feasibility factors: feasibility e.g. technical feasibility, operational feasibility; resource considerations e.g. hardware resources, software resources; risk considerations e.g. technical risks, deployment risks

Future alignment: alignment with requirements e.g. requirement coverage, stakeholder needs; future scalability e.g. system growth, expansion capacity.

LO3 **Present the design of a given computing project brief to an intended audience**

*Audience profiling:*

Audience types: technical audience expectations e.g. system detail, technical accuracy; non-technical audience expectations e.g. clarity, high-level explanation; mixed-discipline stakeholders e.g. managers, technical staff

Communication needs: communication needs e.g. information depth, explanation level; audience understanding levels e.g. prior knowledge, experience; opportunities for clarification e.g. questions, feedback points.

*Presentation structure:*

Context and rationale: summary of project context e.g. project background, problem overview; rationale for design decisions e.g. design reasoning, justification; key features e.g. system features, core functions

Requirements and constraints: compliance with requirements e.g. requirement coverage, specification alignment; explanation of constraints e.g. technical limits, resource limits; key features e.g. functional elements, non-functional aspects.

*Communication clarity elements:*

Language and sequencing: terminology appropriate to audience e.g. technical terms, plain language; logical sequencing e.g. ordered structure, step progression; explanation e.g. structured explanation, supporting narration

Visual support: visual support e.g. diagrams, charts; annotated views e.g. labels, callouts; simplified representations e.g. abstractions, overview diagrams.

*Visual presentation:*

Presentation materials: slides e.g. presentation slides, briefing slides; logical diagrams e.g. flow diagrams, architecture diagrams; requirement-to-design mapping sheets e.g. traceability tables, alignment charts

Supporting materials: system representations e.g. system views, component views; summary tables e.g. comparison tables, overview tables; logical diagrams e.g. dependency diagrams, interaction diagrams.

*Professional presentation techniques:*

Structure and consistency: coherence e.g. clear narrative, consistent message; structure e.g. section organisation, ordered content; consistency e.g. formatting consistency, terminology consistency

Clarity and accessibility: emphasis e.g. key points, highlights; use of explanation aids e.g. examples, visuals; readability e.g. clear text, legible layout; accessibility considerations e.g. colour contrast, readable fonts.

*Engagement considerations:*

Audience interaction: audience understanding levels e.g. comprehension level, familiarity; opportunities for clarification e.g. discussion points, questions; visual emphasis e.g. highlighted content, focal elements

Presentation layout: layout and formatting considerations e.g. slide layout, spacing; visual emphasis e.g. colour use, headings; readability e.g. font size, spacing.

LO4 **Refine the design of a given computing project brief based on stakeholder feedback**

*Feedback characteristics:*

Functional and usability feedback: functional improvement suggestions e.g. feature changes, behaviour changes; usability concerns e.g. navigation issues, accessibility issues; domain-specific accuracy considerations e.g. correctness issues, domain rules

Performance and security feedback: performance observations e.g. speed issues, resource usage; security risks e.g. vulnerabilities, access weaknesses; data issues e.g. data quality, data integrity

*Feedback analysis:*

Evaluation focus: relevance to project goals e.g. requirement alignment, objective support; feasibility e.g. technical feasibility, resource feasibility; alignment with constraints e.g. budget limits, time limits

Decision-making: cost–benefit considerations e.g. effort versus value, trade-offs; prioritisation of changes e.g. critical changes, optional changes; relevance to project goals e.g. scope fit, design intent.

*Refinement factors:*

Design optimisation: opportunities for improvement e.g. simplified processes, improved usability; optimisation of design elements e.g. more efficient data flows, improved system structure; maintenance of design coherence e.g. consistent components, integrated functionality

Design impact: impact of proposed changes e.g. functionality, usability; dependencies between design elements e.g. component interactions, data flows; unintended consequences e.g. increased complexity, reduced performance.

*Refinement Artefacts:*

Updated visuals: updated diagrams e.g. revised architectures, updated flows; enhanced logical structures e.g. refined logic, improved organisation; clarified representations e.g. clearer diagrams, clearer models

Supporting documentation: improved supporting documentation e.g. updated notes, clarified explanations; design rationale e.g. reasons for changes, links to stakeholder feedback; consolidated records e.g. final specifications, supporting annotations.

*Versioning and traceability:*

Change documentation: documentation of changes e.g. change logs, revision records; rationale for adjustments e.g. justification notes, decision reasons; traceability to stakeholder comments e.g. feedback mapping, response tracking

Comparison tracking: comparison to original design e.g. before-and-after views, version comparison; documentation of changes e.g. updated records, annotated revisions; rationale for adjustments e.g. design reasoning, improvement justification.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Conduct small-scale research in relation to a given computing project brief | | |
| **P1** Review the user needs, legal and ethical considerations of a given computing project brief.<br><br>**P2** Conduct, using a range of research methodologies and data collection techniques, a small-scale research project in relation to a given computing project brief. | **M1** Justify the selection of the research methods and data collection techniques used in conducting the small-scale research of the given computing project brief. | **D1** Evaluate the effectiveness of the chosen research methods and data collection techniques used in conducting the small-scale research of the given computing project brief. |

| Pass | Merit | Distinction |
|---|---|---|
| **LO2** Develop a design based on the requirements of a given computing project brief | | **LO2, LO3 and LO4**<br>**D2** Evaluate the effectiveness of the original and optimised design in meeting the requirements of the computing project brief, considering the decisions made in the optimisation of the original design. |
| **P3** Develop a design that addresses the requirements of the given computing project brief.<br>**P4** Review the effectiveness of the design in meeting the requirements of the given computing project brief. | **M2** Justify the decisions made in the design of the given computing project brief including why alternative options were rejected. | |
| **LO3** Present the design of a given computing project brief to an intended audience | | |
| **P5** Develop an appropriate presentation containing the design of a given computing project brief for an intended audience.<br>**P6** Present the design of the given computing project brief to an intended audience. | **M3** Analyse the effectiveness of the presentation in communicating the design of the given computing project brief to an intended audience. | |
| **LO4** Refine the design of a given computing project brief based on stakeholder feedback | | |
| **P7** Discuss, with a range of relevant stakeholders, the effectiveness of the design in relation to the given computing project brief.<br>**P8** Review the results of the stakeholder feedback to identify areas where the design of the given computing project brief could be refined or improved. | **M4** Refine, using the results of the feedback analysis, the design for a given computing project brief. | |

## Recommended Resources

### Textbooks

Ambler, S.W. and Lines, M. (2022) *Choose Your WoW!: A Disciplined Agile Approach to Optimizing Your Way of Working*. 2nd edn. Newtown Square, PA: Project Management Institute.

Dennis, A., Wixom, B.H. and Roth, R.M. (2021) *Systems Analysis and Design*. 8th edn. Hoboken, NJ: Wiley.

Goodwin, K. (2023) *Designing for the Digital Age: How to Create Human-Centred Products and Services*. 2nd edn. Indianapolis, IN: Wiley.

Wysocki, R.K. (2019) *Effective Project Management: Traditional, Agile, Extreme, Hybrid*. 8th edn. Indianapolis, IN: Wiley.

Cadle, J., Paul, D. and Turner, P. (2023) *Business Analysis Techniques: 99 Essential Tools for Success*. 3rd edn. Swindon: BCS Learning & Development Ltd.

O'Leary, D.A., O'Leary, T.J. and O'Leary, L.I. (2021). *Computing Essentials: Making IT Work for You*. New York: McGraw-Hill Education.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.gov.uk/service-manual/user-research/start-by-learning-user-needs | UK Government Digital Service<br>Understanding User Needs in Service Design<br>(Article) |
| www.nibusinessinfo.co.uk/content/manage-business-risks | NI Business Info<br>Risk Assessment and Risk Management<br>(General reference) |
| projectdelivery.gov.uk/teal-book/home/part-f-solution-delivery/chapter-31-user-need-and-requirements/ | UK Government Project Delivery<br>Chapter 31: User needs and requirements<br>(General reference) |
| developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Design_and_accessibility | Mozilla Developer Network<br>Designing for Accessibility<br>(Tutorials) |

| | |
|---|---|
| www.ibm.com/think/architectures | IBM Technology<br>Architecture Patterns and Design Guidance<br>(Article) |
| learn.microsoft.com/en-us/azure/architecture/guide/design-principles/ | Microsoft Learn<br>Design Principles for Technical Architectures<br>(General reference) |
| www.nngroup.com/articles/ux-writing-study-guide/ | Nielsen Norman Group<br>Communicating Design Clearly to Users<br>(Article) |
| www.smashingmagazine.com/2013/02/designing-great-feedback-loops/ | Smashing Magazine<br>User Experience Feedback Loops<br>(Article) |
| www.atlassian.com/team-playbook/plays/retrospective | Atlassian Team Playbook<br>Using Retrospectives to Improve Solutions<br>(Tutorials) |

## Journals and articles

Abumosa, M. (2024) 'University Students' Perspectives on the Use of Interactive Presentation Technologies', *International Journal of Technology in Education and Science,* 8(4), 645–667. Available at: doi.org/10.46328/ijtes.579.

Brown, A., Goldstein, M. H., Clay, J., Demirel, H. O., Li, X., & Sha, Z. (2024) 'A Study on Generative Design Reasoning and Students' Divergent and Convergent Thinking', *Journal of Mechanical Design*, 146(3). Available at: doi.org/10.1115/1.4064564.

Gardner, T., Leonard, H. C., Waite, J., & Sentance, S. (2022) 'What do We Know about Computing Education for K-12 in Non-formal Settings? A Systematic Literature Review of Recent Research', *Proceedings of the ACM Conference on International Computing Education Research (ICER)*. Available at: doi.org/10.1145/3501385.3543960.

Wacnik, P., Daly, S. R., & Verma, A. (2025) 'Participatory design: A systematic review and insights for future practice', *Design Science*, 11, e21.
Available at: doi.org/10.1017/dsj.2025.10009.

## Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 405: Core Software Design and Development*

*Unit 406: Core Software Testing*

*Unit 409: Core Cybersecurity*

*Unit 410: Database Engineering*

*Unit 411: Core Data Analytics*

*Unit 413: Core Networking*

*Unit 416: Core Cloud Computing*

*Unit 420: Core AI.*

# Unit 403: Professional Practice

**Unit code:** D/652/3255

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Professional practice is a fundamental requirement for anyone working in modern computing roles, where technical ability must be matched with clear communication, ethical awareness and responsible working behaviours. Computing professionals operate across a wide range of environments, from software development and data analysis to networks, cybersecurity and cloud services, often working with mixed technical and non-technical teams. This unit introduces students to the professional standards, behaviours and communication practices needed to operate effectively within contemporary digital workplaces.

The aim of this unit is to develop students' understanding of professional conduct within the computing sector. It focuses on the communication methods, collaborative approaches and professional responsibilities that underpin effective working practice. The unit also introduces students to the legal, ethical and sustainability considerations that influence decision-making in computing roles, helping them to understand their responsibilities to organisations, users and society.

The unit covers key aspects of professional communication and collaboration, including the use of appropriate tools, documentation and presentation techniques. Students explore workplace expectations, stakeholder awareness and the importance of professional behaviour in different organisational contexts. The unit also requires students to plan and deliver a professional workshop on a technical computing topic, and to develop a Continuing Professional Development (CPD) plan that supports personal skills development, employability and progression within the computing sector.

On successful completion of this unit, students will have developed transferable professional skills that are essential across the computing industry. These include clear communication, teamwork, reflective practice and professional judgement. The unit supports progression into a range of entry-level and technician-level roles, such as Junior Software Developer, IT Support Technician, Data Analyst, Network Technician, Cybersecurity Analyst, or Cloud Support Engineer, as well as progression to further Level 5 study in computing disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore a range of effective communication and collaboration skills in a professional context

LO2 Explain the principles of professional, legal, and ethical practice in computing

LO3 Deliver a workshop on an identified technical computing topic that shows professionalism

LO4 Present a CPD plan on personal development and employability skills to an intended audience.

## Essential Content

LO1 **Explore a range of effective communication and collaboration skills in a professional context**

*Professional communication methods:*

Communication tools: platforms and applications e.g. email tools, messaging platforms; collaboration tools e.g. shared workspaces, project boards; conferencing tools e.g. video platforms, screen sharing tools; content creation tools e.g. presentation software, diagram tools

Communication channels: email e.g. project update, meeting request; instant messaging e.g. direct message, group chat; video conferencing e.g. virtual meeting, screen share

Communication records: meeting minutes e.g. action log, decision log; technical documentation e.g. user guide, Application Programming Interface (API) guide; reports e.g. status report, incident report

Communication delivery: presentations e.g. slide deck, demo; diagrams e.g. flowchart, architecture diagram; tone e.g. formality, professionalism; non-verbal cues e.g. eye contact, posture

Communication in organisational contexts: communication approaches e.g. internal teams, external stakeholders; organisational cultures e.g. corporate environments, client-facing settings.

*Collaboration approaches:*

Team structure: teamwork e.g. co-operation, co-ordination; roles e.g. developer, tester; responsibilities e.g. task ownership, accountability; conflict resolution e.g. mediation, compromise

Workflow management: co-ordination e.g. scheduling, synchronisation; task allocation e.g. assignments, workload balance; meeting facilitation e.g. agenda, timekeeping; decision-making e.g. consensus, voting

Control methods: peer review e.g. code review, document review; version control e.g. commit history, branching; shared documentation e.g. shared notes, shared templates.

*Professional context considerations:*

Workplace standards: workplace expectations e.g. punctuality, responsiveness; organisational culture e.g. values, norms; professionalism e.g. conduct, etiquette; confidentiality e.g. access control, non-disclosure; policy development e.g. creation of security policies aligned to legislation, regulation and organisational requirements, security awareness campaigns

Communication quality: clarity e.g. plain language, structure; accuracy e.g. correctness, completeness; accessibility e.g. readability, captions; inclusivity e.g. respectful language, bias awareness

Stakeholder focus: audience analysis e.g. user needs, technical level; stakeholder management e.g. prioritisation, expectation setting; time management e.g. deadlines, scheduling.

*Tools supporting communication and collaboration:*

Communication platforms: email clients e.g. Outlook, Gmail; messaging platforms e.g. Teams, Slack; video conferencing e.g. Zoom, Meet; shared calendars e.g. Outlook Calendar, Google Calendar

Content management: document repositories e.g. SharePoint, Google Drive; shared workspaces e.g. Teams site, SharePoint site; wikis e.g. Confluence, Wiki pages

Delivery management: collaboration suites e.g. Microsoft 365, Google Workspace; issue trackers e.g. Jira, Azure DevOps; whiteboards e.g. Microsoft Whiteboard, Miro; version control systems e.g. Git, Subversion.

*Interpersonal and professional behaviours:*

Communication behaviours: active listening e.g. questioning, summarising; respect e.g. courtesy, professionalism; courtesy e.g. politeness, patience; assertiveness e.g. confidence, clarity

Performance behaviours: adaptability e.g. flexibility, role change; resilience e.g. persistence, coping; initiative e.g. proactivity, ownership; accountability e.g. responsibility, follow-through

Development behaviours: professionalism e.g. conduct, integrity; reflection e.g. self-review, feedback; continuous improvement e.g. iteration, upskilling; operational ownership: ownership of deployed systems e.g. responsibility for running services, maintaining availability, responding to issues; accountability culture: accountability e.g. shared responsibility, ownership of decisions, action follow ups; failure response: learning e.g. incident review, reflective learning; CPD: continuous improvement e.g. iterative enhancement, applying lessons learned.

LO2 **Explain the principles of professional, legal, and ethical practice in computing**

*Professional principles:*

Professional standards: codes of conduct e.g. organisational codes, professional body rules; professional standards e.g. British Computer Society (BCS), ISC2, ISACA codes of conduct; principles e.g. integrity, confidentiality, professional competence, lawful practice e.g. recognised frameworks, industry benchmarks

Professional responsibility: professional responsibility e.g. duty of care, role accountability; professional accountability e.g. task ownership, decision justification

Workplace professionalism: role boundaries e.g. job scope, responsibility limits; workplace professionalism e.g. conduct expectations, behaviour standards, professional communication; professional conduct e.g. workplace behaviour, responsibility; secure working practices: secure environments e.g. device security, password management

Professional development: ethical professionalism e.g. ethical awareness, responsible behaviour; professional judgement e.g. informed decisions, risk awareness; professional currency, evolving practice.

*Legal frameworks:*

Data protection law: data protection law e.g. UK General Data Protection Regulation (GDPR), data protection principles; lawful processing e.g. legal basis, authorised use, permitted purposes

Information rights: intellectual property law e.g. ownership rights, licensing rules; copyright e.g. protected works, usage permissions, content restrictions

Governance requirements: regulatory compliance e.g. legal adherence, policy alignment; records management e.g. record storage, access control; safe data handling e.g. data protection, secure storage.

*Ethical considerations:*

Ethical values: fairness e.g. unbiased decisions, equal treatment; transparency e.g. clear processes, open communication, system clarity;

AI bias: output bias e.g. unequal outcomes, skewed predictions, discriminatory results; representation bias e.g. underrepresentation of groups, exclusion of minority data, sampling imbalance; confirmation bias e.g. reinforcement of existing patterns, feedback loops, biased recommendations

Ethical data use e.g. responsible data handling, lawful processing; societal impact e.g. automation effects, organisational change; implications of emerging technologies e.g. AI, automation; data accountability e.g. stewardship, responsible decision-making; transparency considerations e.g. explainability, user awareness

Social responsibility: inclusivity e.g. accessible systems, diverse users; societal impact e.g. social consequences, public trust

Responsible technology use: responsible technology use e.g. appropriate application, safe design; digital citizenship e.g. online responsibility, ethical behaviour.

*Sustainability and computing practice:*

Environmental impact: environmental impact e.g. energy use, emissions effects, impact of AI data centres; carbon awareness e.g. footprint recognition, reduction awareness

Sustainable technology: sustainable computing e.g. efficient systems, reduced waste; green IT e.g. low-energy hardware, eco-friendly systems, efficient devices

Resource management: energy efficiency e.g. reduced consumption, optimised performance; electronic waste e.g. recycling processes, disposal methods

Asset life cycle: hardware life cycle e.g. usage stages, replacement cycles, impact of AI hardware requirements; sustainable design e.g. efficiency planning, long-term usability.

*Risk and compliance awareness:*

Risk types: data misuse e.g. unauthorised access, improper handling; security risk e.g. system breaches, cyber threats

Compliance control: regulatory breach e.g. policy violations, legal failure; compliance monitoring e.g. audits, regular checks

Security considerations: security culture assessment e.g. evaluating organisational behaviours, awareness and adherence to security practices; security awareness activities e.g. awareness campaigns, user guidance, promotion of secure working practices

Risk management: risk assessment e.g. analysis, impact evaluation; incident response e.g. issue handling, recovery actions

Organisational governance: documentation e.g. policy records, procedural guides; organisational policy e.g. internal rules, compliance standards.

LO3 **Deliver a workshop on an identified technical computing topic that shows professionalism**

*Technical topic identification:*

Professional relevance: professional relevance e.g. workplace application, industry context; audience relevance e.g. learner needs, user expectations

Topic scope: clarity of scope e.g. defined boundaries, focused coverage; reliability of information sources e.g. trusted websites, authoritative references.

*Workshop structure:*

Content organisation: topic introduction e.g. session overview, objectives outline; key concept explanation e.g. core ideas, simplified principles; example scenarios e.g. real tasks, familiar situations

Delivery flow: structured progression e.g. logical sequence, step-by-step order; timing control e.g. pacing, session management.

*Use of technical evidence:*

Supporting materials: definitions e.g. key terms, precise meanings; models e.g. simplified diagrams, conceptual frameworks

Visual support: visuals e.g. slides, diagrams; structured explanations e.g. ordered points, clear breakdowns.

*Professional delivery expectations:*

Communication quality: clarity of verbal communication e.g. clear speech, appropriate language; professional tone e.g. formal style, respectful delivery; audience engagement e.g. questions, interaction

Presentation skills: confidence e.g. assured delivery, steady manner; pacing e.g. appropriate speed, time awareness.

*Workshop preparation:*

Preparation methods: organisation of materials e.g. ordered slides, prepared notes; selection of examples e.g. relevant cases, clear illustrations

Accessibility considerations: adaptation for accessibility e.g. readable text, inclusive delivery.

LO4 **Present a CPD plan on personal development and employability skills to an intended audience**

*Personal development awareness:*

Self-assessment: strengths identification e.g. technical strengths, communication strengths; weaknesses identification e.g. skill gaps, confidence gaps, experience gaps; self-reflection e.g. learning journals, feedback reflection

Skills analysis: skills audit e.g. skill inventories, capability checklists; competency mapping e.g. role requirements, development gaps; learning needs analysis e.g. training needs, knowledge gaps

Impact of AI: CPD acceleration e.g. faster skills acquisition, reduced time to competence; content access e.g. AI-curated resources, intelligent searching; professional expectations e.g. continuous upskilling, currency of technical knowledge, alignment with emerging technologies, evidence of CPD activity

Development management: development priorities e.g. key skills focus, career relevance; goal alignment e.g. short-term goals, career goals, learning goals; progress evaluation e.g. milestone review, reflection outcomes

Performance improvement: feedback use e.g. tutor feedback, peer feedback; performance review e.g. progress reviews, self-evaluation, assessment feedback; confidence building e.g. presentation practice, skill reinforcement.

*Employability skills in computing:*

Communication and collaboration: communication skills e.g. written communication, verbal communication; teamwork e.g. role contribution, shared responsibility, collaboration tasks

Cognitive capability: problem-solving skills e.g. issue identification, solution development; analytical thinking e.g. comparison, evaluation, structured reasoning

Professional effectiveness: professionalism e.g. conduct standards, reliability; adaptability e.g. role flexibility, task variation, technology change

Workplace organisation: digital literacy e.g. digital tools, online platforms; time management e.g. task planning, prioritisation, scheduling.

*CPD planning elements:*

Goal setting: career goals e.g. target roles, sector focus; learning objectives e.g. skill improvement, role readiness, knowledge development

Development activities: development activities e.g. projects, practical tasks, workshops; training courses e.g. short courses, online learning

Learning strategies: self-directed learning e.g. independent study, skills practice; industry standards e.g. professional benchmarks, best practice guidance

Monitoring and review: timescales e.g. short-term planning, review dates; progress monitoring e.g. performance tracking, evidence collection.

*Career progression awareness:*

Career structure: job roles e.g. entry-level roles, technician roles; career pathways e.g. technical routes, specialist routes

Role alignment: role requirements e.g. skills expectations, experience levels; transferable skills e.g. communication, teamwork, organisation

Employability context: sector expectations e.g. conduct standards, professional behaviour; professional standards e.g. codes of conduct, workplace norms

Labour awareness: workplace culture e.g. team norms, working practices; labour market awareness e.g. skill demand, job availability.

*Professional presentation to an intended audience:*

Audience preparation: audience analysis e.g. knowledge levels, expectations; terminology accuracy e.g. correct language, technical terms

Presentation design: presentation structure e.g. logical flow, clear sections; visual design e.g. slides, diagrams, layout consistency

Delivery skills: communication clarity e.g. clear explanations, appropriate pace; professional tone e.g. formal language, confident delivery

Engagement management: delivery style e.g. structured explanation, interactive approach; question handling e.g. clarification, professional responses.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explore a range of effective communication and collaboration skills in a professional context</td><td rowspan="2">D1 Evaluate, using a selection of real-world examples, the suitability of communication and collaboration approaches for a range of professional computing scenarios.</td></tr>
<tr><td>P1 Explore, using a selection of real-world examples, communication methods and collaboration approaches used in professional computing contexts.<br><br>P2 Review, using a selection of real-world examples, the tools used to support effective communication and collaboration in professional computing contexts.</td><td>M1 Assess, using a selection of real-world examples, the benefits and limitations of communication and collaboration approaches used in professional computing contexts.</td></tr>
<tr><td colspan="2">LO2 Explain the principles of professional, legal, and ethical practice in computing</td><td rowspan="2">D2 Evaluate, using a selection of real-world examples, the implications of professional, legal, ethical and sustainability principles on responsible computing practice.</td></tr>
<tr><td>P3 Explain, using a selection of real-world examples, the professional, legal and ethical principles and associated risks relevant to computing practice and how the risks can be mitigated.<br><br>P4 Discuss, using a selection of real-world examples, the sustainability considerations relevant to computing practice.</td><td>M2 Analyse, using a selection of real-world examples, how professional, legal, ethical and sustainability considerations influence decision-making in computing activities within organisational contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Deliver a workshop on an identified technical computing topic that shows professionalism | | **LO3 and LO4**<br>**D3** Evaluate the effectiveness of CPD planning and workshop delivery in developing professional behaviour and employability skills within the computing industry. |
| **P5** Plan a workshop session on an identified technical computing topic that shows professionalism.<br>**P6** Deliver a workshop on an identified technical computing topic that shows professionalism and gather appropriate feedback. | **M3** Analyse the feedback obtained to identify strengths and weaknesses in the clarity and professionalism of the workshop on the identified technical computing topic. | |
| **LO4** Present a CPD plan on personal development and employability skills to an intended audience | | |
| **P7** Develop a CPD plan outlining personal development and employability skills.<br>**P8** Present a CPD plan outlining personal development and employability skills to an intended audience to obtain feedback. | **M4** Analyse, using the feedback obtained from the intended audience, the effectiveness of the CPD plan in supporting personal and professional growth. | |

## Recommended Resources

### Textbooks

Bourgeois, D.T. *et al.* (2023) *Information Systems for Business and Beyond*. Washington: Open Textbook Library.

Cottrell, S. (2023) *Skills for Success: Personal Development and Employability*. 5th edn. London: Bloomsbury Academic.

Reynolds, G.W. (2024) *Ethics in Information Technology*. 7th edn. Boston, MA: Cengage Learning.

Sommerville, I. and Tilley, S. (2024) *Software Engineering*. 11th edn. Harlow: Pearson.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.open.edu/openlearn/digital-computing/hybrid-working-digital-communication-and-collaboration/content-section-0?intro=1 | Open University OpenLearn<br>Online article introducing digital communication and collaboration skills for hybrid and workplace contexts.<br>(General reference) |
| creately.com/guides/collaborative-communication/ | Creately<br>Collaborative Communication: Definition, Principles, Templates, Strategies & Examples<br>(Article) |
| www.skillsyouneed.com/ips/communication-skills.html | SkillsYouNeed<br>Communication Skills<br>(Article) |
| www.purdueglobal.edu/blog/information-technology/ethics-information-technology/ | Purdue Global<br>Ethical Issues in Information Technology (IT)<br>(Article) |
| www.geeksforgeeks.org/blogs/ethical-issues-in-information-technology-it/ | GeeksforGeeks<br>Ethical Issues in Information Technology (IT)<br>(Article) |

| | |
|---|---|
| www.sheffield.ac.uk/study-skills/university/communication/presentations | University of Sheffield<br>How to plan, create and deliver presentations<br>(Tutorial) |
| www.mindtools.com/aeffdn2/planning-a-workshop/ | Mind Tools<br>Step-by-step guidance on planning and delivering effective workshops and presentations in professional settings.<br>(Tutorial) |
| www.skillsyouneed.com/rhubarb/presentation-skills-professionals.html | SkillsYouNeed<br>Effective Presentation Skills for Professionals<br>(Tutorial) |
| www.prospects.ac.uk/careers-advice/getting-a-job/cpd-opportunities-for-graduates/ | Prospects<br>CPD Opportunities for Graduates<br>(Article) |
| ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-protection-principles/a-guide-to-the-data-protection-principles/ | Information Commissioner's Office (ICO)<br>Guide to the data protection principles (General reference) |

**Journals and articles**

AlGhamdi, R. (2023) 'Development of Soft Skills among Computing Students in Online Task-Based Learning: Insights from Technical Communication Course', *International Journal of Technology in Education*, 6(2), 260–282.
Available at: doi.org/10.46328/ijte.394.

Allison, J. (2023) 'Classifying the Characteristics of Effective Continuing Professional Development (CPD) for Computer Science Teachers in the 16-18 Sector', *ACM Transactions on Computing Education*, 23(2).
Available at: doi.org/10.1145/3582275.

Liu, W. *et al.* (2024) 'Implementation of Technical Communication Pedagogy in Higher Education Institutions: A Systematic Literature Review', *International Journal of Learning, Teaching and Educational Research*, 23(9).
Available at: doi.org/10.26803/ijlter.23.9.16.

Ribeiro, D. and Varajão, J. (2025) 'Codes of ethics and conduct in information systems: Towards a unified framework', *Management Review Quarterly*.
Available at: doi.org/10.1007/s11301-025-00521-9.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set).*

# Unit 404: Digital Systems Analysis

**Unit code:** **F/652/3256**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

Digital systems underpin how modern organisations operate, communicate and adapt to change. As businesses increasingly depend on technology to deliver services and manage information, the ability to analyse digital systems effectively has become essential. This unit focuses on understanding business needs, system behaviour and stakeholder expectations in order to plan for essential organisational change and digital transformation.

The aim of this unit is to develop students' understanding of how digital systems are analysed within an organisational environment. It focuses on the role of digital and business analysts in identifying problems, defining requirements and supporting the design and evaluation of effective solutions. The unit introduces students to recognised principles, methods and frameworks that are used to support decision-making and structured change.

The unit covers key areas required to meet the learning outcomes. Students explore the core principles of digital and business analysis across the change life cycle, including governance and stakeholder involvement. They examine requirements analysis processes and recognised elicitation and documentation techniques used to define system needs accurately. The unit also develops skills in business process and data modelling, before considering how designed solutions are reviewed and evaluated against organisational objectives, constraints and performance measures.

On successful completion of this unit, students will be able to explain business analysis concepts; review and analyse requirements; design clear process models; and evaluate solution suitability. These skills support progression into roles such as Junior Business Analyst, Systems Analyst, Digital Change Analyst, Business Support Analyst or further study in digital systems development, analysis or transformation-focused pathways.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain core principles of business analysis across the organisational change life cycle

LO2 Review the process of requirements analysis using recognised techniques

LO3 Design a business process model for an identified organisational scenario

LO4 Review the suitability of the designed solution for an identified organisational scenario.

## Essential Content

LO1 **Explain core principles of business analysis across the organisational change life cycle**

*Business analysis principles:*

Value focus: value delivery e.g. customer value, business value; problem definition e.g. business problems, opportunity statements; solution alignment e.g. strategic alignment, business objectives

Context focus: organisational context e.g. organisational goals, operating environment; continuous improvement e.g. process improvement, performance improvement.

*Change life cycle:*

Life cycle stages: initiation e.g. business case, change drivers; analysis e.g. problem analysis, requirements analysis; design e.g. solution design, process design; implementation e.g. solution deployment, change delivery; evaluation e.g. outcome review, benefit review

Benefits focus: benefits identification e.g. expected benefits, measurable benefits; evaluation e.g. benefit realisation, outcome assessment; implementation e.g. benefit tracking, improvement actions.

*Business analyst roles and responsibilities:*

Analysis activities: requirements management e.g. requirement tracking, requirement control; analysis support e.g. impact analysis, option analysis; decision support e.g. evidence provision, option comparison

Engagement activities: stakeholder engagement e.g. stakeholder communication, stakeholder involvement; facilitation e.g. workshop facilitation, discussion facilitation; decision support e.g. recommendation support, evaluation input.

*Stakeholders and governance:*

Stakeholder management: stakeholder identification e.g. internal stakeholders, external stakeholders; stakeholder influence e.g. interest levels, influence levels; decision authority e.g. decision roles, approval authority

Control mechanisms: approval structures e.g. approval levels, approval processes; change control e.g. change requests, impact control; governance frameworks e.g. project governance, organisational governance.

*Professional frameworks:*

Knowledge structure: Business Analysis Body of Knowledge (BABOK) knowledge areas e.g. analysis planning, requirements analysis; perspectives e.g. organisational perspective, solution perspective; standards-based working e.g. professional standards, industry standards

Professional practice: ethical practice e.g. professional conduct, ethical standards; standards-based working e.g. consistent methods, documented standards, professional currency.

LO2 **Review the process of requirements analysis using recognised techniques**

*Requirements analysis process:*

Process stages: requirements elicitation e.g. information gathering, stakeholder input; requirements documentation e.g. specification records, requirement descriptions; validation e.g. requirement confirmation, stakeholder agreement; verification e.g. requirement checking, accuracy review; traceability e.g. requirement linking, change tracking.

*Elicitation techniques:*

Information gathering: interviews e.g. stakeholder interviews, structured interviews; workshops e.g. facilitated workshops, requirement workshops; observation e.g. workplace observation, task observation

Document-based techniques: surveys e.g. questionnaires, feedback collection; document analysis e.g. existing documents, system documentation.

*Requirements representation:*

Requirement formats: user stories e.g. story structure, acceptance focus; use cases e.g. actor interactions, scenario flows

Requirement types: functional requirements e.g. system behaviour, user actions; non-functional requirements e.g. performance needs, usability needs; acceptance criteria e.g. success conditions, completion measures.

*Prioritisation and management techniques:*

Prioritisation methods: MoSCoW prioritisation e.g. must have, should have, could have, won't have; dependency identification e.g. requirement dependencies, sequence dependencies

Management controls: change impact awareness e.g. impact identification, consequence awareness; requirements versioning e.g. version control, update tracking.

*Traceability and quality:*

Traceability focus: requirements traceability awareness e.g. requirement mapping, forward traceability; alignment to business objectives e.g. strategic alignment, objective linkage; change impact visibility e.g. impact tracking, change awareness.

*Requirements risk and quality assurance:*

Risk factors: ambiguity e.g. unclear wording, vague statements; completeness e.g. missing requirements, coverage gaps; consistency e.g. conflicting requirements, alignment issues

Risk consequences: project delays e.g. schedule slippage, milestone overruns; cost increases e.g. budget overrun, rework effort; quality issues e.g. defect leakage, non-compliance; stakeholder impact e.g. customer dissatisfaction, requirement rejection

Quality controls: validation risk e.g. incorrect approval, misunderstanding risk; change impact e.g. scope change, requirement instability.

LO3 **Design a business process model for an identified organisational scenario**

*Business process modelling concepts:*

Core components: process scope e.g. process boundaries, modelling limits; inputs and outputs e.g. data inputs, process outputs; actors e.g. users, system roles; events e.g. trigger events, start events; decision points e.g. decision rules, branching paths.

*Modelling techniques:*

Modelling approaches: Business Process Model and Notation (BPMN) 2.0 e.g. standard notation, process flows; suppliers, inputs, process, outputs, customers (SIPOC) e.g. suppliers, customers; value chain e.g. primary activities, support activities; value chain models e.g. activity sequences, value creation.

*Process clarity and consistency:*

Quality principles: standard notation e.g. recognised symbols, modelling standards; logical flow e.g. clear sequence, logical order; completeness e.g. full coverage, missing steps; readability e.g. visual clarity, simple layout; stakeholder interpretation e.g. shared understanding, user interpretation.

*Process performance and decision support:*

Performance focus: bottlenecks e.g. delays, constraints; handoffs e.g. task transfer, responsibility transfer; efficiency awareness e.g. effort reduction, streamlined flow

Decision enablement: information visibility e.g. accessible information, clear data; decision points e.g. approval points, routing options.

*Data and structure models:*

Data representation: entity–relationship diagrams (ERD) e.g. entities, relationships; data normalisation e.g. data organisation, redundancy reduction

Process structure: process steps e.g. task steps, activity steps; process sequence e.g. step order, activity flow; process states e.g. start states, end states; information flow diagrams e.g. data movement, information exchange.

LO4 **Review the suitability of the designed solution for an identified organisational scenario**

*Solution suitability concepts:*

Suitability factors: fitness for purpose e.g. intended use, business need; alignment to requirements e.g. requirement matching, scope coverage; feasibility e.g. technical feasibility, operational feasibility; constraints e.g. time limits, budget limits.

*Verification and validation:*

Quality assurance principles: verification principles e.g. requirement checking, specification review; validation principles e.g. stakeholder confirmation, acceptance review; verification versus validation distinction e.g. build correctness, solution appropriateness.

*Testing and acceptance theory:*

Testing concepts: user acceptance testing (UAT) concepts e.g. alpha testing, beta testing, business acceptance, operational acceptance, contract acceptance, regulatory acceptance, usability; test purpose e.g. fault identification, requirement confirmation; test plans e.g. test structure, test scope; test cases e.g. test scenarios, test conditions; expected outcomes e.g. pass criteria, expected results.

*Benefits and performance evaluation:*

Evaluation measures: benefits mapping e.g. benefit identification, benefit tracking; Key Performance Indicators (KPIs) e.g. performance measures, success metrics; cost–benefit considerations e.g. cost comparison, value assessment.

*Stakeholder and organisational alignment:*

Alignment considerations: stakeholder expectations e.g. user needs, management expectations; business objectives e.g. strategic goals, operational goals; operational impact e.g. process impact, resource impact; performance measures e.g. efficiency measures, effectiveness measures.

*Organisational and solution constraints:*

Constraint factors: cost e.g. budget limits, financial resources; time e.g. delivery timescales, deadlines; technical feasibility e.g. system capability, technical limits; resource limitations e.g. staff availability, skill availability.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain core principles of business analysis across the organisational change life cycle</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a selection of real-world organisational examples, the effectiveness of business analysis principles and requirements analysis techniques in supporting successful organisational change across the change life cycle.</td></tr>
<tr><td>P1 Explain, using a selection of real-world organisational examples, the role of a Business Analyst and how they apply the core principles of business analysis across the change life cycle.<br><br>P2 Discuss, using a selection of real-world organisational examples, how stakeholders, governance and professional frameworks can impact on the role of the Business Analyst.</td><td>M1 Analyse, using a selection of real-world organisational examples, how core business analysis principles support effective change across the change life cycle.</td></tr>
<tr><td colspan="2">LO2 Review the process of requirements analysis using recognised techniques</td></tr>
<tr><td>P3 Review, using a selection of real-world organisational examples, the process of using recognised elicitation and documentation techniques to analyse and represent a range of different requirements.<br><br>P4 Discuss, using a selection of real-world organisational examples, how requirements can be managed to maintain quality and minimise risk.</td><td>M2 Assess, using a selection of real-world organisational examples, the effectiveness of different requirements analysis techniques in managing risk and maintaining quality.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a business process model for an identified organisational scenario | | **LO3 and LO4**<br>**D2** Justify, using the review outcomes and stakeholder feedback, the decisions made in the design of the business process model in being able to comprehensively address organisational needs. |
| **P5** Design, using a selection of business modelling techniques, the business process requirements of an identified organisational scenario.<br>**P6** Design, using a selection of data modelling techniques, a data model that addresses the requirements of an identified organisational scenario. | **M3** Justify the selection of the process and data modelling techniques used to design the model that meets the requirements of an identified organisational scenario. | |
| **LO4** Review the suitability of the designed solution for an identified organisational scenario | | |
| **P7** Review, using defined evaluation measures, the suitability of the designed business process model for the identified scenario.<br>**P8** Deliver a presentation on the suitability of the designed business process model for an identified organisational scenario to a range of identified organisational stakeholders. | **M4** Evaluate, using the review outcomes and stakeholder feedback, the suitability of the designed solution in meeting stakeholder needs and intended outcomes. | |

 © Pearson Education Limited 2026

## Recommended Resources

### Textbooks

Cadle, J., Paul, D. and Turner, P. (2023) *Business Analysis Techniques: 99 Essential Tools for Success*. 3rd edn. Swindon: BCS Learning & Development Ltd.

Dennis, A., Wixom, B.H. and Roth, R.M. (2021) *Systems Analysis and Design*. 8th edn. Hoboken, NJ: Wiley.

Dumas, M., La Rosa, M., Mendling, J. and Reijers, H.A. (2023) *Fundamentals of Business Process Management*. 3rd edn. Berlin: Springer.

Paul, D. and Cadle, J. (2024) *Business Analysis*. 5th edn. Swindon: BCS Learning & Development Ltd.

Robertson, J. and Robertson, S. (2021) *Mastering the Requirements Process*. 4th edn. Upper Saddle River: Addison-Wesley.

Sharp, A. and McDermott, P. (2023) *Workflow Modeling: Tools for Process Improvement and Application Development*. 2nd edn. Boston: Artech House.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.iiba.org/career-resources/a-business-analysis-professionals-foundation-for-success/the-foundation-for-effective-business-analysis | International Institute of Business Analysis (IIBA)<br>The Business Analysis Standard outlining core principles, concepts and professional practice<br>(General reference) |
| www.batimes.com/articles/what-is-business-analysis/ | BA Times<br>Overview of business analysis roles, principles and organisational value<br>(Article) |
| www.softwaretestinghelp.com/requirements-elicitation-techniques/ | Software Testing Help<br>Requirements elicitation techniques and stakeholder engagement methods<br>(Tutorials) |
| community.atlassian.com/forums/App-Central-articles/Understanding-the-MoSCoW-prioritization-How-to-implement-it-into/ba-p/2463999 | Atlassian Community<br>MoSCoW prioritisation technique for managing and reviewing requirements<br>(Article) |

| | |
|---|---|
| www.omg.org/bpmn/ | Object Management Group (OMG)<br>Business Process Model and Notation (BPMN) standard and modelling reference<br>(General reference) |
| www.visual-paradigm.com/tutorials/business-process-modeling-tutorial/ | Visual Paradigm<br>Business process modelling concepts and BPMN diagramming guidance<br>(Tutorials) |
| www.geeksforgeeks.org/software-engineering/software-engineering-verification-and-validation/ | GeeksforGeeks<br>Verification and validation concepts used to review system suitability<br>(Article) |
| www.productplan.com/glossary/moscow-prioritization/ | ProductPlan<br>Requirements prioritisation and decision-making techniques for solution evaluation using MoSCoW<br>(General reference) |
| creately.com/guides/business-process-modeling-tutorial/ | Creately<br>Business process modelling techniques for analysing and improving systems<br>(Tutorials) |

**Journals and articles**

Ven, M. van de *et al.* (2023) 'Key performance indicators for business models: a systematic review and catalog', *Information Systems and e-Business Management*, 21, pp. 753–794. Available at: doi.org/10.1007/s10257-023-00650-2.

Mohamed, N., Mazen, S. and Helmy, W. (2025) 'A comprehensive review of software requirements dependencies analysis techniques', *Iraqi Journal for Computer Science and Mathematics*, 6(3). Available at: doi.org/10.52866/2788-7421.1279.

Lopes, T. and Guerreiro, S. (2023) 'Assessing business process models: a literature review on BPMN testing and verification', *Business Process Management Journal*, 29(8), pp. 133–162. Available at: doi.org/10.1108/BPMJ-11-2022-0557.

Pitchikala, K.M. (2024) 'Selecting Effective Metrics for Evaluating and Measuring Software Application Performance', *International Journal of Science and Research*, 13(9), pp. 223–227. Available at: doi.org/10.21275/SR24903025128.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 403: Professional Practice.*

# Unit 405: Core Software Design and Development

**Unit code:** H/652/3257

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Software design and development play a central role in the way modern organisations operate, underpinning systems used in business, healthcare, education, entertainment, and public services. From simple applications to large-scale information systems, well-designed software supports reliability, efficiency, and adaptability. As digital technologies continue to evolve, there is an increasing demand for professionals who not only understand how to write code, but also how to design software that is structured, maintainable, and suited to its intended purpose.

The aim of this unit is to introduce students to the core principles and practices that support effective software design and development. The unit focuses on how design decisions influence the quality, reliability, and usability of software systems, and highlights the importance of applying recognised approaches throughout the software development life cycle. This unit will allow students to develop an understanding of professional design practices that are widely used within the software industry.

The unit covers a range of foundational topics essential to software development. Students explore fundamental software design principles and examine different software development life cycle models used in industry. Standard modelling techniques are introduced to help students represent and communicate software structure and behaviour clearly. The unit also explores how programming constructs can be applied effectively in software design, before culminating in the development of a software application for an identified purpose.

On successful completion of this unit, students will be able to explain key software design principles, interpret and use modelling techniques, and develop software solutions that meet defined requirements. These skills support progression into roles such as Junior Software Developer, Application Developer, Software Tester, Systems Support Analyst, and Trainee Software Engineer, as well as further study in software engineering and related computing disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain fundamental principles of software design and development

LO2 Demonstrate the use of standard modelling techniques to represent software systems

LO3 Explore how programming constructs can be used effectively in software design

LO4 Develop a software application for an identified purpose using a range of software design principles.

## Essential Content

LO1 **Explain fundamental principles of software design and development**

*Software design principles:*

Structural principles: abstraction, e.g. interface definition, data hiding; modularity, e.g. component separation, module boundaries

Component quality: cohesion, e.g. single responsibility, logical grouping; coupling, e.g. dependency reduction, interface reliance

Design organisation: separation of concerns, e.g. presentation logic, business logic

Long-term design qualities: reusability, e.g. shared libraries, generic components; maintainability, e.g. readable code, structured updates; scalability, e.g. growth handling, increased workload.

*Software development life cycle (SDLC) models:*

Sequential approaches: Waterfall model, e.g. linear phases, fixed requirements

Adaptive approaches: Agile methodologies, e.g. sprint cycles, continuous feedback

Incremental development: iterative models, e.g. repeated refinement, version improvement; incremental models, e.g. feature addition, staged delivery

Combined approaches: hybrid approaches, e.g. mixed phases, adaptive structure.

*Development methodologies comparison:*

Process characteristics: sequential development characteristics, e.g. defined stages, upfront planning; adaptive development characteristics, e.g. evolving requirements, iterative delivery

Change responsiveness: flexibility, e.g. requirement adaptation, process adjustment; change management, e.g. controlled updates, version handling

Process control: documentation focus, e.g. design records, specification tracking.

*Methodology selection considerations:*

Project definition: project scope, e.g. feature boundaries, deliverable limits; complexity, e.g. system size, interaction levels

People involvement: stakeholder involvement, e.g. user feedback, client input

Risk and constraints: risk management, e.g. technical uncertainty, delivery risk; resource constraints, e.g. staff availability, tool access; time constraints, e.g. deadlines, development schedules.

*Requirements engineering:*

Requirement capture: definition e.g. stakeholder needs, system requirements; elicitation e.g. interviews, surveys

Requirements management: analysis e.g. classification, prioritisation; specification e.g. documentation, models; validation e.g. consistency checks, completeness checks; management e.g. change management, traceability.

LO2 **Demonstrate the use of standard modelling techniques to represent software systems**

*Unified Modelling Language (UML) overview:*

Modelling purpose: purpose of modelling, e.g. design clarification, requirement representation; system visualisation, e.g. structural overview, interaction mapping

Abstraction and communication: abstraction of software components, e.g. simplified representation, component focus; communication of design intent, e.g. shared understanding, design explanation.

*Structural modelling techniques:*

Static structure representation: class diagrams, e.g. class structure, relationship overview; object relationships, e.g. associations, dependencies

Class definition elements: attributes, e.g. data properties, state representation; methods, e.g. behaviour definition, operation listing

Relationship types: associations, e.g. linkage definition, interaction reference; inheritance, e.g. generalisation, specialisation; aggregation, e.g. whole-part structure, containment.

*Behavioural modelling techniques:*

User interaction representation: use case diagrams, e.g. system usage, functional scope; actors, e.g. user roles, external systems

System scope and interaction: system boundaries, e.g. included functions, excluded elements; interaction flows, e.g. process steps, user actions

Sequence representation: sequence diagrams, e.g. message order, object interaction; message sequencing, e.g. call order, response flow.

*Model interpretation and communication:*

Design clarity and documentation: design consistency, e.g. notation accuracy, structural alignment; system documentation, e.g. design records, model archives

Stakeholder communication: stakeholder understanding, e.g. shared interpretation, requirement clarity; representation of functional requirements, e.g. system functions, expected behaviour.

*Model suitability and limitations:*

Purpose and abstraction constraints: purpose alignment, e.g. modelling intent, system focus; abstraction limitations, e.g. detail omission, simplification boundaries

Complexity and perspective: complexity representation, e.g. system scale, interaction density; stakeholder perspective, e.g. technical audience, non-technical audience.

LO3 **Explore how programming constructs can be used effectively in software design**

*Programming data structures:*

Linear collections: arrays, e.g. indexed storage, fixed size; lists, e.g. dynamic sizing, sequential access

Access models: stacks, e.g. last-in-first-out, call stack; queues, e.g. first-in-first-out, task scheduling

Storage and resources: data organisation, e.g. structured storage, logical grouping; memory considerations, e.g. allocation efficiency, usage limits; file input and output, e.g. persistent storage, data streams.

*Control structures:*

Decision control: selection structures, e.g. conditional branching, decision paths; logic paths, e.g. execution routes, outcome handling

Repetition control: iteration structures, e.g. loops, repeated execution; flow control, e.g. execution order, control transfer.

*Functional decomposition:*

Procedural units: functions, e.g. reusable operations, named routines; methods, e.g. object behaviour, encapsulated actions

Data exchange: parameter passing, e.g. input arguments, data transfer; by reference, e.g. shared memory access, variable linkage; by value, e.g. copied data, isolated variables

Output and structure: return values, e.g. result output, status codes; modular decomposition, e.g. logical separation, structured design.

*Error handling mechanisms:*

Fault detection: exception handling, e.g. runtime errors, handled exceptions; validation logic, e.g. input checking, rule enforcement

System resilience: fault tolerance, e.g. graceful failure, recovery strategies; robustness considerations, e.g. error resistance, stable behaviour.

*Algorithmic concepts:*

Algorithm operations: searching techniques, e.g. linear search, binary search; sorting techniques, e.g. bubble sort, merge sort

Algorithm quality: efficiency considerations, e.g. time complexity, resource usage; readability, e.g. clear logic, maintainable structure.

LO4 **Develop a software application for an identified purpose using a range of software design principles**

*Software design principles application:*

Design structure: modularity, e.g. component separation, independent modules; separation of concerns, e.g. layered design, responsibility division

Component quality: cohesion, e.g. focused functionality, logical grouping; coupling, e.g. dependency control, interface use.

*Software structure and organisation:*

Code clarity: naming conventions, e.g. meaningful identifiers, consistent naming; code readability, e.g. clear logic, structured layout

Documentation and organisation: documentation standards, e.g. inline comments, design notes; logical file structure, e.g. folder hierarchy, file grouping.

*Testing and quality considerations:*

Testing concepts: unit testing concepts, e.g. test cases, isolated testing; verification, e.g. specification checking, requirement matching

Quality checking: validation, e.g. user needs confirmation, functional accuracy; defect identification, e.g. fault detection, issue logging.

*Maintainability and reliability factors:*

Long-term maintenance: code clarity, e.g. understandable logic, readable structure; extensibility, e.g. feature expansion, modular growth

System stability: error resilience, e.g. fault handling, recovery behaviour; consistency, e.g. uniform style, predictable behaviour.

*Application purpose alignment:*

Requirement focus: functional requirements alignment, e.g. feature fulfilment, expected behaviour; user needs alignment, e.g. usability expectations, user goals

Context awareness: operational context, e.g. deployment environment, usage conditions.

*Software quality attributes:*

Core quality attributes: functionality, e.g. correct operation, feature completeness; reliability, e.g. stable performance, fault tolerance

User and maintenance qualities: usability, e.g. ease of use, intuitive interaction; maintainability, e.g. ease of update, supportability

Performance and deployment: performance, e.g. response time, efficiency; portability, e.g. platform compatibility, environment transfer.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain fundamental principles of software design and development</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a selection of real-world examples, the effectiveness of software design, development and modelling techniques in helping develop robust and scalable software solutions.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the fundamental principles of software design.<br><br>P2 Discuss, using a selection of real-world examples, the principles of the software development life cycle and methodologies, considering the factors that can affect the choice of a development methodology.</td><td>M1 Analyse, using a selection of real-world examples, how software design principles and software development life cycle models influence the structure and quality of software solutions.</td></tr>
<tr><td colspan="2">LO2 Demonstrate the use of standard modelling techniques to represent software systems</td></tr>
<tr><td>P3 Demonstrate the use of standard modelling techniques to develop a selection of models that represent the structure and behaviour of software systems.<br><br>P4 Explain how the developed models can be interpreted and communicated to a range of stakeholders, considering model suitability and limitations.</td><td>M2 Assess the effectiveness of the different models in communicating an understanding of the software system designs to a range of stakeholders.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Explore how programming constructs can be used effectively in software design | | |
| **P5** Explore, using a range of programmatic examples, the use of programming structures, control structures and error handling mechanisms in software design.<br><br>**P6** Demonstrate, using a range of programmatic examples, the application of functional decomposition and algorithmic concepts in software design. | **M3** Analyse, using a range of programmatic examples, how programming constructs contribute to structured, reliable, and maintainable software design. | **D2** Evaluate, using a range of programmatic examples, the suitability of programming constructs in effectively addressing software design requirements and design constraints. |
| **LO4** Develop a software application for an identified purpose using a range of software design principles | | |
| **P7** Develop a software design and test plan that uses a range of software design and modelling techniques to develop an application that meets an identified purpose.<br><br>**P8** Develop a software application that applies software design principles and software quality to meet an identified purpose. | **M4** Analyse, using the results of the implemented test plan, the quality, maintainability and reliability of the software application developed for an identified purpose. | **D3** Justify, using the results of the application testing, the decisions made in the design and development of the application in meeting the identified requirements and supporting quality, maintainability and reliability. |

## Recommended Resources

### Textbooks


Arlow, J. and Neustadt, I. (2021) *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*. 2nd edn. Boston: Addison-Wesley Professional.

Hu, C. (2023) *An Introduction to Software Design: Concepts, Principles, Methodologies, and Techniques*. Cham: Springer.

McConnell, S. (2024) *Code Complete: A Practical Handbook of Software Construction*. 3rd edn. Redmond: Microsoft Press.

Ousterhout, J. (2024) *A Philosophy of Software Design*. 2nd edn. Palo Alto: Yaknyam Press.

Pressman, R.S. and Maxim, B.R. (2023) *Software Engineering: A Practitioner's Approach*. 9th edn. New York: McGraw-Hill.

Sommerville, I. and Tilley, S. (2024) *Software Engineering*. 11th edn. Harlow: Pearson.


### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.hostinger.com/tutorials/software-design | Hostinger<br>Software design principles and practices<br>(Article) |
| softjourn.com/insights/core-principles-of-software-development | Softjourn<br>Core principles of software development<br>(Article) |
| realpython.com/python-control-flow/ | Real Python flow structures in Python<br>(Tutorial) |
| www.tutorialspoint.com/uml/index.htm | TutorialsPoint<br>UML Tutorial<br>(Tutorial) |
| www.geeksforgeeks.org/system-design/unified-modeling-language-uml-introduction/ | GeeksforGeeks<br>UML diagrams introduction<br>(General reference) |

| | |
|---|---|
| www.datacamp.com/tutorial/functional-decomposition | DataCamp<br>Functional Decomposition: A Practical Guide to System Design<br>(Tutorial) |
| www.tutorialspoint.com/article/unit-testing-tutorial-for-beginners-concepts-types-tools | TutorialsPoint<br>Unit Testing: Concepts, Types, and Tools<br>(Tutorial) |

**Journals and articles**

Gadhi, A. *et al.* (2023) 'Applying UML and Machine Learning to Enhance System Analysis and Design', *International Journal of Communications, Network and System Sciences*, 16, pp. 67–76. Available at: doi.org/10.4236/ijcns.2023.165005.

Gobov, D. and Zuieva, O. (2025) 'Software Quality Attributes in Requirements Engineering', *International Journal of Information Technology and Computer Science*, 17(4), pp. 38–48. Available at: doi.org/10.5815/ijitcs.2025.04.04.

Runge, T. *et al.* (2023) 'Flexible correct-by-construction programming', *Logical Methods in Computer Science*, 19(2). Available at: doi.org/10.48550/arXiv.2211.15261.

Yas, Q.M., Alazzawi, A. and Rahmatullah, B. (2023) 'A Comprehensive Review of Software Development Life Cycle methodologies: Pros, Cons, and Future Directions', *Iraqi Journal for Computer Science and Mathematics*, 4(4). Available at: doi.org/10.52866/ijcsm.2023.04.04.014.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 403: Professional Practice*

*Unit 404: Digital Systems Analysis*

*Unit 406: Core Software Testing*

*Unit 407: Applied Software Testing*

*Unit 408: Applied Software Development*

*Unit 409: Core Cybersecurity*

*Unit 410: Database Engineering*

*Unit 411: Core Data Analytics.*

# Unit 406: Core Software Testing

**Unit code:** J/652/3258

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Software underpins almost every modern organisation, from business and finance to healthcare, education and public services. As systems become more complex and increasingly relied upon, the need for reliable, secure and well-tested software has become critical. Software testing plays a vital role in identifying defects, reducing risk and ensuring that systems behave as intended for users. This unit introduces students to the core principles of software testing and quality assurance, placing testing firmly within the wider context of the software development life cycle and the expectations of industry.

The aim of this unit is to develop students' understanding of how structured software testing supports quality, reliability and continuous improvement in digital systems. Students will explore why testing is carried out, how it contributes to meeting user and business requirements, and how recognised standards and best practices shape professional testing activity. The unit is designed to build confidence in applying testing concepts in realistic scenarios, preparing students to contribute effectively within development and quality-focused teams.

The unit covers the essential knowledge and practical skills required for core software testing. Key areas include the principles, objectives and terminology used in software testing and quality assurance, alongside an overview of industry standards, frameworks and professional guidelines. Students will also develop practical skills in designing test plans and test cases for given scenarios, considering coverage, traceability and quality. The unit concludes with the review and evaluation of test results, encouraging students to use evidence to identify strengths, weaknesses and opportunities for improvement in testing activity.

On successful completion of this unit, students will be able to explain core testing concepts, apply recognised standards and best practices, design structured test artefacts, and review test outcomes to support informed decisions. These skills are directly relevant to a range of entry-level and junior roles within the digital sector, including Junior Software Tester, Quality Assurance Assistant, Software Support Analyst, Test Analyst, and Trainee Software Developer, as well as providing a strong foundation for further study in software testing, quality assurance and software engineering.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the principles, objectives, and terminology of software testing and quality assurance

LO2 Review the industry standards and best practices for software testing

LO3 Design a set of test plans and test cases for a given software testing scenario

LO4 Review the results of the test plans and test cases for the given software testing scenario.

## Essential Content

LO1 **Explain the principles, objectives, and terminology of software testing and quality assurance**

*Principles and objectives of software testing:*

Types of testing: unit testing e.g. component testing, function testing; integration testing e.g. module interaction, interface testing; system testing e.g. end-to-end testing, system validation; acceptance testing e.g. user acceptance, business validation; functional testing e.g. feature verification, input–output validation; non-functional testing e.g. performance testing, security testing

Testing approaches and prioritisation: risk-based testing e.g. high-risk focus, critical functionality; prioritisation e.g. defect likelihood, business impact; representative test cases e.g. user scenarios, boundary conditions

Testing purpose and quality goals: purpose of testing e.g. defect identification, requirement validation; confidence in software quality e.g. reliability assurance, predictable behaviour; fitness for purpose e.g. user needs alignment, functional suitability

Defects and risk considerations: fault detection: logic errors e.g. incorrect conditions, calculation errors; security vulnerabilities e.g. authentication flaws, data exposure; performance bottlenecks e.g. slow response times, resource constraints; defect prevention e.g. early reviews, clear specifications, design validation; defect detection e.g. test execution, fault identification; risk reduction e.g. impact mitigation, failure likelihood reduction

Timing and alignment: early testing benefits e.g. early fault discovery, reduced rework, improved quality; alignment with business requirements e.g. requirement traceability, acceptance criteria, business goals.

*Quality assurance and quality control concepts:*

Quality approaches: quality assurance processes e.g. process definition, standards application; quality control activities e.g. inspections, testing execution

Focus and responsibility: process-focused approaches e.g. consistency improvement, prevention focus, life cycle support; product-focused approaches e.g. defect identification, output quality, functional correctness

Prevention and improvement: prevention e.g. proactive quality planning, process control; continuous quality improvement e.g. feedback application, incremental improvement, lessons learned.

*Testing terminology and core concepts:*

Defect-related terms: defects e.g. incorrect behaviour, missing functionality, performance issues; failures e.g. system crashes, incorrect outputs

Test artefacts and data: test cases e.g. functional tests, negative tests, regression tests, structural tests; test data e.g. valid data, boundary data

Control and traceability: entry and exit criteria e.g. readiness conditions, completion thresholds, quality gates; traceability e.g. requirement linkage, test coverage mapping.

*Testing within the software development life cycle:*

Life cycle structure: software development life cycle stages e.g. requirements stage, development stage, testing stage

Validation concepts: verification concepts e.g. specification checking, design review; validation concepts e.g. user needs confirmation, acceptance checking

Development models: Agile testing principles e.g. iterative testing, collaboration, frequent feedback.

LO2 **Review the industry standards and best practices for software testing**

*Industry standards for software testing:*

Software testing standards: ISO/IEC/IEEE 29119 series e.g. concepts and definitions (29119-1), test processes (29119-2), test documentation (29119-3); standardised terminology e.g. common testing language, shared definitions

Structured documentation and compliance: structured test documentation standards e.g. ISO/IEC/IEEE 29119-3 test plans and reports; compliance requirements e.g. organisational compliance, audit expectations

Agile and DevOps testing standards: Agile testing practices e.g. iterative testing, incremental delivery; DevOps testing practices e.g. continuous integration, continuous testing; collaborative testing e.g. cross-functional teams, shared responsibility; automation integration e.g. continuous integration/continuous deployment (CI/CD) pipelines, test automation

Open Worldwide Application Security Project (OWASP): security testing framework e.g. vulnerability testing, penetration testing; security standards e.g. OWASP Top 10 vulnerabilities, risk awareness; secure development practices e.g. secure coding, threat prevention; testing integration e.g. DevSecOps pipelines, continuous security validation.

*Testing frameworks and professional guidelines:*

Professional testing frameworks: International Software Testing Qualifications Board (ISTQB) principles e.g. testing fundamentals, life cycle understanding, common practices; testing life cycle models e.g. planning stages, execution stages

Test maturity concepts: test maturity concepts e.g. process capability, improvement levels, consistency development

Testing environments: development environment e.g. unit testing, feature testing; integration environment e.g. module interaction, interface validation; system testing environment e.g. end-to-end testing, requirement validation; staging environment e.g. pre-deployment testing, user acceptance validation; production-like environment e.g. real-world simulation, performance validation

Application of testing: regression testing e.g. functionality verification, change impact; smoke testing e.g. build stability, core functionality; sanity testing e.g. bug fix verification, targeted testing; functional testing e.g. feature validation, requirement checking; non-functional testing e.g. performance evaluation, security validation; acceptance testing e.g. user validation, business approval; manual testing e.g. exploratory testing, user interaction; automated testing e.g. scripted execution, regression automation.

*Best practices in software testing:*

Core testing practices: test documentation practices e.g. clear test cases, maintainable records; requirement traceability e.g. requirement mapping, coverage checking

Risk and defect handling: risk-based testing approaches e.g. prioritised testing, critical functionality focus; defect management processes e.g. defect tracking, status control.

*Quality governance and continuous improvement:*

Governance activities: quality audits e.g. ISO 9001 quality management audits, process compliance checks; test process reviews e.g. effectiveness evaluation, improvement identification

Organisational improvement: lessons learned e.g. post-test reviews, improvement actions; organisational learning e.g. shared practice, process refinement.

LO3 **Design a set of test plans and test cases for a given software testing scenario**

*Test planning fundamentals:*

Test objectives: test objectives e.g. checking requirements, finding defects; test scope definition e.g. included features, excluded features; assumptions and constraints e.g. system limits, time limits

Resource and scheduling considerations: resource considerations e.g. tester availability, tool access; scheduling concepts e.g. test phases, planned timelines.

*Test design techniques and documentation:*

Test design artefacts: test plans e.g. test approach, responsibilities; test cases e.g. functional tests, negative tests, regression tests; test data requirements e.g. valid data, invalid data

Design quality and priority: coverage considerations e.g. requirement coverage, functional coverage; prioritisation approaches e.g. important features first, high-risk areas.

*Traceability and coverage management:*

Traceability management: requirements traceability matrices e.g. requirement-to-test links, coverage tracking; linkage between requirements and test cases e.g. change awareness, test updates

Coverage evaluation: coverage metrics e.g. tests run, requirements covered.

*Indicators of test design quality:*

Design clarity: clarity of test objectives e.g. clear purpose, understandable scope; alignment with requirements e.g. requirement coverage, correct mapping

Coverage quality: completeness of coverage e.g. main features tested, key scenarios included; defect detection capability e.g. fault discovery, issue identification

Test sustainability: maintainability of test cases e.g. clear structure, easy updates; reproducibility e.g. repeatable steps, consistent results.

*Test execution preparation:*

Execution readiness: environment validation e.g. system availability, correct configuration; data preparation e.g. test data setup, data completeness

Defect handling and reporting: defect logging structures e.g. defect records, severity levels; reporting dashboards e.g. progress summaries, defect counts.

LO4 **Review the results of the test plans and test cases for the given software testing scenario**

*Test effectiveness measures:*

Test effectiveness indicators: test coverage indicators e.g. features tested, requirements covered; pass and fail rates e.g. tests passed, tests failed; test completion status e.g. tests executed, tests remaining; regression considerations e.g. retesting changes, repeated checks

Defect trends and indicators: defect trends e.g. increasing defects, repeated issues; defect density indicators e.g. defects per feature, defects per test area; test completion status e.g. tests completed, tests outstanding.

*Analysis of test results:*

Test result analysis: defect analysis e.g. defect types, defect causes; test outcome interpretation e.g. expected outcomes, actual outcomes; root cause identification e.g. common causes, repeated problems

Coverage evaluation: identification of gaps and redundancies e.g. missing tests, duplicate tests; effectiveness of test design e.g. relevant test cases, appropriate coverage; coverage indicators e.g. areas covered, areas not covered.

*Continuous improvement in testing:*

Improvement identification: recommendations for improvement e.g. additional tests, test updates; refinement of test plans e.g. scope changes, priority updates; process feedback loops e.g. tester feedback, issue reviews

Regression considerations: regression considerations e.g. retesting changes, repeated checks; process feedback e.g. lessons noted, improvement actions.

*Quality reporting and decision support:*

Reporting outputs: summary reporting e.g. test summaries, defect summaries; stakeholder communication e.g. progress updates, issue reporting; test completion status e.g. work completed, work remaining

Decision support: evidence-based recommendations e.g. test results use, improvement justification; alignment with quality objectives e.g. quality goals, reliability targets; improvement justification e.g. reasons for change, supporting evidence.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the principles, objectives, and terminology of software testing and quality assurance</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a selection of real-world examples, the extent to which industry standards, testing frameworks and professional best practices support the application of software testing principles and quality assurance objectives within the software development life cycle.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the terminology, principles and objectives of software testing within the software development life cycle.<br><br>P2 Discuss, using a selection of real-world examples, the concepts of quality assurance and quality control.</td><td>M1 Analyse, using a selection of real-world examples, how software testing impacts quality assurance and control across different stages of the software development life cycle.</td></tr>
<tr><td colspan="2">LO2 Review the industry standards and best practices for software testing</td></tr>
<tr><td>P3 Review, using a selection of real-world examples, the application of industry standards, testing frameworks and professional guidelines used within software testing.<br><br>P4 Explore, using a selection of real-world examples, how best practices in software testing support quality management and continuous improvement.</td><td>M2 Assess, using a selection of real-world examples, the impact of software testing standards and best practices on the effectiveness and reliability of testing processes.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a set of test plans and test cases for a given software testing scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and implementation of the test plans and test cases in supporting the development of reliable software and maintaining continuous improvement. |
| **P5** Design a set of test plans and test cases for a given software testing scenario.<br>**P6** Review the designed test plans and test cases for a given software testing scenario in terms of traceability, coverage and quality. | **M3** Evaluate the effectiveness of the designed test plans and test cases in supporting coverage and traceability. | |
| **LO4** Review the results of the test plans and test cases for the given software testing scenario | | |
| **P7** Implement the designed test plans and test cases for a given software testing scenario.<br>**P8** Review the results of the test plans and test cases for the given software testing scenario.<br>**P9** Present the results of the test plans and test cases for the given software testing scenario to an identified audience to communicate quality outcomes and to support decisions. | **M4** Analyse the effectiveness of the test results to identify strengths and areas for improvement in the test plans and test cases. | |

## Recommended Resources

### Textbooks

Ammann, P. and Offutt, J. (2024) *Introduction to Software Testing*. 3rd edn. Cambridge: Cambridge University Press.

Black, R. (2023) *Foundations of Software Testing: ISTQB Certification*. 4th edn. London: BCS Learning & Development.

Lewis, W.E. (2024) *Software Testing and Continuous Quality Improvement*. Boca Raton: CRC Press.

Myers, G.J., Sandler, C. and Badgett, T. (2024) *The Art of Software Testing*. 4th edn. Hoboken, NJ: Wiley.

Reid, S. (2023) *ISO/IEC/IEEE 29119 Software Testing Standards: A Practitioner's Guide*. 2nd edn. London: BCS Learning & Development.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/software-engineering/software-engineering-seven-principles-of-software-testing/ | GeeksforGeeks<br>Seven principles of software testing<br>(Article) |
| quality.arc42.org/standards/iso-iec-ieee-29119 | arc42 Quality Model<br>Practical overview of the ISO/IEC/ IEEE 29119 software testing standards<br>(Resource) |
| www.testrail.com/blog/create-a-test-plan/ | TestRail<br>How To Create A Test Plan<br>(Article) |
| martinfowler.com/testing/ | Martin Fowler<br>Software Testing Guide<br>(Resource) |
| www.softwaretestinghelp.com | Software Testing Help<br>Practical explanations of testing concepts, plans and test cases<br>(Tutorials) |

 © Pearson Education Limited 2026

| | |
|---|---|
| agilealliance.org/agile101/agile-glossary | Agile Alliance<br>Agile Glossary – Learn the unique terminology used in Agile development from the experts at Agile Alliance.<br>(General reference) |
| www.atlassian.com/software-testing | Atlassian<br>Test planning, test management and reporting concepts<br>(General reference) |
| www.ministryoftesting.com | Ministry of Testing<br>Community-led resources on modern testing practices and effectiveness<br>(General reference) |
| mastersoftwaretesting.com/testing-fundamentals/software-testing-life-cycle/test-analysis | Master Software Testing<br>Test Analysis in Software Testing: A Practical Guide to Analyzing Test Results<br>(Article) |
| www.testdevlab.com/blog/how-to-measure-test-effectiveness-metrics | TestDevLab<br>How to Measure the Effectiveness of Your Tests (Key Metrics)<br>(Blog) |
| www.geeksforgeeks.org/software-testing/requirement-traceability-matrix/ | GeeksforGeeks<br>Requirements Traceability Matrix – RTM<br>(Article) |

## Journals and articles

Alam, M.M., Priti, S.I., Fatema, K. and Hasan, M. (2024) 'Ensuring Excellence: A Review of Software Quality Assurance and Continuous Improvement in Software Product Development', *Studies in Big Data*, 163, pp. 331–346. Springer. Available at: doi.org/10.1007/978-3-031-73632-2_28.

Li, Q., Zhang, L. and Liu, S. (2024) 'Improving the Efficiency of Software Reliability Demonstration Testing by Introducing Testing Effectiveness', *Symmetry*, 16(10), 1334. Available at: doi.org/10.3390/sym16101334.

Thakare, M.V. and Gunjal, M.G. (2025) 'An Overview of Software Testing Methodologies and Best Practices', *International Journal of Advanced Scientific Research*, 10(3), pp. 108–111. Available at: www.allscientificjournal.com/assets/archives/2025/vol10issue3/10074.pdf.

Verma, A.S., Choudhary, A. and Tiwari, S. (2023) 'Software test case generation tools and techniques: A review', I*nternational Journal of Mathematical, Engineering and Management Science*s, 8(2), pp. 293–315. Available at: doi.org/10.33889/IJMEMS.2023.8.2.018.

## Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 405: Core Software Design and Development*

*Unit 407: Applied Software Testing.*

# Unit 407: Applied Software Testing

**Unit code:** K/652/3259

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Software testing is a fundamental part of modern software development, helping organisations ensure that digital systems are reliable, secure and fit for purpose. As software is used across almost every sector, including finance, healthcare, retail and digital services, effective testing plays a key role in reducing risk, improving quality and supporting user confidence. This unit explores how testing is carried out in professional environments and how tools and automation are used to support consistent and repeatable testing practices.

The aim of this unit is to develop students' understanding of how software testing tools, automation and emerging artificial intelligence techniques are applied within real-world testing activities. Students will gain insight into how testing supports the wider software development life cycle and how structured testing approaches contribute to the delivery of robust and maintainable systems.

The unit covers a range of topics that reflect current industry practice. Students will review different categories of software testing tools used for managing tests, tracking defects, executing automated tests and reporting results. The unit also examines how automation and AI are used to improve testing efficiency and coverage, before progressing to the design and implementation of automated test types within a defined testing scenario using an automated testing platform.

On successful completion of this unit, students will have developed both knowledge and practical skills that are directly relevant to entry-level roles in software testing and quality assurance. They will be able to review testing tools, apply automation and AI concepts, design automated tests and implement an automated testing solution. These skills support progression into roles such as Software Tester, Quality Assurance Technician, Test Analyst, Junior Automation Engineer or Software Development Technician, as well as further study in software engineering or related disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Review the range of tools available to support software testing

LO2 Describe how automation and AI can be applied to software testing

LO3 Design a series of automated software test types for a specified software testing scenario

LO4 Implement the designed automated test types using an automated testing platform for the specified software testing scenario.

## Essential Content

LO1 **Review the range of tools available to support software testing**

*Test management and defect tracking tools:*

Test planning and organisation: test planning support e.g. test schedules, resource planning; test case organisation e.g. test suites, test libraries

Defect handling: defect logging e.g. defect records, issue entries; workflow tracking e.g. status transitions, resolution stages

Traceability and collaboration: traceability e.g. requirement links, test coverage; collaboration features e.g. comments, shared dashboards.

*Automation and execution tools:*

Automation support: automation frameworks e.g. test frameworks, automation libraries; scripting support e.g. test scripts, code editors

Execution management: execution scheduling e.g. timed runs, triggered execution

Compatibility support: cross-browser support e.g. browser variations, rendering differences; platform compatibility e.g. operating systems, device types.

*Integration tools:*

Version control integration: source code repositories e.g. version history, change tracking; collaboration support e.g. shared repositories, team workflows

Environment management tools: test environment configuration e.g. environment variables, configuration files; environment consistency e.g. aligned test conditions, controlled setups.

Workflow integration: tool interoperability e.g. linking test tools with development tools; execution co-ordination e.g. synchronised test runs, shared outputs.

*Reporting and monitoring tools:*

Results and reporting: test result dashboards e.g. pass-fail summaries, execution status; execution reports e.g. run reports, test outcomes

Quality measurement: defect metrics e.g. defect counts, defect severity; quality indicators e.g. defect density, test effectiveness

Analysis and trends: trend analysis e.g. defect trends, execution history.

*Real-world examples:*

A range of real-world software testing tools e.g.

Test management and defect tracking tools: Jira e.g. test tracking, issue management; Azure DevOps e.g. test planning, workflow tracking; Bugzilla e.g. defect logging, issue tracking

Automation and execution tools: Selenium e.g. browser automation, test scripting; Cypress e.g. end-to-end testing, UI testing; Playwright e.g. cross-browser testing, automated execution

Integration tools: Git e.g. version control, change tracking; GitHub e.g. repository management, collaboration workflows; Azure DevOps e.g. pipeline integration, development workflows

Reporting and monitoring tools: Allure e.g. test reporting, result visualisation; ReportPortal e.g. test analytics, defect insights; Grafana e.g. monitoring dashboards, performance visualisation.

LO2 **Describe how automation and AI can be applied to software testing**

*Automation in software testing:*

Automation advantages: automation benefits e.g. reduced manual effort, improved repeatability; test consistency e.g. standardised outcomes, reduced human error

Testing effectiveness: regression testing e.g. repeated test execution, checking changes; testing efficiency e.g. faster execution, reduced effort; scalability awareness e.g. running many tests, repeated test cycles.

*Test automation approaches:*

Automation foundations: test scripting e.g. scripted test cases, reusable scripts; automation frameworks e.g. structured frameworks, tool-supported execution; automated execution e.g. scheduled tests, unattended runs

Test structure design: modularity e.g. reusable components, clear test structure; data-driven testing e.g. external test data, parameterised tests.

*Continuous integration and delivery context:*

Pipeline execution: continuous integration/continuous deployment (CI/CD) pipelines e.g. automated builds, automated test runs; execution stages e.g. build testing, deployment testing

Workflow alignment: build validation e.g. smoke tests, basic build checks; workflow integration e.g. development pipelines, testing stages; feedback cycles e.g. test reports, defect feedback.

*Artificial Intelligence applications in testing:*

Test Intelligence: test generation e.g. automated test creation, requirement-based tests; test prioritisation e.g. ordering tests by importance, focusing on changed areas

Result analysis: anomaly detection e.g. unusual failures, inconsistent results; pattern recognition e.g. repeated failures, common defect types.

*AI-supported test maintenance:*

Execution stability: self-healing locators e.g. adaptive selectors, user interface (UI) change handling; dynamic waits e.g. intelligent timing, synchronisation handling

Maintenance efficiency: test instability e.g. flaky tests, unreliable execution; maintenance reduction e.g. automated updates, reduced rework; long-term reliability e.g. stable test suites, reduced maintenance effort.

*AI techniques in testing contexts (application focus):*

Requirement interpretation: natural language processing e.g. reading requirements, identifying test conditions; test requirements e.g. supporting test case creation, improving traceability

Visual analysis: visual comparison e.g. checking layout consistency, screenshot comparison; pattern recognition e.g. spotting UI changes, identifying visual issues.

*Real-world examples:*

A range of real-world software testing tools e.g.

CI/CD and pipeline tools: Jenkins e.g. automated builds, continuous integration; GitHub Actions e.g. workflow automation, pipeline execution; Azure DevOps e.g. CI/CD pipelines, deployment workflows

Test frameworks: TestNG e.g. structured test execution, test configuration; JUnit e.g. unit testing, test validation; pytest e.g. Python testing, test automation

AI-supported testing tools: Applitools e.g. visual testing, UI validation; ReportPortal e.g. test analytics, anomaly detection; BrowserStack Test Observability e.g. test insights, execution monitoring

AI-supported maintenance tools: Applitools e.g. visual comparison, UI change detection; Tricentis Tosca e.g. model-based testing, automation management; TestSigma e.g. test automation, maintenance reduction.

LO3 **Design a series of automated software test types for a specified software testing scenario**

*Automated test types:*

Purpose and parameters of automated testing: test purpose e.g. defect identification, requirement validation; test parameters e.g. input data, expected results; execution conditions e.g. environment configuration, test data setup

Unit tests: component behaviour e.g. individual functions, isolated logic; automated execution e.g. repeatable runs, fast feedback

Integration tests: component interaction e.g. data flow between modules, interface connections; dependency handling e.g. service stubs, controlled test environments; execution sequencing e.g. ordered test runs, dependency awareness

System tests: end-to-end behaviour e.g. full workflow testing, system functionality; environment awareness e.g. test environments, deployment configuration

Acceptance tests: user requirements validation e.g. business rules, expected outcomes; scenario coverage e.g. real-world usage, user journeys.

*Test design considerations:*

Test coverage requirements: functional coverage e.g. key features, expected behaviour; non-functional awareness e.g. performance expectations, reliability checks

Automated testing protocol: test execution protocol e.g. predefined steps, scripted workflows; validation protocol e.g. expected results comparison, pass fail criteria; defect handling protocol e.g. error logging, issue tracking

Risk-based testing: risk identification e.g. critical features, failure impact; test focus e.g. high-risk areas, priority functions; defect prevention e.g. early testing, focused validation

Test data requirements: representative data e.g. valid inputs, realistic values; data variation e.g. boundary values, alternative cases

Test automation strategy: automation strategy planning e.g. tool selection, framework definition; test coverage strategy e.g. prioritised test cases, critical path testing; maintenance strategy e.g. reusable scripts, test updates; scalability strategy e.g. CI/CD integration, parallel execution.

*Scenario-based test design:*

Functional requirements alignment: requirement mapping e.g. test cases linked to requirements, coverage tracking; validation focus e.g. expected outputs, acceptance criteria

Non-functional requirements awareness: performance considerations e.g. response times, throughput; usability considerations e.g. user interaction flows, accessibility checks

Environment constraints: platform limitations e.g. operating systems, browser versions; configuration constraints e.g. environment settings, test conditions.

*Test documentation and structure:*

Test case specifications: clear steps e.g. preconditions, execution steps; expected results e.g. pass fail criteria, outcome descriptions

Naming conventions: readable identifiers e.g. consistent naming, descriptive titles; traceability support e.g. requirement references, test IDs

Maintainability principles: modular tests e.g. reusable steps, shared components; reusability considerations e.g. adaptable tests, minimal duplication.

LO4 **Implement the designed automated test types using an automated testing platform for the specified software testing scenario**

*Automated testing platform definition:*

Platform purpose: automated testing support e.g. repeatable execution, consistent results; applied scope e.g. defined scenario, controlled environment

Platform composition: combined tools e.g. test frameworks, execution tools; supporting elements e.g. test scripts, reporting components.

*Automated testing platform components:*

Test frameworks: framework structure e.g. organised test suites, reusable logic; execution support e.g. running tests, managing outcomes

Test execution engines: automated runs e.g. triggered execution, scheduled runs; environment awareness e.g. test environments, configuration settings

Reporting modules: result reporting e.g. pass fail summaries, execution results; visibility support e.g. test status, defect indicators.

*Platform structure and configuration:*

Modular organisation: separated components e.g. test logic, configuration files; maintainability support e.g. easy updates, reduced duplication

Configuration management: environment settings e.g. test parameters, system variables; execution control e.g. run options, test selection.

*Integration and operation:*

Toolchain integration: connected tools e.g. frameworks with CI tools, reporting integration; data flow e.g. test results, execution feedback

Execution scheduling: automated triggers e.g. timed runs, pipeline triggers; operational consistency e.g. repeatable processes, reliable execution.

*Platform effectiveness considerations:*

Reliability of execution: stable test runs e.g. consistent outcomes, minimal failures; error handling e.g. clear failure reporting, recoverable issues

Consistency of results: repeatable outcomes e.g. same inputs, same results; quality support e.g. confidence in test outputs

Alignment with testing requirements: requirement coverage e.g. linked tests, scenario needs; quality contribution e.g. defect detection, assurance support.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Review the range of tools available to support software testing</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a selection of real-world examples, the effectiveness of identified manual, automated and AI testing tools in meeting testing requirements within a given testing context.</td></tr>
<tr><td>P1 Review, using a selection of real-world examples, the range of tools used to support the management, tracking, automation and execution of software testing across different stages of the testing process.<br><br>P2 Explore, using a selection of real-world examples, the range of tools used to support the integration, reporting and monitoring of software testing across different stages of the testing process.</td><td>M1 Analyse, using a selection of real-world examples, the strengths and limitations of software testing tools in supporting effective system and application testing.</td></tr>
<tr><td colspan="2">LO2 Describe how automation and AI can be applied to software testing</td></tr>
<tr><td>P3 Describe, using a selection of real-world examples, how automation can be applied to software testing to support continuous integration and delivery.<br><br>P4 Review, using a selection of real-world examples, how artificial intelligence techniques can be applied to support software testing.</td><td>M2 Assess, using a selection of real-world examples, how automation and artificial intelligence testing contributes to the development of robust and reliable systems and applications.</td></tr>
</table>

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO3 Design a series of automated software test types for a specified software testing scenario</td><td rowspan="5">LO3 and LO4<br><br>D2 Justify the decisions made in the design and implementation of the automated test types in thoroughly addressing the functional and non-functional requirements of the specified software testing scenario.</td></tr>
<tr><td>P5 Design a series of automated test types that tests the functional and non-functional requirements of a specified software testing scenario.<br><br>P6 Review the designed automated test types for a given software testing scenario to inform improvements.</td><td>M3 Optimise, using the results of the review, the designed automated test types to ensure that the requirements of the specified software scenario have been thoroughly addressed.</td></tr>
<tr><td colspan="2">LO4 Implement the designed automated test types using an automated testing platform for the specified software testing scenario</td></tr>
<tr><td>P7 Implement the designed automated test types using an automated testing platform for the specified software testing scenario.<br><br>P8 Review the results of the automated testing for the given software testing scenario, considering the functional and non-functional requirements of the scenario.<br><br>P9 Present the results of the automated testing for the given software testing scenario to an identified audience obtaining relevant feedback.</td><td>M4 Evaluate, using the results of the review and audience feedback, the effectiveness of the test results to identify strengths and areas for improvement in the automated testing plan.</td></tr>
</table>

## Recommended Resources

### Textbooks

Bierig, R. *et al.* (2021) *Essentials of Software Testing*. Cambridge: Cambridge University Press.

Black, R., Davenport, J., Olszewska, J. and Wright, J. (2023) *Artificial Intelligence and Software Testing: Building Systems You Can Trust*. Swindon: BCS Learning & Development.

Crispin, L. and Gregory, J. (2023) *Agile Testing: A Practical Guide for Testers and Agile Teams*. 2nd edn. Boston: Addison-Wesley.

Parsa, S. (2023) *Software Testing Automation: Testability Evaluation, Refactoring, Test Data Generation and Fault Localization*. Cham: Springer.

Winteringham, M. (2024) *Software Testing with Generative AI*. New York: Manning Publications.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.browserstack.com/guide/test-automation-workflow | BrowserStack<br>Test Automation Workflow: A Step-by-Step Guide to Efficient Testing<br>(Article) |
| www.softwaretestingmagazine.com/knowledge/how-to-use-ai-in-software-testing-for-smarter-qa/ | Software Testing Magazine<br>How to Use AI in Software Testing for Smarter QA<br>(Article) |
| www.browserstack.com/guide/devops-selenium | BrowserStack<br>Automation testing, CI/CD pipelines and cross-platform testing practices<br>(Tutorials) |
| www.browserstack.com/guide/automation-pipeline | BrowserStack<br>Automation pipelines and testing best practice<br>(Tutorials) |

| | |
|---|---|
| www.geeksforgeeks.org/software-testing/automation-testing-software-testing/ | GeeksforGeeks<br>Software testing automation, CI/CD tools and testing concepts<br>(Resource) |
| www.geeksforgeeks.org/software-testing/ai-testing/ | GeeksforGeeks<br>What is AI testing?<br>(Resource) |
| testsigma.com/tutorials/ | Testsigma<br>A complete collection of tutorials guide you through test automation in Testsigma.<br>(Tutorials) |
| www.softwaretestinghelp.com/automation-testing-tutorial-2/ | Software Testing Help<br>Types of Automation Testing<br>(Tutorial) |

## Journals and articles

Escalante-Viteri, A. and Mauricio, D. (2025) 'Artificial Intelligence in Software Testing: A Systematic Review of a Decade of Evolution and Taxonomy', *Algorithms*, 18(11), 717. Available at: doi.org/10.3390/a18110717.

Garousi, V. *et al.* (2024) 'AI-powered software testing tools: A systematic review and empirical assessment of their features and limitations', *arXiv*. Available at: arxiv.org/abs/2409.00411.

Katru, C.R., Gami, S.J. and Shah, K.N. (2025) 'Automation for the future: Harnessing AI and ML to reshape software testing and maintenance', *International Journal of Computer Trends and Technology*, 73(7), pp. 63–72. Available at: ijcttjournal.org/archives/ijctt-v73i7p108.

Singh, H.R. (2025) 'Harnessing the Power of AI in Software Testing', *International Journal of Scientific and Research Publications*, 15(7). Available at: doi.org/10.29322/IJSRP.15.07.2025.p16312.

## Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 405: Core Software Design and Development*

*Unit 406: Core Software Testing.*

# Unit 408: Applied Software Development

**Unit code:** R/652/3260

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Software applications play a central role in modern organisations, supporting everything from routine business operations to complex digital services. As demand for reliable and well-structured software continues to grow, there is an increasing need for practitioners who understand not only how to write code, but how to apply appropriate development approaches within real-world contexts. This unit introduces students to applied software development, focusing on how different programming paradigms are used to design and build effective software solutions.

The aim of this unit is to develop students' understanding of how commonly used programming paradigms contribute to the development of modern software systems. Students will explore object-oriented, event-driven and functional programming approaches, and examine how each paradigm supports different software requirements. As well as this theoretical understanding, students will apply industry-recognised development practices to produce a programmed solution that reflects professional standards.

The unit covers a range of fundamental topics that support practical software development. Students will study object-oriented principles and patterns used to structure and maintain software applications, event-driven techniques that enable user interaction and responsive behaviour, and functional programming approaches commonly used in data processing and modern development environments. The unit also introduces essential software development practices, including version control, testing, debugging and basic deployment considerations, which are required to produce maintainable and professional solutions.

On successful completion of this unit, students will have developed a foundation in applied software development and an awareness of how professional software solutions are designed and implemented. These skills support progression into further study or entry-level roles within the digital sector, including Junior Software Developer, Application Developer, Software Tester, Systems Support Analyst, and Technical Support Engineer.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain object-oriented principles and their application in software development

LO2 Describe how event-driven programming principles are used in software development

LO3 Review how functional programming is used in data transformation and software development

LO4 Develop a programmed solution for a given scenario using a range of industry best practices.

## Essential Content

LO1 **Explain object-oriented principles and their application in software development**

*Object-oriented programming concepts:*

Core modelling constructs: classes e.g. blueprints, program structures; objects e.g. instances, program entities; attributes e.g. variables, data fields; methods e.g. functions, behaviours

Encapsulation and inheritance principles: encapsulation e.g. access control, data hiding; abstraction e.g. simplified models, interfaces; inheritance e.g. parent classes, child classes; polymorphism e.g. method overriding, interface implementation.

*Object relationships and structures:*

Structural relationships: class hierarchies e.g. superclass structures, subclass structures; interfaces e.g. contracts, method specifications; dependency relationships e.g. class dependencies, module dependencies

Object composition relationships: associations e.g. object links, references; aggregation e.g. whole –part relationships, shared ownership; composition e.g. strong ownership, life cycle dependency.

*Application of object-oriented principles:*

Code organisation principles: code modularity e.g. separate modules, reusable components; separation of concerns e.g. layered design, responsibility division; reuse e.g. shared code, libraries

Software quality characteristics: maintainability e.g. readable code, structured code; scalability e.g. system growth, increased demand; real-world modelling e.g. business entities, system concepts.

*Object-oriented patterns:*

Pattern concepts and examples: pattern meaning e.g. reusable solutions, design guidance; factory e.g. object creation, instance control; singleton e.g. single instance, shared access; facade e.g. simplified interfaces, system access; strategy e.g. interchangeable algorithms, behaviour selection.

*Object-oriented design in practice:*

Design and system application: common design considerations e.g. performance factors, usability factors; alignment with software requirements e.g. functional needs, non-functional needs; application within software systems e.g. enterprise systems, application software; domain models e.g. conceptual models, class diagrams; layered architectures e.g. presentation layer, business layer.

LO2 **Describe how event-driven programming principles are used in software development**

*Event-driven programming fundamentals:*

Event flow elements: events e.g. clicks, key presses; event sources e.g. buttons, input fields; event listeners e.g. registered components, observers; event handlers e.g. response methods, handling routines; callbacks e.g. return functions, triggered functions.

*User interface and interaction models:*

Interaction mechanisms: graphical user interfaces (GUIs) e.g. windows, menus; user actions e.g. selections, inputs; system responses e.g. screen updates, notifications; accessibility considerations e.g. keyboard access, screen readers; responsiveness e.g. fast feedback, smooth interaction.

*Event-driven architectures and patterns:*

Architectural structures: Model–View–Controller (MVC) e.g. separation of layers, interface logic; observer pattern e.g. notification mechanisms, state updates; decoupling of components e.g. loose coupling, independent modules.

*Asynchronous and concurrent behaviour:*

Execution behaviour: asynchronous execution e.g. background tasks, delayed processing; non-blocking operations e.g. UI continuation, task queues; event queues e.g. message queues, task lists; concurrency considerations e.g. shared resources, execution order.

LO3 **Review how functional programming is used in data transformation and software development**

*Core functional programming concepts:*

Functional characteristics: pure functions e.g. predictable output, no side effects; immutability e.g. unchanging data, fixed values; function composition e.g. chained functions, combined operations; higher-order functions e.g. functions as inputs, functions as outputs; recursion e.g. self-referencing functions, repeated execution.

*Functional data processing techniques:*

Data operations: mapping e.g. data conversion, value transformation; filtering e.g. condition checks, data selection; reducing e.g. aggregated results, combined values; transformations e.g. data reshaping, format changes; pipelines e.g. processing stages, sequential steps; data streams e.g. continuous data, streamed values.

*Functional programming in contemporary software development:*

Software integration: integration with object-oriented approaches e.g. mixed paradigms, hybrid design; use within contemporary languages e.g. modern syntax, language features; lambda expressions e.g. inline functions, concise logic; stream APIs e.g. data streams, collection processing.

*Benefits and limitations of functional programming:*

Performance and suitability considerations: predictability e.g. consistent results, deterministic behaviour; testability e.g. isolated testing, repeatable tests; parallelism support e.g. concurrent tasks, workload distribution; learning curve e.g. unfamiliar concepts, syntax difficulty; suitability for different problem domains e.g. data processing, algorithm design; comparative suitability for data transformation and software development contexts e.g. transformation tasks, application logic.

LO4 **Develop a programmed solution for a given scenario using a range of industry best practices**

*Software development best practices:*

Development standards: version control e.g. code repositories, change tracking; basic branching e.g. feature branches, development branches; coding standards e.g. naming conventions, formatting rules; documentation e.g. code comments, technical guides; code readability e.g. clear structure, consistent layout.

*Development workflows and quality assurance:*

Testing and improvement processes: test plans e.g. testing scope, test objectives; test cases e.g. input conditions, expected outcomes; manual testing e.g. exploratory testing, user checks; automated testing e.g. unit tests, regression tests; test coverage e.g. code paths, execution scope; debugging e.g. fault isolation, error tracing; refactoring e.g. code improvement, structure refinement; continuous integration e.g. build automation, frequent integration.

*Build, integration, and deployment considerations:*

Delivery mechanisms: build pipelines e.g. automated builds, build stages; dependency management e.g. libraries, package versions; environment configuration e.g. development settings, deployment settings; continuous deployment concepts e.g. frequent releases, automated deployment.

*Professional and maintainable software solutions:*

Quality attributes: reliability e.g. consistent behaviour, error handling; maintainability e.g. structured code, readable code; scalability e.g. system expansion, user growth

Alignment and evaluation considerations: alignment with requirements e.g. functional requirements, non-functional requirements; industry expectations for applied software development e.g. professional standards, workplace practices; evaluation considerations e.g. success criteria, review measures; effectiveness against requirements e.g. requirement fulfilment, solution suitability

Design decision considerations: trade-offs in design decisions e.g. performance balance, complexity balance.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain object-oriented principles and their application in software development</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Evaluate, using a selection of programming examples, the effectiveness of object-oriented, event-driven and functional programming principles when applied in a range of software development contexts.</td></tr>
<tr><td>P1 Explain, using a selection of programming examples, the principles, relationships and constructs of object-oriented software.<br><br>P2 Explore, using a selection of programming examples, how object-oriented patterns and design principles can be applied in a range of software development contexts.</td><td>M1 Assess, using a selection of programming examples, the suitability of object-oriented patterns and principles in a range of software development contexts.</td></tr>
<tr><td colspan="2">LO2 Describe how event-driven programming principles are used in software development</td></tr>
<tr><td>P3 Describe, using a selection of programming examples, how event-driven programming principles are used in the development of Graphical User Interfaces (GUIs).<br><br>P4 Explore, using a selection of programming examples, how event-driven architectures and patterns can be used to support both synchronous and asynchronous behaviour in a range of software applications.</td><td>M2 Analyse, using a selection of programming examples, the use of event-driven programming principles in a range of software development contexts.</td></tr>
<tr><td colspan="2">LO3 Review how functional programming is used in data transformation and software development</td></tr>
<tr><td>P5 Review, using a selection of programming examples, the use of functional programming and data processing in contemporary software development.<br><br>P6 Discuss, using a selection of programming examples, the benefits and limitations of functional programming in a range of software development contexts.</td><td>M3 Analyse, using a selection of programming examples, the use of functional programming in data transformation and software development contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Develop a programmed solution for a given scenario using a range of industry best practices | | |
| **P7** Develop a programmed solution for a given scenario using a range of industry best practices.<br><br>**P8** Implement a quality assurance and deployment plan for the programmed solution to ensure that it aligns with the requirements of the given scenario. | **M4** Evaluate the developed programmed solution against industry best practices to ensure that the solution is professional and maintainable. | **D2** Justify the decisions made in the development of the programmed solution in meeting scenario requirements and industry best practices. |

## Recommended Resources

### Textbooks

Fowler, M. (2024) *Refactoring: Improving the Design of Existing Code*. 3rd edn. Boston: Addison-Wesley.

Gamma, E. *et al*. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA: Addison-Wesley.

McLaughlin, B., Pollice, G. and West, D. (2024) *Head First Object-Oriented Analysis and Design*. 2nd edn. Sebastopol: O'Reilly Media.

Richards, M. and Ford, N. (2025) *Fundamentals of Software Architecture*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Sommerville, I. and Tilley, S. (2024) *Software Engineering*. 11th edn. Harlow: Pearson.

Widman, J. (2022) *Learning Functional Programming: Managing code complexity by thinking functionally*. Sebastopol, CA: O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/ | GeeksforGeeks<br>Object Oriented Programming Concepts<br>(Tutorial) |
| realpython.com/python3-object-oriented-programming/ | RealPython<br>Object-oriented programming concepts using Python<br>(Tutorial) |
| learn.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop | Microsoft Learn<br>Object-oriented programming in C#<br>(Tutorial) |
| developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events | MDNWebDocs<br>Introduction to events<br>(Tutorial) |
| learn.microsoft.com/en-us/dotnet/desktop/winforms/forms/events?tabs=dotnet | Microsoft Learn<br>Event handling overview<br>(General reference) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.freecodecamp.org/news/tag/functional-programming/ | freeCodeCamp<br>Functional programming concepts explained<br>(Article) |
| www.baeldung.com/java-functional-programming | Baeldung<br>Functional Programming in Java<br>(Tutorial) |
| www.atlassian.com/continuous-delivery/software-testing | Atlassian<br>Software testing fundamentals<br>(General reference) |
| martinfowler.com/articles/continuousIntegration.html | MartinFowler<br>Continuous Integration<br>(Article) |

**Journals and articles**

Eugster, P.T. *et al*. (2003) 'The many faces of publish/subscribe', *ACM Computing Surveys*, 35(2), pp. 114–131. Available at: doi.org/10.1145/857076.857078.

Hughes, J. (1989) 'Why Functional Programming Matters', *The Computer Journal*, 32(2), pp. 98–107. Available at: doi.org/10.1093/comjnl/32.2.98.

Meyer, B. (1989) 'Object-oriented software construction', *Science of Computer Programming*, 12(1), pp. 1–20. Available at: doi.org/10.1016/0167-6423(89)90034-8.

Raheem, A. *et al.* (2024) 'Exploring continuous integration and deployment strategies for streamlined DevOps processes in software engineering practices', *World Journal of Advanced Research and Reviews*, 24(3), pp. 2813–2830. Available at: doi.org/10.30574/wjarr.2024.24.3.3988.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 403: Professional Practice*

*Unit 404: Digital Systems Analysis*

*Unit 405: Core Software Design and Development*

*Unit 406: Core Software Testing.*

# Unit 409: Core Cybersecurity

**Unit code:** **T/652/3261**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

Cybersecurity plays a critical role in modern organisations as digital systems, networks and data have become central to how services are delivered and businesses operate. Organisations of all sizes rely on secure digital infrastructure to protect sensitive information, maintain system availability and ensure trust with users and stakeholders. As cyber threats continue to increase in frequency and evolve in complexity, a clear understanding of cybersecurity principles has become essential across the computing and wider technology sector.

The aim of this unit is to provide students with a solid foundation in core cybersecurity concepts and practices. It is designed to develop an understanding of how digital systems are protected, why security controls are required, and how regulatory and organisational requirements influence cybersecurity decision-making. The unit supports students in developing the knowledge needed to recognise risks, evaluate threats and contribute to secure system design within a professional context.

The unit covers key areas that underpin effective cybersecurity practice. Students will explore fundamental cybersecurity concepts, including digital assets, vulnerabilities, threats and security controls. They will examine common cybersecurity architectures and models used to protect systems and data, before reviewing a range of cyber threats that affect existing digital environments. The unit then brings this knowledge together through the design of a regulatory-compliant cybersecurity solution for a given organisational scenario.

On successful completion of this unit, students will be able to analyse cybersecurity risks, review threats to digital systems and design appropriate security solutions at a foundational level. These skills support progression into further study within cybersecurity, networking and digital systems, as well as entry-level roles such as Junior Cybersecurity Analyst, IT Support Technician, Network Technician, Systems Administrator, or Risk and Compliance Assistant within the wider digital and IT sector.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the core concepts of cybersecurity

LO2 Explore the core cybersecurity architectures and models used to protect digital systems

LO3 Review a range of cyberthreats to existing digital systems

LO4 Design a regulatory compliant cybersecurity solution for an identified scenario.

## Essential Content

### LO1 Discuss the core concepts of cybersecurity

*Core cybersecurity principles:*

Fundamental principles: confidentiality e.g. access restrictions, data privacy; integrity e.g. data accuracy, unauthorised changes, error prevention; availability e.g. system uptime, backup access; confidentiality, integrity, and availability (CIA) triad e.g. security balance, protection priorities, risk trade-offs

Security context: information assurance e.g. data protection, information reliability, trust management; risk awareness e.g. threat likelihood, impact consideration

Threat landscape understanding: threat awareness e.g. common attacks, attacker behaviour, emerging risks; vulnerability awareness e.g. software flaws, weak configurations; security principles e.g. secure design, defence planning

Security culture assessment: recognised approaches e.g. frameworks, maturity models; assessment methods e.g. structured evaluations, surveys; findings e.g. strengths, weaknesses, improvement areas.

*Digital assets, values and classification:*

Asset categories: hardware assets e.g. servers, network devices; software assets e.g. operating systems, applications, utilities; data assets e.g. databases, files

Ownership and sensitivity: asset ownership e.g. system owners, responsible roles; asset value e.g. business importance, operational dependence, replacement cost; information sensitivity e.g. confidential data, restricted information

Data types and importance: data classification e.g. public data, confidential data; personal data e.g. names, contact details, identifiers; critical systems e.g. payment systems, essential services.

*Vulnerabilities and threats:*

Technical weaknesses: software vulnerabilities e.g. unpatched software, insecure code; hardware vulnerabilities e.g. exposed ports, ageing equipment

Human and configuration issues: human factors e.g. user error, poor awareness, unsafe behaviour; configuration weaknesses e.g. default settings, incorrect setup

Threat characteristics: threat sources e.g. external attackers, insiders; intentional threats e.g. targeted attacks, malicious activity, planned breaches; accidental threats e.g. human error, system failure.

*Security controls, safeguards and conceptual models:*

Control types: administrative controls e.g. policies, training; technical controls e.g. firewalls, authentication systems, encryption; physical controls e.g. locks, secure locations

Control purposes: preventive controls e.g. access restrictions, security hardening; detective controls e.g. monitoring, intrusion detection

Security concepts and access: defence-in-depth e.g. layered security, multiple controls, overlapping protection; least privilege e.g. minimal access, permission control; access control e.g. authentication, authorisation

Security requirements (functional and non-functional) e.g. access control, confidentiality, integrity; conflicting design constraints e.g. usability, cost, performance, operational feasibility; trade-off decisions e.g. control suitability, proportionality, balancing protection with system and business requirements.

LO2 **Explore the core cybersecurity architectures and models used to protect digital systems**

*Network security architectures:*

Network boundaries: perimeter security e.g. firewalls, network borders; demilitarised zones (DMZ) e.g. DMZs, controlled access areas, buffer networks; network segmentation e.g. virtual local area networks (VLANs), isolated networks

Trust models: trusted networks e.g. internal systems, corporate networks; untrusted networks e.g. public networks, external connections; external access e.g. remote access, internet connections.

*Security models and architectural approaches:*

Security strategies: layered security e.g. multiple controls, defence layers, overlapping protection; zero trust architecture e.g. continuous verification, no implicit trust

Access control approaches: access control models e.g. rule-based access, policy enforcement; role-based access control e.g. job roles, permission sets, user groups; attribute-based access control e.g. user attributes, context rules

Identity and access processes: authentication models e.g. passwords, biometrics; authorisation models e.g. access approval, permission checks.

*System and data protection mechanisms:*

Access assurance: authentication e.g. passwords, login credentials; authorisation e.g. permission levels, access rules; identity management e.g. user identities, account management, life cycle control

Data protection methods: encryption e.g. data encryption, secure storage; multi-factor authentication e.g. SMS codes, authenticator apps.

*Infrastructure protection strategies and constraints:*

Protection measures: endpoint protection e.g. anti-malware software, device security; secure configuration e.g. hardened systems, minimal services, secure settings

Design and trade-offs: system resilience e.g. fault tolerance, recovery planning; scalability e.g. system growth, user expansion, capacity planning; cost considerations e.g. budget limits, resource constraints.

LO3 **Review a range of cyberthreats to existing digital systems**

*Malware-based threats:*

Malware types: viruses e.g. file infection, system damage; trojans e.g. hidden software, unauthorised access, deceptive applications; ransomware e.g. data encryption, ransom demands

Automated threats: spyware e.g. data monitoring, information leakage; botnets e.g. remote control networks, automated attacks, co-ordinated activity.

*Network-based threats:*

Service disruption attacks: denial-of-service attacks e.g. service overload, resource exhaustion, traffic flooding; distributed denial-of-service attacks e.g. multiple sources, co-ordinated traffic

Interception techniques: person-in-the-middle attacks e.g. data interception, session manipulation, Domain Name System (DNS) poisoning; spoofing e.g. fake identities, address impersonation.

*Internal Human Threats:*

Deception techniques: phishing e.g. email scams, fake messages; spear phishing e.g. targeted emails, personalised attacks, impersonation

Communication attacks: vishing e.g. phone scams, voice deception; smishing e.g. SMS scams, text-based deception.

*System exploitation techniques:*

Software and configuration issues: unpatched vulnerabilities e.g. outdated software, missing updates; misconfigurations e.g. incorrect settings, open services, exposed systems

Credential weaknesses: weak credentials e.g. simple passwords, reused passwords

Third-party and supply chain threats: software supply chain attacks e.g. compromised software updates, malicious code injection; Managed Service Provider (MSP) compromise e.g. trusted supplier access, outsourced service breaches; vendor risks e.g. third-party vulnerabilities, supplier ecosystem weaknesses.

*Threat actors:*

External attackers: cybercriminals e.g. financial crime, data theft; organised crime groups e.g. co-ordinated attacks, profit-driven operations

Internal and emerging actors: insiders e.g. employee misuse, internal threats; automated threats e.g. scripts, artificial intelligence agents, automated tools.

*Threat impact considerations:*

Data and service impact: data loss e.g. deleted files, corrupted data; service disruption e.g. system outages, downtime, availability loss

Business consequences: financial loss e.g. recovery costs, lost revenue, fines; reputational damage e.g. loss of trust, public perception, customer confidence.

LO4 **Design a regulatory compliant cybersecurity solution for an identified scenario**

*Security requirements analysis:*

Organisational context: organisational context e.g. business environment, operational setting; business requirements e.g. functional needs, service goals, performance expectations; regulatory requirements e.g. legal duties, compliance obligations, statutory controls; business continuity planning e.g. continuity strategies, service recovery planning, maintaining operations during disruption

Risk and asset considerations: risk considerations e.g. threat likelihood, impact assessment; asset protection needs e.g. data protection, system safeguarding; data sensitivity e.g. confidential data, personal information, restricted records

Risk assessment: structured risk evaluation, recognised methodologies e.g. NIST Risk Management Framework (RMF), Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE), Factor Analysis of Information Risk (FAIR); recognised frameworks e.g. ISO/IEC 27005, NIST Cybersecurity Framework (CSF), ISO/IEC 31000

Life cycle management: service management practices e.g. incident management, change management, service operation; service life cycle e.g. service strategy, design, transition, operation, continual service improvement; core practices e.g. incident management, change enablement, problem management.

*Regulatory, legal and compliance considerations:*

UK and international legal frameworks: Data Protection Act 2018; General Data Protection Regulation (GDPR); Computer Misuse Act 1990; legislation governing lawful access, intelligence gathering and surveillance including Intelligence Services Act 1994 and Regulation of Investigatory Powers Act 2000

Cyber security standards and frameworks: ISO/IEC 27001 information security management systems; Cyber Essentials; principles of confidentiality, integrity and availability (CIA triad)

Personal data protection and privacy requirements: lawful basis for processing; data subject rights; data minimisation; data security; breach reporting requirements; accountability and organisational responsibility

Organisational governance: organisational policies e.g. security policies, acceptable use rules; compliance requirements e.g. audit needs, reporting obligations, evidence requirements; governance requirements e.g. accountability, oversight structures

Application of standards in practice: risk-based approach; security controls selection; policy development; governance structures; alignment of organisational processes to recognised standards.

Knowledge currency, advancements, contemporary practice

*Security architecture design components:*

Structural design: network architecture e.g. network boundaries, secure zones; system architecture e.g. system layout, component interactions; boundary protections e.g. firewalls, gateway controls

Access and protection mechanisms: access control structures e.g. role-based access, permission models; authentication mechanisms e.g. passwords

*Security control selection and justification:*

Control decision-making: control selection e.g. appropriate controls, relevant safeguards; control suitability e.g. system fit, risk relevance, proportionality

Alignment and constraints: risk alignment e.g. controls matched to risks, threat mitigation; compliance alignment e.g. regulatory fit, policy compliance, standard adherence; security case development e.g. linking requirements to controls, justification of security measures, risk-aligned design decisions.

*Solution effectiveness considerations:*

Risk and compliance outcomes: risk reduction e.g. threat mitigation, exposure reduction; compliance coverage e.g. legal requirements, policy fulfilment, audit readiness

Operational viability: operational feasibility e.g. manageability, staff capability; system performance e.g. system efficiency, reliability; cost impact e.g. budget constraints, resource use.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Discuss the core concepts of cybersecurity</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Evaluate, using a selection of real-world examples, how a thorough understanding of cybersecurity concepts, architectures and threats can effectively protect networked systems against evolving threats.</td></tr>
<tr><td>P1 Discuss, using a selection of real-world examples, the core concepts of cybersecurity and the digital assets they impact.<br><br>P2 Review, using a selection of real-world examples, the different threats and vulnerabilities that affect digital systems and the security controls and models used to protect them.</td><td>M1 Analyse, using a selection of real-world examples, how core cybersecurity concepts are applied to secure digital assets within identified organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Explore the core cybersecurity architectures and models used to protect digital systems</td></tr>
<tr><td>P3 Explore, using a selection of real-world examples, the core cybersecurity architectures and models used to protect digital systems.<br><br>P4 Review, using a selection of real-world examples, the mechanisms used to protect the data and infrastructure of digital systems.</td><td>M2 Assess, using a selection of real-world examples, the strengths and limitations of different cybersecurity architectures, models and protection mechanisms in securing digital systems.</td></tr>
<tr><td colspan="2">LO3 Review a range of cyberthreats to existing digital systems</td></tr>
<tr><td>P5 Review, using a selection of real-world examples, a range of cyberthreats to existing digital systems.<br><br>P6 Discuss, using a selection of real-world examples, the ways that cyberthreats can impact existing digital systems and organisations.</td><td>M3 Analyse, using a selection of real-world examples, the severity of the impact of different cyberthreats on existing digital systems and organisations.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Design a regulatory compliant cybersecurity solution for an identified scenario | | |
| **P7** Design a regulatory compliant cybersecurity solution for an identified scenario.<br><br>**P8** Present the regulatory compliant cybersecurity solution for an identified scenario to a group of internal stakeholders to obtain feedback on strengths and areas for improvement. | **M4** Optimise, using the results of the stakeholder feedback, the proposed cybersecurity solution so that it effectively meets the regulatory and security requirements of the identified organisational scenario. | **D2** Evaluate the effectiveness of the optimised cybersecurity solution in comprehensively meeting the regulatory and security requirements of the identified organisational scenario. |

## Recommended Resources

### Textbooks

Dove, N. (2022) *Learning Cybersecurity Fundamentals*. Sebastopol, CA: O'Reilly Media.

Easttom, W. (2023) *Computer Security Fundamentals*. 5th edn. Indianapolis, IN: Pearson IT Certification.

Engemann, K.J. and Witty, J.A. (2024) *Cybersecurity Risk Management: Enhancing Leadership and Expertise*. Boca Raton, FL: CRC Press.

Kim, D. and Solomon, M.G. (2021) *Fundamentals of Information Systems Security*. 4th edn. Burlington, MA: Jones & Bartlett Learning.

Stallings, W. (2023) *Cryptography and Network Security: Principles and Practice*. 8th edn. Harlow: Pearson.

Pfleeger, C.P., Pfleeger, S.L. and Coles-Kemp, L. (2023) *Security in Computing*. 6th edn. Boston, MA: Addison-Wesley.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.cisa.gov/topics/cybersecurity-best-practices | Cybersecurity and Infrastructure Security Agency (CISA)<br>Foundational cybersecurity practices and organisational cyber hygiene<br>(General reference) |
| www.ncsc.gov.uk/collection/cyber-security-design-principles | UK National Cyber Security Centre (NCSC)<br>Guidance outlining core secure by design principles for building resilient digital systems and services.<br>(General reference) |
| ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/security/ | Information Commissioner's Office (ICO)<br>UK GDPR Guidance on Cybersecurity, Information Security, and Organisational Responsibilities<br>(General reference) |
| www.cloudflare.com/learning/ddos/glossary/malware/ | Cloudflare<br>What Is Malware?<br>(General reference) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.cisa.gov/topics/cyber-threats-and-advisories/malware-phishing-and-ransomware | Cybersecurity and Infrastructure Security Agency (CISA)<br>Overview of common cyberthreats<br>(General reference) |
| www.nist.gov/cyberframework | National Institute of Standards and Technology (NIST)<br>Cybersecurity Framework 2.0 overview and risk-based security principles<br>(Framework/General reference) |
| www.nist.gov/itl/smallbusinesscyber/cybersecurity-basics | National Institute of Standards and Technology (NIST)<br>Introductory cybersecurity principles and risk awareness for organisations<br>(General reference) |
| owasp.org/Top10/2025/0x00_2025-Introduction/ | OWASP Foundation<br>The most common categories of software security weaknesses.<br>(General reference) |
| www.ncsc.gov.uk/guidance/phishing | UK National Cyber Security Centre (NCSC)<br>Phishing attacks: defending your organisation<br>(General reference) |
| ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/security/a-guide-to-data-security/ | Information Commissioner's Office (ICO)<br>A guide to data security<br>(Regulatory guidance) |
| www.cybok.org/resources/ | CyBOK<br>Cybersecurity body of knowledge resource base<br>(Resource Hub) |

**Journals and articles**

Del-Real, C., De Busser, E. and van den Berg, B. (2025) 'Security and privacy by design principles for digital technologies', *International Review of Law, Computers & Technology*. Available at: doi.org/10.1080/13600869.2025.2457227.

Gadhi, A. *et al*. (2024) 'Cyber Resilience through Real-Time Threat Analysis in Information Security', *International Journal of Communications, Network and System Sciences*, 17(4), pp. 51–67. doi.org/10.4236/ijcns.2024.174004.

Gambo, M.L. and Almulhem, A. (2025) 'Zero Trust Architecture: A Systematic Literature Review', *Journal of Network and Systems Management*, 34, Article 25. Available at: doi.org/10.1007/s10922-025-09998-x.

Santos, P. *et al*. (2025) 'A systematic review of cyber threat intelligence', *Sensors*, 25(14), Article 4272. Available at: doi.org/10.3390/s25144272.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 403: Professional Practice*

*Unit 404: Digital Systems Analysis*

*Unit 405: Core Software Design and Development*

*Unit 406: Core Software Testing*

*Unit 408: Applied Software Development.*

 © Pearson Education Limited 2026

# Unit 410: Database Engineering

**Unit code:** **Y/652/3262**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

Modern organisations depend on well-engineered databases to support operational systems, digital services and data-driven decision-making across sectors such as finance, healthcare, government and e-commerce. As data volumes grow and system demands increase, organisations require robust database solutions that are secure, scalable and aligned with business needs. This unit explores database engineering from an enterprise perspective, focusing on how databases are designed, structured and implemented to support reliable information systems.

The aim of this unit is to develop students' understanding of how enterprise databases are engineered to meet organisational and technical requirements. Students will examine database architectures, data modelling principles and relational database concepts, gaining insight into how design and implementation decisions influence performance, integrity, security and maintainability within real-world scenarios.

The unit covers key concepts that underpin professional database practice. Students will review a range of database architectures used in enterprise environments, explore the principles of developing normalised data models, and design a structured relational database schema for a given scenario. The unit also focuses on the practical implementation of relational databases using industry tools, including the application of SQL, transaction management, testing, optimisation, and governance considerations.

On successful completion of this unit, students will be able to analyse database requirements, design structured data solutions and implement enterprise-ready relational databases. These skills support progression into roles such as Database Administrator, Junior Data Engineer, Software Developer, Systems Analyst, and Business Intelligence Technician, as well as further study in database systems, data analytics and software engineering.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Review the range of database architectures available for enterprise solutions

LO2 Explore the key concepts in developing a normalised data model for an identified scenario

LO3 Design a normalised relational database schema for an identified scenario

LO4 Implement a normalised relational database using a range of industry tools for an identified scenario.

## Essential Content

LO1 **Review the range of database architectures available for enterprise solutions**

*Database models:*

Model types: hierarchical model e.g. tree structures, parent-child records; network model e.g. graph links, multiple parent records; relational model e.g. tables, rows and columns

NoSQL models: key–value e.g. cache storage, session data; document e.g. JSON documents, semi-structured data; column-family e.g. wide column storage, analytical queries; graph e.g. node relationships, network analysis

Design characteristics: structural characteristics e.g. data organisation, relationship representation; use cases e.g. transaction processing, analytics scenarios; enterprise suitability e.g. scalability needs, organisational requirements.

*ANSI/SPARC three-level architecture:*

Architecture levels: external level e.g. user views, application interfaces; conceptual level e.g. logical database structure, entity relationships; internal level e.g. physical storage, file organisation

Data separation: separation of user views e.g. tailored access, security isolation; logical structure e.g. data relationships, schema definitions; physical storage e.g. disk layout, indexing methods.

*Schemas and data abstraction:*

Schema types: external schemas e.g. user-specific views, application schemas; conceptual schema e.g. overall database design, global structure; internal schema e.g. storage schema, physical design

Schema relationships: schema mapping e.g. external-to-conceptual mapping, conceptual-to-internal mapping; abstraction layers e.g. logical abstraction, physical abstraction

Data independence: logical data independence e.g. schema changes, application stability; physical data independence e.g. storage changes, performance optimisation.

*Enterprise deployment architectures:*

Deployment models: client–server databases e.g. central servers, remote clients; embedded databases e.g. local applications, device-level storage

Distribution models: distributed databases e.g. multiple locations, replicated data; centralised databases e.g. single data store, controlled access.

*Workload types:*

Transaction workloads: online transaction processing (OLTP) e.g. order processing, real-time updates; operational workloads e.g. day-to-day operations, business transactions

Analytical workloads: online analytical processing (OLAP) e.g. reporting systems, data analysis; analytical workloads e.g. trend analysis, decision support.

LO2 **Explore the key concepts in developing a normalised data model for an identified scenario**

*Requirements and scenario definition:*

Business context: business requirements e.g. organisational needs, operational objectives; data requirements e.g. data types, data sources; assumptions e.g. system constraints, user expectations; constraints e.g. technical limits, regulatory factors; scope definition e.g. system boundaries, project limits; validation requirements e.g. accuracy checks, stakeholder approval.

*Entities and attributes:*

Entity structure: entity identification e.g. real-world objects, data subjects; attribute types e.g. simple attributes, composite attributes; primary attributes e.g. key identifiers, unique values; derived attributes e.g. calculated values, dependent data.

*Relationships and cardinality:*

Relationship types: one-to-one relationships e.g. person-passport, user-profile; one-to-many relationships e.g. customer-orders, department-employees; many-to-many relationships e.g. students-modules, products-suppliers

Relationship constraints: optionality e.g. optional participation, mandatory participation; participation constraints e.g. minimum occurrence, maximum occurrence.

*Entity relationship modelling:*

Modelling approaches: Entity–relationship diagrams (ERD) e.g. conceptual models, logical models; Chen notation e.g. entity symbols, relationship diamonds; crow's foot notation e.g. cardinality markers, relationship lines.

*Normalisation principles:*

Normal forms: First Normal Form (1NF) e.g. atomic attributes, no repeating groups; Second Normal Form (2NF) e.g. full functional dependency, primary key dependency; Third Normal Form (3NF) e.g. removal of transitive dependency, non-key dependency

Advanced forms: Fourth Normal Form (4NF) e.g. multivalued dependency removal, independent attribute separation, elimination of redundant data; Fifth Normal Form (5NF) e.g. join dependency elimination, lossless table decomposition, reconstruction integrity; Sixth Normal Form (6NF) e.g. irreducible table structures, temporal data separation, single-fact tables; Seventh Normal Form (7NF) e.g. fully constrained schemas, domain-key normal form principles, logical constraint enforcement.

*Business rules and constraints:*

Logic representation: business logic representation e.g. organisational policies, process rules; integrity rules e.g. entity integrity, referential integrity; data validity requirements e.g. acceptable values, data accuracy.

LO3 **Design a normalised relational database schema for an identified scenario**

*Logical database design:*

Schema structure: relational schema structure e.g. table organisation, schema layout; table definitions e.g. column definitions, data storage; attribute domains e.g. value ranges, permitted formats; normalised table structures e.g. reduced redundancy, structured relationships.

*Keys and relationships:*

Key types: primary keys e.g. unique identifiers, entity identification; foreign keys e.g. table links, relationship enforcement; candidate keys e.g. alternative identifiers, uniqueness options; compound keys e.g. multi-attribute identifiers, combined uniqueness; composite keys e.g. related attributes, joint identification

Relationship integrity: referential integrity e.g. consistent references, valid relationships; cascade effects e.g. update cascades, delete cascades.

*Constraints and integrity controls:*

Constraint types: NOT NULL constraints e.g. mandatory fields, required data; UNIQUE constraints e.g. value uniqueness, duplicate prevention; CHECK constraints e.g. value rules, validation conditions; referential constraints e.g. enforced relationships, key consistency.

*Performance and structure considerations:*

Performance factors: indexes e.g. query acceleration, search efficiency; views e.g. virtual tables, controlled data access; table relationships e.g. join paths, relational structure; data types e.g. storage efficiency, value accuracy; collation e.g. sorting rules, character comparison.

*Access and security structure:*

Access control: role-based access control e.g. user roles, permission grouping; user permissions e.g. read access, write access; schema ownership e.g. control responsibility, administrative rights.

LO4 **Implement a normalised relational database using a range of industry tools for an identified scenario**

*Database implementation tools:*

Implementation platforms: Relational Database Management Systems (RDBMS) e.g. structured data storage, table-based systems; Industry tools e.g. MySQL, PostgreSQL, SQL Server.

*Data definition and manipulation:*

Schema operations: use of Data Definition Language (DDL) e.g. table creation, schema definition; schema creation e.g. database structure, object organisation; table alteration e.g. add, modify, drop; deletion e.g. remove tables, remove unused objects; object renaming e.g. tables, columns, schemas; object definitions e.g. views, triggers, indexes

Data operations: Data Manipulation Language (DML) e.g. data insertion, data updates; data population e.g. record insertion, sample data loading.

*Transaction management:*

Transaction control: transactions e.g. grouped operations, database consistency; commit operations e.g. permanent changes, transaction completion; rollback operations e.g. error recovery, transaction reversal

Transaction properties: atomicity e.g. all-or-nothing execution, operation completeness; consistency e.g. data validity, integrity preservation.

*Verification and testing:*

Testing approaches: test data e.g. sample records, realistic datasets; verification queries e.g. correctness checks, validation queries; unit testing e.g. component tests, individual operation checks; system testing e.g. full implementation tests, integrated behaviour.

*Optimisation and maintenance:*

Performance methods: query execution plans e.g. optimiser strategies, execution flow; index tuning e.g. performance improvement, access speed; query optimisation techniques e.g. efficient queries, reduced execution cost.

*Backup and recovery:*

Data protection: backup strategies e.g. full backups, incremental backups; restore procedures e.g. data recovery, system restoration; data availability e.g. continuity planning, access reliability.

*Governance and compliance:*

Regulatory considerations: data protection principles e.g. lawful processing, data security; General Data Protection Regulation (GDPR) considerations e.g. personal data handling, compliance obligations; documentation standards e.g. system records, audit trails; regulatory updates.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Review the range of database architectures available for enterprise solutions</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate the suitability of database architectures and relational database concepts in the development of enterprise solutions that support the business requirements for an identified scenario.</td></tr>
<tr><td>P1 Review, using a selection of real-world examples, the different database architectures and models used in a range of enterprise solutions.<br><br>P2 Describe, using a selection of real-world examples, the different schemas and workload types used in a range of enterprise solutions.</td><td>M1 Analyse, using a selection of real-world examples, the strengths and limitations of database architectures used in a range of organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Explore the key concepts in developing a normalised data model for an identified scenario</td></tr>
<tr><td>P3 Explore, using a selection of real-world examples, the key database concepts involved in developing a normalised data model for an identified scenario.<br><br>P4 Describe, using a selection of real-world examples, the key business concepts involved in developing a normalised data model for an identified scenario.</td><td>M2 Assess, using a selection of real-world examples, the importance of business and database concepts in developing a data model for an identified scenario.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a normalised relational database schema for an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of the relational database schema in how comprehensively it meets the structural and behavioural requirements of the given business scenario. |
| **P5** Design a normalised relational database schema that meets the structural and behavioural requirements of an identified scenario.<br>**P6** Review the relational database schema design with a range of stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the stakeholder feedback, the normalised relational database schema design so that it comprehensively addresses the requirements of the identified scenario. | |
| **LO4** Implement a normalised relational database using a range of industry tools for an identified scenario | | |
| **P7** Implement a normalised relational database for an identified scenario using appropriate industry tools and languages.<br>**P8** Implement a test plan that thoroughly tests the relational database in the context of meeting the structural and behavioural requirements of the identified scenario. | **M4** Evaluate, using the results of the test plan, the effectiveness of the relational database in how well it meets the structural and behavioural requirements of the identified scenario. | |

## Recommended Resources

### Textbooks

Bagui, S.S. and Earp, R.W. (2022) *Database Design Using Entity–Relationship Diagrams*. 3rd edn. Boca Raton: CRC Press.

Connolly, T. and Begg, C. (2023) *Database Systems: A Practical Approach to Design, Implementation, and Management*. 7th edn. Harlow: Pearson Education.

Coronel, C. and Morris, S. (2024) *Database Systems: Design, Implementation, and Management*. 13th edn. Boston, MA: Cengage Learning.

Elmasri, R. and Navathe, S.B. (2016) *Fundamentals of Database Systems*. 7th edn. Harlow: Pearson.

Kleppmann, M. and Riccomini, C. (2026) *Designing Data-Intensive Applications*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Teorey, T.J. *et al.*. (2022) *Database Modeling and Design: Logical Design*. 6th edn. Burlington, MA: Morgan Kaufmann.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| hevodata.com/learn/enterprise-database-architecture/ | Hevo Data<br>A comprehensive guide to enterprise data architecture<br>(Guide) |
| www.geeksforgeeks.org/dbms/dbms-architecture-2-level-3-level/ | GeeksforGeeks<br>Types of DBMS Architecture<br>(Article) |
| www.datanamic.com/support/learn/articles/database-normalization.html | Datanamic<br>Database Normalization<br>(Tutorial) |
| www.digitalocean.com/community/tutorials/database-normalization | DigitalOcean<br>Database Normalization: 1NF, 2NF, 3NF & BCNF Examples<br>(Tutorial) |

 © Pearson Education Limited 2026

| | |
|---|---|
| dbschema.com/blog/design/design-a-relational-database-schema/ | DbSchema<br>How to design a relational database schema – step-by-step guide for 2026<br>(General reference) |
| www.lucidchart.com/pages/tutorial/database-design-and-structure | Lucidchart<br>Database Structure and Design Tutorial<br>(Tutorial) |
| www.geeksforgeeks.org/sql/sql-transactions/ | GeeksforGeeks<br>SQL Transactions<br>(Article) |
| learn.microsoft.com/en-us/azure/architecture/data-guide/relational-data/online-analytical-processing | Microsoft Learn<br>Online analytical processing<br>(Article) |
| deepdatawithmivaa.com/what-is-the-7th-normal-form/ | Deep Data with Mivaa<br>What Is the 7th normal form?<br>(Blog) |

### Journals and articles

Batini, C. and Scannapieco, M. (2016) 'Data and information quality: Dimensions, principles and techniques', *Data-Centric Systems and Applications.* Cham: Springer. Available at: doi.org/10.1007/978-3-319-24106-7.

Dimensions, Principles and Techniques, *Data-Centric Systems and Applications*. Springer. Available at: doi.org/10.1007/978-3-319-24106-7.

Leis, V. *et al*. (2015) 'How good are query optimizers, really?', *Proceedings of the VLDB Endowment*, 9(3), pp. 204–215. Available at: doi.org/10.14778/2850583.2850594.

Rajendran, R.K. and Priya, T.M. (2023) 'Designing an Efficient and Scalable Relational Database Schema: Principles of Design for Data Modeling', *Software Principles of Design for Data Modeling*. IGI Global. Available at: doi.org/10.4018/978-1-6684-9809-5.ch013.

Stonebraker, M., *et al*. (2022) 'The end of an architectural era: It's time for a complete rewrite', *Communications of the ACM*, 65(8), pp. 72–79. Available at: doi.org/10.1145/3226595.3226637.

### Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 403: Professional Practice*

*Unit 404: Digital Systems Analysis*

*Unit 405: Core Software Design and Development*

*Unit 411: Core Data Analytics.*

# Unit 411: Core Data Analytics

**Unit code:** **A/652/3263**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

Data analytics plays an increasingly important role in how organisations operate, make decisions and respond to change. Across public, private and third-sector environments, data is used to monitor performance, identify patterns, support planning and improve services. As a result, there is a growing need for individuals who can understand data, apply basic analytical techniques and interpret findings in a clear and meaningful way.

The aim of this unit is to provide students with a clear and practical introduction to the core principles of data analytics. It focuses on developing an understanding of how data is collected, prepared, analysed and reported within organisational contexts. The unit also introduces students to essential statistical concepts and highlights the importance of working within legislative, governance and ethical frameworks. Together, these elements support students in developing a responsible and structured approach to data analysis.

The unit covers four main areas of study. Students begin by exploring the stages of the data analysis life cycle and the methodologies that underpin effective analytical practice. The unit then introduces a range of statistical techniques used to summarise, compare and interpret data. Legislative, governance and ethical considerations are also examined, including data protection, privacy and organisational responsibility. The unit concludes by requiring students to apply appropriate analytical techniques to an identified dataset in order to generate evidence-based conclusions.

On successful completion of this unit, students will be able to analyse data using appropriate techniques, interpret results accurately and communicate findings that support decision-making. They will also understand the legal, ethical and governance responsibilities associated with working with data. These skills are directly relevant to entry-level and junior roles such as Junior Data Analyst, Business Analyst, Reporting Analyst, Operations Analyst and Data Assistant, and provide a foundation for progression to further study in data analytics or related digital disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Describe the principles and methodologies underpinning the data analytics life cycle

LO2 Explore the range of statistical techniques used to analyse and interpret data

LO3 Review the legislative and governance requirements applicable to data analytics

LO4 Apply appropriate analytical techniques to generate evidence-based conclusions from an identified dataset.

## Essential Content

LO1 **Describe the principles and methodologies underpinning the data analytics life cycle**

*Data analytics life cycle stages:*

Problem definition: problem definition e.g. defined issue, analytical question; analytical objectives e.g. analysis goals, insight targets; business context e.g. organisational setting, operational environment

Stakeholder context: stakeholder needs e.g. user expectations, decision requirements, information needs; requirements e.g. data requirements, reporting needs, analytical criteria; scope e.g. project boundaries, data limits, analysis coverage

Planning approach: planning e.g. analysis plan, task sequencing, activity scheduling; constraints e.g. time limits, data availability; success criteria e.g. measurable outcomes, acceptance standards; outcomes e.g. analytical outputs, decision inputs.

*Data acquisition and preparation:*

Data sourcing: data sources e.g. internal systems, external databases; primary data e.g. surveys, observations; secondary data e.g. reports, existing records

Data structure: structured data e.g. tables, spreadsheets, databases; unstructured data e.g. text files, emails, images

Data quality: data quality e.g. fitness for purpose, reliability; accuracy e.g. correctness, precision; completeness e.g. missing values, full records

Data preparation: cleaning e.g. error correction, removing duplicates; formatting e.g. standard units, consistent structures.

*Concepts of data analysis:*

Analytical focus: data summarisation e.g. totals, averages, counts; descriptive analysis e.g. basic statistics, summaries; pattern recognition e.g. repeated behaviour, regular trends

Insight identification: trend identification e.g. increases, decreases; anomalies e.g. outliers, unusual values; insights e.g. observations, conclusions

Data limitations: data limitations e.g. small samples, missing data; variability e.g. spread, fluctuation.

*Evaluation and reporting principles:*

Result validation: result checking e.g. verification, review; findings interpretation e.g. meaning of results, implications

Communication methods: communication e.g. written reports, presentations; clarity e.g. clear language, logical structure; visualisation e.g. charts, graphs

Decision support outputs: recommendations e.g. actions, priorities; decision support e.g. evidence, justification.

LO2 **Explore the range of statistical techniques used to analyse and interpret data**

*Descriptive statistical measures:*

Data overview: datasets e.g. collections, datasets, records; summaries e.g. totals, counts, grouped values; interpretation e.g. meaning, significance, relevance

Central tendency: central tendency e.g. typical value, average value, central value; mean e.g. arithmetic mean, calculated average, numerical average; median e.g. middle value, ordered midpoint, rank position; mode e.g. most frequent value, common value, repeated value

Data dispersion: range e.g. minimum value, maximum value; variance e.g. level of spread, data variation; standard deviation e.g. dispersion measure, variation indicator.

*Probability and distributions:*

Probability concepts: probability e.g. chance, likelihood; outcomes e.g. possible results, observed results

Distribution features: distributions e.g. value spread, frequency shape; frequencies e.g. counts, occurrences; proportions e.g. percentages, relative amounts

Normal distribution: normal distribution e.g. bell curve, standard distribution; shape e.g. symmetry, skewness; spread e.g. dispersion, variability.

*Relationship analysis techniques:*

Variable identification: variables e.g. measurable factors, data attributes; relationships e.g. connections, associations

Correlation analysis: correlation e.g. relationship strength, association level; direction e.g. positive relationship, negative relationship

Visual comparison: scatterplots e.g. plotted points, visual patterns; trends e.g. upward trend, downward trend; interpretation e.g. meaning, implications.

*Sampling and statistical awareness:*

Sampling concepts: sampling e.g. data selection, subset choice; populations e.g. full groups, target sets; samples e.g. selected data, representative data

Sampling quality: representativeness e.g. population reflection, coverage quality; bias e.g. selection bias, sampling bias; sample size e.g. number of observations, data volume

Statistical reliability: assumptions e.g. expected conditions, standard behaviour; limitations e.g. data restrictions, weaknesses; reliability e.g. consistency, dependability.

LO3 **Review the legislative and governance requirements applicable to data analytics**

*Data protection legislation:*

Legislative framework: data protection e.g. personal data handling, information protection; legislation e.g. legal requirements, regulatory rules; GDPR e.g. UK General Data Protection Regulation (GDPR), data regulation

Lawful processing: lawfulness e.g. legal basis, permitted use; fairness e.g. fair processing, reasonable use; transparency e.g. openness, clear communication

Key principles: purpose limitation e.g. defined use, restricted purpose; data minimisation e.g. limited collection, necessary data; accountability e.g. responsibility, record keeping.

*Data privacy and individual rights:*

Privacy principles: privacy e.g. personal privacy, data confidentiality; consent e.g. user agreement, permission choice; individual rights e.g. access rights, correction rights

Data control: data access e.g. subject access, information requests; data accuracy e.g. correct records, updates; data retention e.g. storage period, deletion timing

Responsibilities: personal data e.g. identifiable data, individual information; responsibilities e.g. organisational duties, staff roles.

*Governance and organisational responsibility:*

Governance structures: governance e.g. oversight, control frameworks; data ownership e.g. assigned ownership, data stewardship; accountability e.g. reporting, responsibility lines

Operational controls: policies e.g. internal rules, data policies; procedures e.g. working methods, handling steps; roles e.g. job functions, governance roles

Risk oversight: risk management e.g. risk identification, mitigation planning; compliance awareness e.g. staff awareness, training understanding.

*Ethical considerations in data analytics:*

Ethical principles: ethics e.g. moral standards, professional values; fairness e.g. impartial decisions, equal treatment; transparency e.g. openness, explainability

Bias and impact: bias e.g. data bias, selection bias; societal impact e.g. public effects, community outcomes; AI system bias: algorithmic bias e.g. training data bias, model bias; fairness in AI e.g. equitable outcomes, non-discriminatory predictions; mitigation strategies e.g. bias detection, model validation

Professional judgement: responsibility e.g. duty of care, accountable behaviour, evolving practice, knowledge currency; trust e.g. confidence, credibility; decision-making e.g. informed choices, ethical judgement.

*Security and data handling requirements:*

Security principles: data security e.g. information protection, secure systems; confidentiality e.g. restricted access, authorised use; integrity e.g. data accuracy, data consistency

Operational handling: access e.g. authorised access, permission control; storage e.g. secure storage, retention systems; safeguards e.g. technical controls, procedural controls

Risk prevention: awareness e.g. staff understanding, good practice; risk prevention e.g. threat reduction, incident avoidance.

LO4 **Apply appropriate analytical techniques to generate evidence-based conclusions from an identified dataset**

*Dataset characteristics:*

Dataset structure: dataset structure e.g. table layout, data organisation; variables e.g. measurable attributes, data fields; data types e.g. numerical data, categorical data; unstructured data e.g. text data, multimedia data; formats e.g. documents, images; sources e.g. social media, sensor outputs

Dataset context: scope e.g. data coverage, analysis range; size e.g. number of records, data volume; relevance e.g. suitability, usefulness

Data quality: quality e.g. reliability, fitness for purpose; completeness e.g. missing values, full records.

*Data preparation for analysis:*

Data cleaning: data cleaning e.g. error correction, duplicate removal; missing values e.g. gaps, incomplete records

Data organisation: organisation e.g. data ordering, categorisation; consistency e.g. standard values, aligned formats; accuracy e.g. correctness, validation

Data readiness: checks e.g. validation checks, data review; readiness e.g. usability, preparation state.

*Analytical technique application:*

Technique selection: techniques e.g. analytical methods, statistical tools, analysis approaches; selection e.g. method choice, suitability, relevance; alignment e.g. objective matching, task suitability, analytical fit

Technique use: application e.g. method use, analysis activity; calculations e.g. numerical operations, value computation; summaries e.g. totals, averages

Comparative analysis: comparisons e.g. group comparison, value comparison; relationships e.g. data connections, variable links; outputs e.g. results, findings.

*Insight development and interpretation:*

Pattern identification: patterns e.g. trends, regular behaviour; observations e.g. findings, detected changes

Insight interpretation: interpretation e.g. meaning, significance; context e.g. organisational setting, analysis purpose

Analytical limitations: limitations e.g. data constraints, reliability issues; explanations e.g. result reasoning, cause discussion.

*Evidence-based conclusions and recommendations:*

Conclusion development: conclusions e.g. summary findings, outcome statements; evidence e.g. analysed data, supporting results

Reasoned judgement: reasoning e.g. justification, logical explanation; confidence e.g. certainty level, assurance

Decision outputs: recommendations e.g. suggested actions, priorities; decisions e.g. informed choices, action decisions; outcomes e.g. organisational impact, next steps.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Describe the principles and methodologies underpinning the data analytics lifecycle</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Evaluate, using a selection of real-world examples, the effectiveness of the data analytics life cycle, statistical techniques and legislative requirements in organisational data analytics.</td></tr>
<tr><td>P1 Describe, using a selection of real-world examples, how data acquisition and preparation underpins the stages of the data analytics life cycle.<br><br>P2 Discuss, using a selection of real-world examples, the concepts and principles of data analysis, evaluation and reporting.</td><td>M1 Analyse, using a selection of real-world examples, the applications of the data analytics life cycle in a range of organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Explore the range of statistical techniques used to analyse and interpret data</td></tr>
<tr><td>P3 Explore, using a selection of real-world mathematical examples, the statistical methods and sampling techniques used to analyse and interpret data.<br><br>P4 Review, using a selection of real-world mathematical examples, the techniques of probability and relationship analysis used to analyse and interpret data.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of implementing statistical techniques to analyse and interpret data.</td></tr>
<tr><td colspan="2">LO3 Review the legislative and governance requirements applicable to data analytics</td></tr>
<tr><td>P5 Review, using a selection of real-world examples, the key legislative and governance requirements relevant to data analytics.<br><br>P6 Review, using a selection of real-world examples, the ethical and privacy requirements relevant to data analytics.</td><td>M3 Analyse, using a selection of real-world examples, the impact of legislative and governance requirements on ethical data analytics practices.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Apply appropriate analytical techniques to generate evidence-based conclusions from an identified dataset | | |
| **P7** Apply appropriate analytical techniques to generate evidence-based conclusions from an identified dataset.<br><br>**P8** Review the choice of analytical techniques and the evidence-based conclusions with a range of identified stakeholders to identify strengths and areas for improvements. | **M4** Justify the selection of the applied analytical techniques in supporting the evidence-based conclusions generated for the identified dataset. | **D2** Evaluate the extent to which the selected analytical techniques support valid and reliable evidence-based conclusions. |

## Recommended Resources

### Textbooks

Heumann, C., Schomaker, M. and Shalabh (2024) *Introduction to Statistics and Data Analysis: With Exercises, Solutions and Applications in R*. 2nd edn. Cham: Springer.

Peck, R. and Olsen, C. (2025) *Introduction to Statistics and Data Analysis*. 7th edn. Boston: Cengage.

Sharda, R., Delen, D. and Turban, E. (2024) *Business Intelligence, Analytics, Data Science, and AI*. 5th edn. Harlow: Pearson Education.

Sharda, R., Delen, D. and Turban, E. (2021) *Analytics, Data Science, and Artificial Intelligence: Systems for Decision Support*. 11th edn. Harlow: Pearson Education.

Wickham, H. and Grolemund, G. (2023) *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 2nd edn. Sebastopol, CA: O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.gov.uk/government/publications/the-government-data-quality-framework | Gov.UK<br>The Government Data Quality Framework<br>(Framework) |
| www.ibm.com/think/topics/data-life cycle-management | IBM Think<br>What is data life cycle management (DLM)?<br>(Article) |
| www.upgrad.com/blog/data-analytics-life cycle/ | The Data Analytics Lifecycle: A Complete Guide from Discovery to Decision-Making<br>(Article) |
| www.scribbr.com/statistics/descriptive-statistics/ | Descriptive Statistics \| Definitions, Types, Examples<br>(Reference) |
| statisticsbyjim.com/basics/correlation/ | Statistics By Jim<br>Correlation<br>(Article) |

 © Pearson Education Limited 2026

| | |
|---|---|
| ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/toolkit-for-organisations-considering-using-data-analytics/ | ICO<br>Toolkit for organisations considering using data analytics<br>(Toolkit) |
| www.datagovernance.com/the-data-governance-basics/adg-data-governance-basics/ | The Data Governance Institute<br>Data governance basics<br>(Article) |
| www.statology.org/cleaning-and-preparing-data-for-analysis-a-beginners-guide/ | Statology<br>Cleaning and Preparing Data for Analysis: A Beginner's Guide<br>(Article) |
| www.geeksforgeeks.org/data-analysis/data-analysis-tutorial/ | Geeks For Geeks<br>Data Analytics Tutorial<br>(Tutorial) |
| www.dataversity.net/articles/data-quality-dimensions/ | Dataversity<br>Data Quality Dimensions<br>(Article) |

**Journals and articles**

Han, C., Lu, X. and Zhang, P. (2023) 'Use of statistical methods in translation and interpreting research', *Target*, 35(4), pp. 483–513. Available at: doi.org/10.1075/target.21132.han.

Kaminski, M.E. and Malgieri, G. (2021) 'Algorithmic impact assessments under the GDPR: producing multi-layered explanations', *International Data Privacy Law*, 11(2), pp. 125–144. Available at: doi.org/10.1093/idpl/ipaa020.

Kim, J., Kim, D.H. and Kwak, S.G. (2024) 'Comprehensive guidelines for appropriate statistical analysis methods in research', *Korean Journal of Anesthesiology*, 77(5), pp. 503–517. Available at: doi.org/10.4097/kja.24016.

Szukits, Á. and Móricz, P. (2023) 'Towards data-driven decision making: the role of analytical culture and centralization efforts', *Review of Managerial Science*. Available at: doi.org/10.1007/s11846-023-00694-1.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 403: Professional Practice*

*Unit 404: Digital Systems Analysis*

*Unit 410: Database Engineering*

*Unit 412: Applied Data Analytics.*

 © Pearson Education Limited 2026

# Unit 412: Applied Data Analytics

**Unit code:** D/652/3264

**Unit level:** 4

**Credit value:** 15

---

## Introduction

The ability to collect, analyse and interpret data has become an essential requirement across many modern organisations. Businesses and public sector bodies increasingly rely on data to understand performance, improve services, identify trends and support evidence-based decision-making. Applied data analytics plays a key role in areas such as operations, customer insight, finance, logistics and service planning, enabling organisations to turn raw data into meaningful information.

The aim of this unit is to provide students with a practical introduction to applied data analytics within an enterprise context. It focuses on developing an understanding of how data is structured and organised, how datasets are prepared for analysis, and how analytical techniques can be applied to produce useful insights. The unit emphasises the responsible handling of data and the importance of clear communication of results to different audiences.

The unit covers a range of topics that reflect common data analytics activities found in the workplace. Students will explore different types of data and how they are stored and managed within enterprise systems. They will apply data preparation and cleansing techniques to improve data quality, before using analytical methods to identify patterns, trends and relationships. The unit also develops skills in presenting and communicating analytical findings using appropriate visual and written formats for a range of audiences.

On successful completion of this unit, students will be able to work confidently with enterprise data, apply basic analytical techniques, and communicate data-driven insights clearly and accurately. These skills are relevant to a variety of junior and entry-level roles, including Data Analyst, Business Analyst, Reporting Analyst, Operations Analyst, Junior Data Scientist, and Analytical Support Officer, as well as supporting progression to further study in data analytics or related Higher National units.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the different types of data and their organisation within an enterprise context

LO2 Apply appropriate data preparation and cleansing techniques on an identified dataset

LO3 Implement analytical methods to extract patterns and insights from the identified dataset

LO4 Develop a presentation to communicate the results of the analysis to a range of different audiences.

## Essential Content

LO1 **Explain the different types of data and their organisation within an enterprise context**

*Types of data:*

Data structure types: structured e.g. tables, rows; unstructured e.g. text documents, emails; semi structured e.g. JavaScript Object Notation (JSON) files, eXtensible Markup Language (XML) files

Data classification types: qualitative e.g. opinions, descriptions; quantitative e.g. measurements, counts; categorical e.g. labels, groups

Data value types: numerical e.g. integers, decimals; transactional e.g. sales records, purchases; time series e.g. daily totals, sensor readings

Operational contexts: operational e.g. system logs, process data; enterprise e.g. organisational records, business data.

*Data formats and representations:*

File-based formats: file formats e.g. digital files, stored data; flat files e.g. text files, comma separated values (CSV) files; CSV e.g. spreadsheets, exports; XML e.g. tagged data, structured text

Structured data formats: relational data structures e.g. tables, relationships; hierarchical data structures e.g. tree models, parent-child data; Structured Query Language (SQL) datasets e.g. database tables, queries

Data interchange formats: JSON e.g. web data, Application Programming Interface (API) responses; API-based data access e.g. web services, system connections.

*Data organisation and structure:*

Schema and definition elements: schemas e.g. database designs, structures; metadata e.g. data descriptions, properties; data dictionaries e.g. field definitions, attribute meanings

Database key concepts: primary keys e.g. unique identifiers, record IDs; foreign keys e.g. linked identifiers, references; identifiers e.g. codes, keys

Data modelling principles: normalisation principles e.g. reduced duplication, efficient design; relationships e.g. one-to-many links, associations; tables e.g. structured datasets, records

Data components: attributes e.g. column values, data fields; records e.g. rows, entries.

*Enterprise data environments:*

Data storage technologies: databases e.g. SQL systems, data stores; warehouses e.g. analytical storage, reporting systems; repositories e.g. data stores, archives

Technical platforms: systems e.g. enterprise software, information systems; platforms e.g. data platforms, analytics platforms; cloud e.g. online storage, hosted services

Enterprise applications: applications e.g. business software, analytics tools; services e.g. data services, processing services; organisations e.g. companies, public bodies.

LO2 **Apply appropriate data preparation and cleansing techniques on an identified dataset**

*Data preparation concepts:*

Data pipeline processes: extract e.g. source files, system exports; transform e.g. data changes, structural adjustments; load e.g. databases, target systems

Data acquisition activities: data ingestion e.g. data import, collection; data integration e.g. combined sources, merged datasets; sourcing e.g. internal data, external data

Dataset assessment activities: dataset validation e.g. rule checks, constraints; inspection e.g. data review, data checks; sampling e.g. subsets, test samples

Preparation context: enterprise data quality standards e.g. accuracy rules, completeness rules; preparation e.g. readiness steps, data setup; context e.g. business meaning, usage purpose.

*Data cleansing techniques:*

Data quality issues: errors e.g. incorrect values, invalid entries; omissions e.g. missing values, empty fields; duplicates e.g. repeated records, copied entries

Consistency problems: consistency e.g. uniform values, standard entries; formats e.g. date styles, number formats; alignment e.g. matching fields, aligned columns

Correction actions: accuracy e.g. corrected values, verified data; completeness e.g. filled fields, finished records; correction e.g. fixed errors, updated entries

Removal actions: removal e.g. deleted rows, excluded data.

*Data transformation methods:*

Value restructuring methods: conversion e.g. data type changes, unit changes; formatting e.g. date formats, text formats; standardisation e.g. common formats, shared values

Data summarisation techniques: aggregation e.g. totals, sums; summarisation e.g. summaries, overviews; grouping e.g. categories, grouped values

Data organisation techniques: categorisation e.g. labelled groups, classes; sorting e.g. ordered lists, ranked data; filtering e.g. selected records, relevant data

Structural changes: restructuring e.g. column changes, table changes.

*Tools and technologies for data preparation:*

General software tools: software e.g. data tools, applications; tools e.g. utilities, programs; spreadsheets e.g. Excel files, worksheets

Technical components: applications e.g. desktop software, analytics apps; scripts e.g. basic scripts, automation; functions e.g. formulas, built-in functions

Data processing resources: libraries e.g. data libraries, code libraries; platforms e.g. analytics platforms, data platforms; interfaces e.g. user interfaces, dashboards

Programming tools: utilities e.g. helpers, tools; Languages e.g. Python, R, SQL; Pandas e.g. dataframes, data manipulation.

LO3 **Implement analytical methods to extract patterns and insights from the identified dataset**

*Descriptive analytics methods:*

Basic calculation measures: counts e.g. item totals, occurrences; totals e.g. overall sums, aggregated values; averages e.g. mean values, typical values

Distribution measures: proportions e.g. percentage shares, ratios; frequencies e.g. occurrence counts, repetition rates; ranges e.g. minimum values, maximum values

Summary techniques: summaries e.g. overview statistics, data snapshots; distributions e.g. value spread, data spread; comparisons e.g. category comparisons, time comparisons.

*Exploratory data analysis techniques:*

Pattern identification: trends e.g. upward movement, downward movement; patterns e.g. repeated behaviour, regular changes; relationships e.g. linked values, associations

Variation analysis: variations e.g. fluctuations, data changes; groupings e.g. clustered values, categories; anomalies e.g. outliers, unusual values

Behavioural insights: correlations e.g. related changes, linked variables; contrasts e.g. differences, comparisons; observations e.g. visible features, noticed changes; behaviours e.g. user actions, system activity.

*Statistical and analytical techniques:*

Analytical modelling approaches: regression analysis e.g. trend modelling, prediction lines; clustering techniques e.g. grouped data, similarity clusters; classification concepts e.g. categories, labelled classes

Statistical principles: hypothesis testing principles e.g. assumptions, comparisons; calculations e.g. numeric analysis, computed values; comparisons e.g. dataset contrasts, result checks

Data processing logic: categorisation e.g. data classes, labels; segmentation e.g. divided groups, subsets; aggregation e.g. combined values, totals

Analytical reasoning: filtering e.g. selected data, relevant records; ranking e.g. ordered results, priority lists; interpretation e.g. meaning extraction, result explanation; evaluation e.g. outcome assessment, method review; reasoning e.g. logical conclusions, analytical judgement.

*Performance and insight generation:*

Measurement indicators: key performance indicators (KPIs) e.g. performance targets, success metrics; indicators e.g. performance signals, measures; metrics e.g. numeric measures, score values

Benchmarking concepts: benchmarks e.g. industry standards, targets; outputs e.g. analysis results, data products; findings e.g. discovered facts, observations

Decision support insights: insights e.g. meaningful patterns, conclusions; evidence e.g. supporting data, analysis results; conclusions e.g. final outcomes, summaries; decisions e.g. informed choices, actions; implications e.g. impacts, consequences.

LO4 **Develop a presentation to communicate the results of the analysis to a range of different audiences**

*Data visualisation methods:*

Visual display types: charts e.g. bar charts, line charts; graphs e.g. plotted data, trend lines; tables e.g. data grids, summary tables

Dashboard components: dashboards e.g. interactive screens, reporting views; visuals e.g. graphical elements, visual displays; summaries e.g. key figures, overview data

Design features: layouts e.g. page structure, arrangement; labels e.g. axis titles, data labels; scales e.g. measurement ranges, value scales; formats e.g. visual styles, presentation formats.

*Audience-focused communication:*

Audience groups: audiences e.g. target groups, intended users; stakeholders e.g. business stakeholders, project sponsors; users e.g. system users, data users

Organisational roles: managers e.g. team managers, department heads; teams e.g. project teams, work groups; specialists e.g. data specialists, technical staff

User experience levels: non-specialists e.g. general staff, non-technical users; clients e.g. external customers, service users; groups e.g. audience groups, user groups; roles e.g. job roles, responsibilities

Data storytelling: story structure e.g. narrative flow, logical progression; messaging e.g. key message, central insight; context e.g. background information, situational relevance; story elements: narrative e.g. data story, explanatory storyline; annotations e.g. captions, highlighted insights.

*Analytical reporting formats:*

Written report types: reports e.g. analysis reports, business reports; summaries e.g. executive summaries, brief reports; documents e.g. written outputs, digital files

Presentation materials: presentations e.g. slide decks, briefings; slides e.g. presentation pages, visuals; visuals e.g. charts, graphical content

Explanatory content: commentary e.g. explanations, descriptions; narratives e.g. data stories, explanations; overviews e.g. summary views, high-level analysis.

*Communication clarity and effectiveness:*

Quality characteristics: clarity e.g. understandable content, clear messages; accuracy e.g. correct data, precise values; relevance e.g. useful information, appropriate content

Structural features: consistency e.g. uniform style, standard format; structure e.g. logical order, organised sections; emphasis e.g. highlighted points, key messages

User interpretation: readability e.g. easy reading, clear text; interpretation e.g. meaning understanding, insight explanation; alignment e.g. matched content, consistent message; context e.g. business setting, audience needs

*Real-world examples:*

A range of real-world visualisation and reporting tools e.g.

Data visualisation tools: visualisation software e.g. Tableau, Microsoft Power BI; spreadsheet tools e.g. Microsoft Excel, Google Sheets; programming libraries e.g. Matplotlib, D3.js

Dashboard platforms: dashboard tools e.g. Power BI dashboards, Tableau dashboards; reporting platforms e.g. Google Data Studio (Looker Studio), Qlik Sense; enterprise solutions e.g. SAP Analytics Cloud

Reporting tools: reporting software e.g. Microsoft Word, Google Docs; presentation tools e.g. Microsoft PowerPoint, Google Slides; data reporting platforms e.g. Power BI reports, Tableau reports.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the different types of data and their organisation within an enterprise context</td><td rowspan="2"></td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the different types of data, formats and representation used within an enterprise context.<br><br>P2 Explore, using a selection of real-world examples, how data is structured and organised within identified enterprise data environments.</td><td>M1 Assess, using a selection of real-world examples, the benefits and limitations of the different ways that data is structured and organised within an enterprise context.</td></tr>
<tr><td colspan="2">LO2 Apply appropriate data preparation and cleansing techniques on an identified dataset</td><td rowspan="5">D1 Evaluate, using a selection of real world examples, the effectiveness of different data types and organisational approaches for enterprise data analysis.<br><br>LO2 and LO3<br><br>D2 Justify the decisions made in the selection of preparation, cleaning and analytical techniques in obtaining meaningful patterns, insights and conclusions from an identified dataset.</td></tr>
<tr><td>P3 Apply appropriate data preparation and cleansing techniques to an identified dataset.<br><br>P4 Review the techniques used for data preparation and cleaning with a group of identified stakeholders to identify strengths and areas for improvement.</td><td>M2 Evaluate the effectiveness of the applied data preparation and cleansing techniques in creating a dataset that is suitable for analysis.</td></tr>
<tr><td colspan="2">LO3 Implement analytical methods to extract patterns and insights from the identified dataset</td></tr>
<tr><td>P5 Implement analytical methods to extract patterns and insights from the identified dataset.<br><br>P6 Review the patterns and insights extracted from the identified dataset with a group of identified stakeholders to identify strengths and limitations and to draw meaningful conclusions.</td><td>M3 Evaluate the effectiveness of the implemented analytical methods in generating meaningful conclusions from the identified dataset.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Develop a presentation to communicate the results of the analysis to a range of different audiences | | |
| **P7** Develop a presentation to communicate the results of the analysis to a range of different audiences.<br><br>**P8** Present the results of the analysis to a range of different audiences to obtain meaningful feedback. | **M4** Analyse, using the audience feedback results, how effectively the analytical results have been communicated to different audiences. | **D3** Evaluate the suitability of the communication approaches used for different audiences in presenting analytical results. |

## Recommended Resources

### Textbooks

Bruce, P., Bruce, A. and Gedeck, P. (2024) *Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python*. 2nd edn. Sebastopol, CA: O'Reilly Media.

James, G. *et al*. (2021) *An Introduction to Statistical Learning*. 2nd edn. Cham: Springer.

Maheshwari, A. (2023) *Data Analytics Made Accessible*. McGraw-Hill Education.

Provost, F. and Fawcett, T. (2023) *Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking*. 2nd edn. O'Reilly Media.

Richards, G. and Yeoh, W. (2022) *Business Information Systems*. 3rd edn. Routledge.

Szasz, B. (2024) *Data Analytics Essentials You Always Wanted to Know*. Vibrant Publishers.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/data-analysis/data-analysis-tutorial/ | GeeksforGeeks<br>Data Analytics Tutorial<br>(General reference) |
| www.ibm.com/think/topics/data-warehouse | IBM<br>What is a data warehouse?<br>(Article) |
| www.datacamp.com/tutorial/tutorial-data-cleaning-tutorial | DataCamp<br>Data Cleaning: Understanding the Essentials<br>(Tutorial) |
| www.udacity.com/blog/2025/11/data-cleaning-techniques-how-to-prepare-clean-data-for-analysis.html | Udacity<br>Data Cleaning Techniques: How to Prepare Clean Data for Analysis<br>(Article) |
| towardsdatascience.com/a-data-scientists-essential-guide-to-exploratory-data-analysis-25637eee0cf6/ | Towards Data Science<br>A Data Scientist's Essential Guide to Exploratory Data Analysis<br>(Article) |

| | |
|---|---|
| indatalabs.com/blog/exploratory-data-analysis | InData Labs<br>Exploratory data analysis explained<br>(Article) |
| online.csp.edu/resources/article/data-visualization-techniques-how-to-communicate-insights-effectively/ | CSP Global<br>Data Visualisation Techniques: How to Communicate Insights Effectively<br>(Article) |
| www.datacamp.com/blog/data-visualization-techniques | DataCamp<br>11 Data Visualization Techniques for Every Use-Case with Examples<br>(Blog) |
| www.pragmaticinstitute.com/resources/articles/data/comprehensive-guide-how-to-communicate-data-insights-to-business-stakeholders/ | Pragmatic Institute<br>Comprehensive Guide: How to Communicate Data Insights to Business Stakeholders<br>(Guide) |
| www.geeksforgeeks.org/dbms/data-warehousing-tutorial/ | Geeks for Geeks<br>Data Warehousing Tutorial<br>(Tutorial) |

## Journals and articles

Fernandes, A. *et al*. (2023) 'Data Preparation: A Technological Perspective and Review', *SN Computer Science*, 4, Article 425. Available at: doi.org/10.1007/s42979-023-01828-8.

Koukaras, P. and Tjortjis, C. (2025) 'Data Preprocessing and Feature Engineering for Data Mining: Techniques, Tools, and Best Practices', *AI*, 6(10), 257. Available at: doi.org/10.3390/ai6100257.

Ouyang, W. (2024) 'Data visualization in big data analysis: applications and future trends', *Journal of Computer and Communications*, 12(11), pp. 76–85. doi.org/10.4236/jcc.2024.1211005.

Shu, L. and Bi, C. (2023) 'A study on the design and implementation plan of computer network architecture in education and teaching', *Contemporary Education and Teaching Research*, 4(7), pp. 300–304. Available at: doi.org/10.61360/BoniCETR232012920703.

## Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 403: Professional Practice*

*Unit 404: Digital Systems Analysis*

*Unit 410: Database Engineering*

*Unit 411: Core Data Analytics.*

# Unit 413: Core Networking

**Unit code:** **F/652/3265**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

Computer networks are central to the way modern organisations operate, enabling communication, data sharing and access to digital services. From small offices and educational settings to large enterprises and cloud-based systems, reliable networks support everyday business activity and collaboration. Understanding how networks are structured and how devices communicate is therefore a key skill for anyone entering the IT and digital sector.

The aim of this unit is to provide students with a clear introduction to the core principles of computer networking. It focuses on how networks are designed, how hardware and software components function together, and how basic network solutions are planned and developed to meet identified needs. The unit builds practical understanding alongside essential theory that is transferable across different technologies and platforms.

The unit covers fundamental networking concepts, including network architectures, topologies, layered models, protocols and addressing. Students explore the roles of common network hardware and software, before applying this knowledge to design a basic network for a given scenario. The unit then moves on to the development and testing of that network, ensuring it operates as intended and meets user and organisational requirements.

On successful completion of this unit, students will be able to explain key networking concepts, design simple network solutions, and develop and review basic network implementations. These skills support progression into roles such as IT Support Technician, Junior Network Administrator, Network Support Technician, Infrastructure Technician, or Systems Support Analyst, as well as further study in networking, cybersecurity and infrastructure-related units.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Define the key principles of network architectures

LO2 Explain the functions of network hardware and software

LO3 Design a basic network for an identified scenario

LO4 Develop a basic network for an identified scenario.

## Essential Content

### LO1 Define the key principles of network architectures

*Network architecture models:*

Architecture types: client–server architecture e.g. file servers, application servers, authentication services; peer-to-peer architecture e.g. workstation sharing, local collaboration; hybrid architectures e.g. cloud integration, mixed environments

System structure: centralised systems e.g. data centres, directory services; distributed systems e.g. service replication, load distribution, geographic separation

Evaluation aspects: advantages e.g. performance, manageability, scalability; limitations e.g. cost, complexity; use cases e.g. small businesses, educational networks.

*Network topologies:*

Topology types: star topology e.g. switches, central nodes; mesh topology e.g. redundancy, fault tolerance, wireless links; hybrid topology e.g. enterprise layouts, campus networks

Topology perspectives: physical topology e.g. cabling layout, device placement, hardware paths; logical topology e.g. data flow, traffic paths

Design considerations: resilience considerations e.g. redundancy, availability, fault tolerance; scalability implications e.g. expansion, capacity planning.

*Layered network models:*

Design concepts: abstraction e.g. functional separation, simplified design; layered design e.g. hierarchical structure, protocol organisation; role of layers e.g. interoperability, troubleshooting

Reference models: Open Systems Interconnection (OSI) model e.g. seven layers, standardisation; Transmission Control Protocol/Internet Protocol (TCP/IP) model e.g. four layers, Internet communication, practical implementation

Data handling: data encapsulation e.g. headers, trailers, packet structure; layer communication e.g. peer processes, service interfaces.

*Protocols and addressing principles:*

Protocol structure: protocols e.g. communication rules, data formats; protocol stacks e.g. layered protocols, dependency ordering, system implementation

Addressing concepts: Internet Protocol (IP) addressing e.g. logical addressing, network identification, routing support; IPv4 concepts e.g. 32-bit addressing, dotted decimal, address classes; IPv6 concepts e.g. 128-bit addressing, hexadecimal notation, address exhaustion solution; MAC addressing e.g. hardware identifiers, network interfaces, local delivery

Communication mechanisms: ports e.g. service identification, application mapping, endpoint control; unicast addressing e.g. one-to-one delivery, targeted traffic; broadcast addressing e.g. discovery messages, network announcements.

LO2 **Explain the functions of network hardware and software**

*Network hardware components:*

Core devices: routers e.g. traffic forwarding, network routing, path selection; switches e.g. frame forwarding, Media Access Control (MAC) filtering, Local Area Network (LAN) segmentation; wireless access points e.g. wireless connectivity, signal distribution, device access

Interface components: network interface cards e.g. data transmission, device connectivity; cabling types e.g. copper cabling, fibre cabling, network links

Protection and connectivity: firewalls e.g. traffic filtering, access control; physical media characteristics e.g. bandwidth, distance limits.

*Network software and operating systems:*

Operating platforms: operating systems e.g. Windows, Linux, macOS; security features e.g. user permissions, authentication, access control, file systems; network operating systems e.g. user management, resource control, network services; device firmware e.g. hardware control, system updates

Support software: network services e.g. file sharing, authentication; configuration utilities e.g. device setup, parameter configuration; management interfaces e.g. dashboards, command interfaces

Virtual environments: virtualisation concepts e.g. virtual machines, resource abstraction.

*Core network services and protocols:*

Transmission services: TCP/IP suite e.g. reliable delivery, packet handling; Hypertext Transfer Protocol (HTTP) e.g. web communication, data exchange; User Datagram Protocol (UDP) e.g. fast delivery, streaming data, low latency; File Transfer Protocol (FTP) e.g. file transfer, data movement

Naming and configuration: Domain Name System (DNS) e.g. name resolution, domain lookup; Dynamic Host Configuration Protocol (DHCP) e.g. address assignment, network configuration

Interaction principles: client–server interaction e.g. request-response, service access; data transmission principles e.g. packet delivery, flow control.

*Interaction between hardware and software:*

Packet handling: packet forwarding e.g. data movement, interface selection; routing decisions e.g. path calculation, destination selection; switching processes e.g. frame handling, port forwarding

Address resolution: address resolution e.g. IP mapping, MAC identification

System dependency: protocol operation e.g. rule enforcement, data handling; hardware capabilities e.g. processing speed, interface support; software configuration e.g. parameter settings, feature control.

LO3 **Design a basic network for an identified scenario**

*Network requirements and constraints:*

User needs: user requirements e.g. number of users, access needs; functional requirements e.g. file access, internet connectivity; performance needs e.g. speed expectations, response time

Operational limits: security considerations e.g. access control, data protection; budgetary constraints e.g. cost limits, hardware choices; physical environment limitations e.g. building layout, space availability.

*Logical network design:*

Addressing and structure: IP addressing schemes e.g. address allocation, network organisation; subnetting concepts e.g. address division, traffic management; network segmentation e.g. virtual local area networks (VLAN) separation, traffic isolation

Network behaviour: device roles e.g. routers, switches; traffic flow e.g. data paths, communication routes; logical diagrams e.g. network layout, logical structure.

*Physical network design:*

Infrastructure placement: device placement e.g. rack location, endpoint positioning; cabling layout e.g. cable paths, outlet locations; wireless coverage e.g. signal range, access point placement

Deployment choices: hardware selection e.g. device type, capability matching; physical security considerations e.g. equipment protection, restricted access.

*Design documentation:*

Design records: network diagrams e.g. physical diagrams, logical diagrams; addressing tables e.g. IP listings, device mapping

Design justification: design rationale e.g. decision explanation, design choices; assumptions and limitations e.g. constraints, design boundaries; alignment with requirements e.g. requirement mapping, scenario fit.

LO4 **Develop a basic network for an identified scenario**

*Network implementation concepts:*

Device configuration principles: device configuration principles e.g. interface setup, parameter settings; basic routing configuration e.g. static routes, default routes; switching configuration e.g. VLAN settings, port assignment; dynamic routing protocols e.g. Open Shortest Path First (OSPF), Routing Information Protocol (RIP) for simple route updates; route propagation e.g. automatic updates, route sharing across interconnected subnets

Addressing implementation: addressing implementation e.g. IP assignment, subnet allocation; connectivity validation e.g. link testing, device reachability.

*Development alignment with design:*

Design consistency: consistency between logical design and physical implementation e.g. design accuracy, specification matching; implementation constraints e.g. hardware limits, time constraints; configuration accuracy e.g. correct settings, error reduction.

*Network functionality and operation:*

Operational behaviour: data transmission e.g. packet delivery, data flow; device communication e.g. host connectivity, inter-device exchange; service availability e.g. uptime, accessibility

Problem handling: fault identification e.g. connectivity issues, misconfiguration; basic troubleshooting concepts e.g. testing tools, systematic diagnosis

Failure causes: hardware faults e.g. device failure, cabling issues; software faults e.g. misconfiguration, firmware errors; external factors e.g. power loss, environmental conditions

Failure consequences: service disruption e.g. loss of connectivity, reduced availability; performance degradation e.g. latency increase, throughput reduction; data impact e.g. data loss, data corruption

Operational impact: business disruption e.g. halted operations, reduced productivity; user impact e.g. loss of access, service unavailability; recovery considerations e.g. restoration processes, system resilience.

*Testing and verification:*

Verification activities: connectivity testing e.g. ping tests, link checks; configuration verification e.g. setting review, command output; functional validation e.g. user access, service operation

Performance observation: performance observation e.g. response time, throughput; stability considerations e.g. consistent operation, error prevention.

*Development constraints and limitations:*

Resource boundaries: resource limitations e.g. hardware availability, budget limits; configuration boundaries e.g. feature restrictions, device capability

Future impact: performance constraints e.g. bandwidth limits, processing overhead; scalability considerations e.g. network growth, expansion potential; implementation risks e.g. misconfiguration, design deviation.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Define the key principles of network architectures</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a selection of real world examples, the effectiveness of different network architecture principles, hardware and software in supporting reliable and scalable high-performance networks.</td></tr>
<tr><td>P1 Define, using a selection of real-world examples, the key principles of network architectures, models and topologies.<br><br>P2 Explain, using a selection of real-world examples, the key principles of network protocols and addressing.</td><td>M1 Analyse, using a selection of real-world examples, the benefits and limitations of key network principles in the design of modern networked systems.</td></tr>
<tr><td colspan="2">LO2 Explain the functions of network hardware and software</td></tr>
<tr><td>P3 Explain, using a selection of real-world examples, the functions of network hardware components within a networked environment.<br><br>P4 Describe, using a selection of real-world examples, the functions of network software and services and their interaction with hardware components.</td><td>M2 Assess, using a selection of real-world examples, the implications of different hardware and software choices and interactions in the performance of modern networked systems.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a basic network for an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of the network in how effectively it meets the requirements of the identified scenario, considering user needs and operational constraints. |
| **P5** Design a basic network for an identified scenario, addressing requirements, constraints, and logical and physical design considerations.<br>**P6** Review the design documentation with a range of internal stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the stakeholder feedback, the network design so that it comprehensively addresses the requirements of the identified scenario. | |
| **LO4** Develop a basic network for an identified scenario | | |
| **P7** Develop a basic network for an identified scenario in line with the proposed network design.<br>**P8** Implement a test plan to thoroughly test the performance of the network in meeting the requirements of the identified scenario. | **M4** Evaluate, using the results of the test plan, the effectiveness of the developed network in how well it matches the design and meets the requirements of the identified scenario. | |

## Recommended Resources

### Textbooks

Beasley, J.S. and Nilkaew, P. (2021) *Networking Essentials*. 6th edn. Indianapolis: Pearson IT Certification.

Forouzan, B.A. (2022) *Data Communications and Networking with TCP/IP Protocol Suite*. 6th edn. New York: McGraw Hill.

Kurose, J.F. and Ross, K.W. (2023) *Computer Networking: A Top-Down Approach*. 9th edn. Harlow: Pearson.

Panek, C. (2023) *Networking Fundamentals*. Indianapolis: Wiley.

Peterson, L.L. and Davie, B.S. (2024) *Computer Networks: A Systems Approach*. 6th edn. Burlington, MA: Morgan Kaufmann.

Tanenbaum, A.S., Feamster, N. and Wetherall, D.J. (2021) *Computer Networks*. 6th edn. Harlow: Pearson.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.tutorialspoint.com/data_communication_computer_network/osi_model.htm | TutorialsPoint<br>Explanation of the OSI model and the role of each layer in network communication.<br>(General reference) |
| www.geeksforgeeks.org/computer-networks/difference-between-osi-model-and-tcp-ip-model/ | GeeksforGeeks<br>Comparison of the OSI and TCP/IP models showing how networking layers are structured and mapped.<br>(General reference) |
| www.ibm.com/think/topics/network-topology | IBM Think<br>Explanation of physical and logical network topologies and their impact on network design.<br>(General reference) |

| | |
|---|---|
| www.computernetworkingnotes.com/networking-tutorials/network-topologies-explained-with-examples.html | Computer Networking Notes<br>Tutorial explaining common network topologies with examples, advantages and limitations.<br>(Tutorial) |
| www.geeksforgeeks.org/computer-networks/network-troubleshooting-techniques/ | GeeksforGeeks<br>Overview of basic network troubleshooting techniques including ping and traceroute.<br>(General reference) |
| www.ibm.com/docs/en/zos/2.1.0?topic=network-topology | IBM Documentation<br>Reference documentation defining physical and logical network topology concepts.<br>(General reference) |
| www.geeksforgeeks.org/computer-networks/network-devices-hub-repeater-bridge-switch-router-gateways/ | GeeksforGeeks<br>Explanation of common network hardware devices including switches, routers, access points and gateways.<br>(General reference) |
| www.tutorialspoint.com/article/Basic-Network-Hardware | TutorialsPoint<br>Overview of basic network hardware components and their roles within a network.<br>(General reference) |
| www.geeksforgeeks.org/operating-systems/what-is-a-network-operating-system/ | GeeksforGeeks<br>Explanation of network operating systems and how they manage network resources.<br>(General reference) |
| www.computernetworkingnotes.com/networking-tutorials/network-operating-system-features-and-functions.html | Computer Networking Notes<br>Detailed overview of network operating system features and services.<br>(General reference) |

## Journals and articles

Ariyanto, Y., Harijanto, B. and Watequlis, Y. (2019) 'Implementation Basic Network Design with Netkit For Evaluation of Network Learning', *International Journal of Online Engineering*, 15(8), pp. 18–30. Available at: doi.org/10.3991/ijoe.v15i08.9795.

Cornelissen, F.C., McLellan, R. and Daly, A.J. (2024) 'Developing research networks in schools: the role of network intentionality and research engagement', *Frontiers in Education*, 9. Available at: doi.org/10.3389/feduc.2024.1413128.

Makwana, A.H. and Atkotiya, K.H. (2023) 'A Literature Review: Extensible Network Model Architecture', *International Journal of Innovative Science and Research Technology*, 8(7). Available at: doi.org/10.5281/zenodo.8186566.

Shu, L. and Bi, C. (2023) 'A Study on the Design and Implementation Plan of Computer Network Architecture in Education and Teaching', *Contemporary Education and Teaching Research,* 4(7), pp. 300–304. Available at: doi.org/10.61360/BoniCETR232012920703.

## Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 409: Core Cybersecurity*

*Unit 414: Applied Networking*

*Unit 416: Core Cloud Computing.*

# Unit 414: Applied Networking

**Unit code:** H/652/3266

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Modern organisations rely on robust, secure and well-managed networks to support everyday operations, digital services and strategic decision-making. As business environments increasingly use cloud platforms, remote access and interconnected systems, enterprise networking plays a central role in ensuring availability, performance and security across a wide range of sectors. This unit introduces students to applied enterprise networking, focusing on how networks are monitored, supported, designed and implemented in real organisational contexts.

The aim of this unit is to develop students' understanding of how enterprise networks operate in practice and how they are kept reliable, secure and fit for purpose. Students will build knowledge of key monitoring and diagnostic approaches, explore the role of essential enterprise network services, and apply secure design principles to meet organisational requirements. The unit is designed to reflect current industry practice and support the development of practical, job-relevant networking skills.

The unit covers four closely connected areas. Students begin by examining tools and techniques used to monitor and diagnose enterprise networks, including performance and fault identification. They then explore the purpose and operation of core enterprise network services that support availability, performance and security. Building on this knowledge, students design a secure enterprise network for a given scenario and, finally, implement and evaluate a secure solution, considering testing, performance and enterprise suitability.

On successful completion of this unit, students will have the skills needed to support and contribute to enterprise network environments in a professional setting. They will be able to analyse network performance, support enterprise services, design secure network architectures and evaluate implemented solutions. These skills support progression into roles such as Network Technician, Junior Network Engineer, Infrastructure Support Specialist, Network Operations Technician or Cybersecurity Technician, and provide a strong foundation for further study in networking and information security.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore the tools and techniques for the monitoring and diagnostics of an enterprise network

LO2 Discuss the range of enterprise network services

LO3 Design a secure network for a given enterprise scenario

LO4 Implement a secure network for a given enterprise scenario.

## Essential Content

LO1 **Explore the tools and techniques for the monitoring and diagnostics of an enterprise network**

*Enterprise network monitoring principles:*

Baseline performance measures: baseline performance metrics e.g. normal operating values, network benchmarks; key performance indicators e.g. service thresholds, performance targets

Performance indicators: latency e.g. transmission delay, response time; throughput e.g. data transfer rates, bandwidth utilisation; jitter e.g. packet delay variation, timing inconsistency

Availability and fault management: availability monitoring e.g. uptime tracking, service status checks; fault detection e.g. error identification, alert triggers; incident triage processes e.g. incident logging, fault prioritisation

Analysis and reporting: root-cause analysis concepts e.g. fault causation, problem sources; operational documentation e.g. configuration records, incident records; reporting e.g. performance reports, incident summaries.

*Monitoring and diagnostic tools:*

Management and control protocols: Simple Network Management Protocol (SNMP) e.g. device status polling, network management; SNMPv2 e.g. enhanced security features, improved performance; SNMPv3 e.g. authentication, encryption

Traffic and flow monitoring: flow-based monitoring technologies e.g. traffic flow visibility, usage analysis; NetFlow e.g. traffic measurement, flow records; Internet Protocol Flow Information Export (IPFIX) e.g. standardised flow export, traffic statistics

Logging and packet visibility: Syslog services e.g. event logging, system alerts; Switched Port Analyser (SPAN) e.g. traffic duplication, monitoring ports; port mirroring techniques e.g. packet capture, traffic inspection

Monitoring platforms and alerts: network performance monitoring platforms e.g. monitoring dashboards, performance views; alerting mechanisms e.g. threshold alerts, notifications; logging mechanisms e.g. log storage, event records

Network automation approaches: automation techniques e.g. script-based configuration, orchestration tools; artificial intelligence applications e.g. anomaly detection, predictive analytics; automated monitoring and response e.g. alert correlation, self-healing activities; efficiency improvements e.g. reduced manual intervention, consistent configuration.

*Enterprise diagnostics techniques:*

Traffic and performance evaluation: traffic analysis e.g. packet inspection, bandwidth usage; performance trending e.g. historical analysis, capacity planning; diagnostic data e.g. performance logs, monitoring outputs

Fault identification processes: fault isolation methods e.g. component testing, problem narrowing; correlation of monitoring events e.g. event linking, fault association; escalation workflows e.g. support escalation, incident handling

Enterprise context interpretation: interpretation e.g. data interpretation, result analysis; enterprise network environments e.g. organisational networks, business infrastructures; monitoring events e.g. alerts, system events.

LO2 **Discuss the range of enterprise network services**

*Core enterprise network services:*

Address allocation services: Dynamic Host Configuration Protocol (DHCP) e.g. automatic address assignment, lease management; address allocation reliability, network host configuration

Name resolution services: Domain Name System (DNS) e.g. hostname resolution, domain lookup; name resolution reliability, service continuity

Time synchronisation services: Network Time Protocol (NTP) e.g. clock synchronisation, time accuracy; time consistency, log correlation

Identity and access services: Authentication, Authorisation and Accounting (AAA) e.g. user authentication, access control; identity verification, usage tracking

Directory integration services: directory services integration e.g. centralised user management, credential storage; policy enforcement, identity consistency.

*Service configuration and operation concepts:*

Address and naming management: address management e.g. IP address allocation, subnet control; name resolution e.g. hostname mapping, query handling

Time and identity co-ordination: time synchronisation e.g. device time alignment, clock accuracy; identity management e.g. user identities, access credentials

Service relationship awareness: service dependency relationships e.g. service interconnection, operational reliance; service interaction impact, availability influence

Resilience and availability concepts: service resilience e.g. fault tolerance, service continuity; availability considerations e.g. uptime planning, service reliability.

*Enterprise service support functions:*

Traffic prioritisation mechanisms: Quality of Service (QoS) classifications e.g. traffic classes, service priorities; marking e.g. packet marking, traffic identifiers

Availability and continuity methods: high availability concepts e.g. failover support, redundancy planning; service redundancy e.g. backup services, duplicate components

Performance enhancement approaches: performance optimisation e.g. traffic efficiency, response improvement; capacity usage, throughput improvement

Security support services: security support provided by enterprise network services e.g. access enforcement, service protection; threat mitigation, controlled access.

*Enterprise network service impact:*

Availability outcomes: availability enablement e.g. service uptime, operational continuity; service dependency effects e.g. cascading failures, reliance awareness

Performance outcomes: performance optimisation e.g. efficient data flow, reduced latency; service quality improvement, responsiveness

Security outcomes: security enforcement support e.g. access control enforcement, policy application; risk reduction, protection consistency

Organisational contribution: contribution to organisational network effectiveness e.g. operational efficiency, business support; service reliability, enterprise performance.

LO3 **Design a secure network for a given enterprise scenario**

*Enterprise network requirements:*

Organisational requirements: business requirements e.g. operational objectives, service needs; technical constraints e.g. infrastructure limitations, device capability; scalability considerations e.g. future growth, increased demand

Governance and risk awareness: compliance and regulatory awareness e.g. data protection requirements, organisational standards; risk considerations e.g. threat exposure, business impact

Enterprise environment context: enterprise environments e.g. organisational networks, business infrastructures; requirement alignment e.g. stakeholder needs, operational expectation

Network design considerations: critical assets e.g. network devices, data resources; security considerations e.g. access control, threat protection; accessibility e.g. user access, remote connectivity; reliability e.g. uptime requirements, fault tolerance; performance considerations e.g. bandwidth requirements, response times

Service level and customer requirements: Service level agreements (SLAs) e.g. response times, uptime targets; contractual obligations e.g. service delivery expectations, compliance terms; service performance monitoring e.g. service metrics, performance tracking; customer service considerations e.g. communication practices, user support.

*Secure network architecture concepts:*

Defence structures: layered security models e.g. defence in depth, security layering; trust boundaries e.g. access zones, security separation; trust models e.g. continuous verification, least privilege access, identity-based security; architectural approaches e.g. zero-trust architecture, perimeter-based security

Segmentation approaches: network segmentation e.g. traffic separation, security isolation; virtual local area networks (VLANs) e.g. logical grouping, broadcast domain control

Traffic control and perimeter design: routing domains e.g. internal routing, path management; perimeter security e.g. boundary protection, edge control; internal security zones e.g. protected areas, restricted segments

Logical architecture planning: logical network design e.g. diagrammatic layouts, functional grouping; data flow design e.g. traffic paths, communication flows.

*Security design components:*

Access and policy controls: firewall policy principles e.g. traffic filtering, rule ordering; secure routing and switching concepts e.g. hardened devices, controlled forwarding; secure protocols e.g. encrypted communication, secure transport

Addressing and structure: IP addressing schemes e.g. Classless Inter-Domain Routing (CIDR), Variable Length Subnet Masking (VLSM); addressing structure e.g. efficient allocation, logical organisation

Secure topology: network zones e.g. internal zone, external zone; demilitarised zone e.g. demilitarised zones (DMZ) placement, intermediary network segment; topology structure
e.g. layered topology, segmented network layout

Service placement decisions: security service placement e.g. firewall positioning, inspection points; network service integration e.g. aligned services, secure interaction

Continuity planning: redundancy and resilience planning e.g. failover design, fault tolerance; service continuity considerations e.g. availability planning, operational reliability.

LO4 **Implement a secure network for a given enterprise scenario**

*Secure network implementation concepts:*

Device configuration activities: network device configuration principles e.g. baseline configuration, secure defaults; application of security controls e.g. access rules, protective settings

Network segmentation deployment: implementation of VLANs e.g. logical separation, traffic control; routing e.g. path selection, inter-VLAN communication

Security service deployment: deployment of firewall rules e.g. traffic filtering, rule enforcement; secure network services e.g. protected services, controlled access

Addressing and connectivity setup: IP addressing implementation e.g. address assignment, subnet application; connectivity establishment e.g. interface configuration, link activation

Identity and access management (IAM): authentication methods e.g. multi-factor authentication (MFA), user verification; access control models e.g. role-based access control (RBAC), least privilege access control (LPAC); identity management e.g. user identities, account management.

*Testing and verification:*

Connectivity confirmation: connectivity verification e.g. link testing, end-to-end checks; functional testing e.g. service operation, expected behaviour

Security assurance checks: security validation e.g. access verification, policy enforcement; identification of configuration issues e.g. misconfiguration detection, error discovery

Performance confirmation: performance confirmation e.g. throughput checks, response consistency; operational readiness e.g. stable operation, acceptable performance.

*Operational considerations:*

Security outcomes: security enforcement outcomes e.g. effective access control, policy compliance; functional reliability e.g. dependable operation, reduced faults

Performance stability: performance stability e.g. consistent response, predictable behaviour; enterprise suitability e.g. organisational fit, operational use

Alignment with requirements: alignment with organisational requirements e.g. business needs, user expectations; operational environments e.g. live networks, enterprise contexts

Configuration documentation: configuration records e.g. device settings, security rules; implementation notes e.g. deployment steps, applied changes

Organisational policies and procedures: information management policies e.g. data handling procedures, information classification; recording accuracy requirements e.g. data integrity, validation processes; management controls e.g. access permissions, audit trails; compliance procedures e.g. regulatory adherence, policy enforcement

Operational handover: operational environments e.g. live networks, enterprise contexts; support readiness e.g. administrative access, monitoring awareness; SLA acceptance e.g. stakeholder sign-off, functional approval.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explore the tools and techniques for the monitoring and diagnostics of an enterprise network</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a selection of real-world examples the contribution of network monitoring and diagnostic tools and techniques and network services to the overall effectiveness of an organisational network.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the principles of monitoring an enterprise network.<br><br>P2 Explore, using a selection of real-world examples, the tools and techniques used for monitoring and diagnostics of an enterprise network.</td><td>M1 Analyse, using a selection of real-world examples, the effectiveness of monitoring and diagnostic tools and techniques in supporting enterprise network performance and reliability.</td></tr>
<tr><td colspan="2">LO2 Discuss the range of enterprise network services</td></tr>
<tr><td>P3 Discuss, using a selection of real-world examples, a range of network services, configurations and support functions available to an enterprise network.<br><br>P4 Explore, using a selection of real-world examples, the impact of these network services on the availability, performance and security of an enterprise network.</td><td>M2 Analyse, using a selection of real-world examples, the benefits and limitations of network services in supporting the availability, performance and security of an organisational network.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a secure network for a given enterprise scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of the secure network in how effectively it meets the requirements of the identified enterprise scenario. |
| **P5** Design a secure network solution that meets the requirements of a given enterprise scenario.<br>**P6** Review the design documentation of the secure network with a range of internal stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the stakeholder feedback, the design of the secure network so that it comprehensively addresses the requirements of the identified enterprise scenario. | |
| **LO4** Implement a secure network for a given enterprise scenario | | |
| **P7** Implement a secure network solution for a given enterprise scenario in line with the proposed design.<br>**P8** Implement a test plan to thoroughly test the performance of the secure network in meeting the requirements of the identified enterprise scenario. | **M4** Evaluate, using the results of the test plan, the effectiveness of the developed network in how well it matches the design and meets the requirements of the identified enterprise scenario. | |

## Recommended Resources

### Textbooks

Forouzan, B.A. (2023) *Data Communications and Networking*. 6th edn. New York: McGraw-Hill.

Graziani, R. and Johnson, A. (2021) *Routing and Switching Essentials Companion Guide*. Indianapolis: Cisco Press.

Odom, W. (2023) *CCNA 200-301 Official Cert Guide*, Volume 1. Indianapolis: Cisco Press.

Oppenheimer, P. (2021) *Top-Down Network Design*. 4th edn. Indianapolis: Cisco Press.

Stallings, W. (2015) *Foundations of Modern Networking*: *SDN, NFV, QoE, IoT, and Cloud*. Boston, MA: Pearson.

Tanenbaum, A.S., Feamster, N. and Wetherall, D.J. (2021) *Computer Networks*. 6th edn. Harlow: Pearson.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.solarwinds.com/blog/the-best-network-monitoring-tools | SolarWinds<br>The 14 Best Network Monitoring Tools for 2026.<br>(Article) |
| www.geeksforgeeks.org/computer-networks/network-monitoring-and-packet-capture-techniques/ | GeeksforGeeks<br>Network Monitoring and Packet Capture Techniques<br>(Article) |
| simeononsecurity.com/network-plus/network-services-connectivity-options-architecture/ | SimeonOnSecurity<br>Network + course: Exploring network services, connectivity options, and architecture<br>(Tutorial) |
| www.tutorialspoint.com/data_communication_computer_network/network_services.htm | Tutorials Point<br>Network Services<br>(Tutorial) |

 © Pearson Education Limited 2026

| | |
|---|---|
| nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-215.pdf | NIST<br>Guide to a Secure Enterprise Network Landscape.<br>(PDF) |
| dev.to/pelk/security-architecture-secure-network-design-principles-1oem | DEV Community<br>Security Architecture: Secure Network Design Principles<br>(Article) |
| www.geeksforgeeks.org/computer-networks/simple-network-management-protocol-snmp/ | GeekforGeeks<br>Simple Network Management Protocol (SNMP)<br>(Article) |
| dev.to/berthaw82414312/network-performance-testing-a-comprehensive-guide-3om7 | DEV Community<br>Network Performance Testing:<br>A Comprehensive Guide<br>(Guide) |
| docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html/managing_networking_infrastructure_services/index | Red Hat Documentation<br>Managing Networking Infrastructure Services<br>(General reference) |

**Journals and articles**

Afolalu, O. and Tsoeu, M.S. (2025) 'Enterprise Networking Optimization: A Review of Challenges, Solutions, and Technological Interventions', *Future Internet*, 17(4), 133. doi.org/10.3390/fi17040133.

Ikpe, K. and Ashigwuike, E. (2025) 'Enterprise Network Design and Security Optimization', *Open Access Library Journal*, 12, pp. 1–17. Available at: doi.org/10.4236/oalib.1112489.

Makwana, A.H. and Atkotiya, K.H. (2023) 'A literature review: extensible network model architecture', *International Journal of Innovative Science and Research Technology*, 8(7). Available at: doi.org/10.5281/zenodo.8186566.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 407: Applied Software Testing*

*Unit 409: Core Cybersecurity*

*Unit 414: Applied Networking*

*Unit 416: Core Cloud Computing.*

 © Pearson Education Limited 2026

# Unit 415: Applied Cybersecurity

**Unit code:** J/652/3267

**Unit level:** 4

**Credit value:** 15

---

## Introduction

As organisations become increasingly dependent on digital systems and interconnected networks, protecting enterprise networks has become a critical requirement across all sectors. Cyberthreats continue to grow in scale and sophistication, posing risks to business operations, sensitive data and organisational reputation. This unit is situated within the applied cybersecurity field and focuses on how enterprise networks are protected, tested and reviewed in real-world operational contexts.

The aim of this unit is to develop students' practical understanding of how enterprise networks can be secured against identified cybersecurity threats. Students will examine the tools, techniques and controls used to protect networks, and apply this knowledge to design, implement and review network security strategies. The unit supports the development of applied skills that are directly relevant to technical cybersecurity and network security roles.

The unit introduces students to key enterprise network security concepts, beginning with an exploration of security tools and techniques used to control access, protect data and manage network traffic. Students then design a network security strategy to address an identified threat, before implementing that design through configuration and controlled penetration testing activities. The unit concludes with a review of network security effectiveness, using evidence from testing, monitoring and performance data to inform improvement and compliance decisions.

On successful completion of this unit, students will be able to apply network security tools appropriately, assess the effectiveness of security controls and communicate informed judgements using industry-relevant terminology. These skills support progression into roles such as Junior Cybersecurity Analyst, Network Security Technician, IT Support Specialist, Security Operations Centre (SOC) Assistant and Infrastructure Support Technician, as well as further study in cybersecurity and network management disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore the range of tools and techniques to secure an enterprise network

LO2 Design a network security strategy to secure a network against an identified threat

LO3 Implement the network security strategy to secure the network against an identified threat

LO4 Review the effectiveness of the implemented network security strategy against the identified threat.

## Essential Content

LO1 **Explore the range of tools and techniques to secure an enterprise network**

*Network security infrastructure tools:*

Perimeter and gateway controls: firewalls e.g. hardware firewalls, software firewalls; gateway devices e.g. security gateways, network gateways; security appliances e.g. unified threat management devices, firewall appliances

Monitoring and detection tools: intrusion detection systems e.g. signature-based detection, anomaly detection, rule-based detection; network monitoring tools e.g. traffic monitoring, performance monitoring; endpoint protection e.g. device protection, malware protection, threat prevention; security data correlation e.g. integrating logs from multiple sources, analysing alerts, identifying potential security breaches; anomaly detection e.g. identification of unusual patterns in network traffic, system or protocol behaviour, log data or packet structures; monitoring configuration e.g. Security Information and Event Management (SIEM) tools, log management systems, alert thresholds configuration and tuning.

*Authentication and access control mechanisms:*

Authentication components: user accounts e.g. staff accounts, administrator accounts; authentication factors e.g. knowledge factors, possession factors, biometric factors; multi-factor authentication (MFA) e.g. password and token, password and biometric

Authorisation management: access permissions e.g. read access, write access, execute access; user roles e.g. administrator roles, standard user roles; access policies e.g. security policies, usage policies

Account and credential controls: password controls e.g. password complexity, password expiry; account management e.g. account creation, account removal; user accounts e.g. active accounts, disabled accounts; access permissions e.g. temporary access, restricted access.

*Network protection and data security techniques:*

Secure communication mechanisms: encryption e.g. data encryption, file encryption; secure protocols e.g. Hypertext Transfer Protocol Secure (HTTPS), Transport Layer Security (TLS), Secure Sockets Layer (SSL); key management e.g. key storage, key rotation, access control

Remote access protections: Virtual private networks (VPN) e.g. site-to-site VPNs, remote access VPNs; secure connections e.g. encrypted tunnels, authenticated sessions

Data protection measures: data confidentiality e.g. access restriction, encryption use; data integrity e.g. integrity checks, hashing; backup protection e.g. secure backups, off-site backups; key management e.g. encryption keys, access keys

Threat intelligence: sources e.g. vulnerability databases, threat feeds; indicators of compromise (IoCs) e.g. known malicious IPs, file hashes, attack signatures.

*Network segmentation and traffic management:*

Network structure controls: network segmentation e.g. logical separation, physical separation; virtual local area networks (VLANs) e.g. departmental VLANs, functional VLANs; internal networks e.g. private networks, internal systems

Traffic regulation mechanisms: traffic filtering e.g. Internet Protocol (IP) filtering, protocol filtering; routing rules e.g. static routes, predefined routes; bandwidth control e.g. traffic prioritisation, bandwidth limits.

## LO2 Design a network security strategy to secure a network against an identified threat

*Threat identification and mitigation methods:*

Threat and vulnerability identification: security threats e.g. malware threats, unauthorised access; vulnerabilities e.g. software vulnerabilities, configuration weaknesses; attack vectors e.g. network attacks, credential attacks; threat sources e.g. internal threats, external threats

Mitigation and response controls: risk levels e.g. high risk, low risk; mitigation actions e.g. blocking actions, configuration changes; control measures e.g. technical controls, administrative controls; security responses e.g. containment actions, recovery actions.

*Network hardening approaches:*

Configuration and system controls: secure configurations e.g. baseline settings, approved configurations; system settings e.g. security options, operational settings; service reduction e.g. disabled services, removed services; default accounts e.g. default user accounts, default administrator accounts

Access and authentication controls: password controls e.g. password complexity, password expiry; configuration files e.g. system configuration files, device configuration files; security settings e.g. firewall settings, authentication settings; hardening checklists e.g. configuration checklists, compliance checklists.

*Patch and update management:*

Patch deployment processes: patch management e.g. patch application, patch scheduling; software updates e.g. operating system updates, application updates; firmware updates e.g. device firmware, network firmware; update schedules e.g. planned updates, maintenance periods

Version and record controls: version control e.g. software versions, firmware versions; vulnerability fixes e.g. security fixes, bug fixes; system patches e.g. critical patches, security patches; update records e.g. update logs, change records.

*Access control and configuration reinforcement:*

Access control enforcement: access control lists (ACLs) e.g. network ACLs, file ACLs; user permissions e.g. read permissions, write permissions; authentication settings e.g. login settings, authentication rules; policy rules e.g. access rules, usage rules

Configuration validation and review: configuration changes e.g. approved changes, logged changes; authorised access e.g. permitted users, approved devices; rule enforcement e.g. firewall rules, access rules; permission reviews e.g. access reviews, role reviews.

*Monitoring and logging mechanisms:*

Log and monitoring sources: system logs e.g. operating system logs, application logs; network logs e.g. traffic logs, connection logs; security logs e.g. firewall logs, intrusion logs; event records e.g. system events, security events

Monitoring and alert outputs: alert notifications e.g. security alerts, system alerts; monitoring tools e.g. log monitoring, network monitoring; audit trails e.g. activity tracking, access tracking; incident logs e.g. security incidents, access incidents.

LO3 **Implement the network security strategy to secure the network against an identified threat**

*Penetration testing concepts and scope:*

Testing authorisation and governance: penetration testing e.g. controlled security testing, authorised security testing; testing scope e.g. defined boundaries, agreed targets; authorisation e.g. written permission, management approval; testing rules e.g. rules of engagement, testing conditions; testing objectives e.g. security validation, weakness identification

Legal and ethical requirements: ethical standards e.g. professional conduct, responsible testing; legal boundaries e.g. computer misuse law, data protection law; testing permissions e.g. written consent, stakeholder approval; testing documentation e.g. authorisation records, scope documents.

*Network reconnaissance and scanning techniques:*

Discovery and mapping activities: network scanning e.g. active scanning, passive scanning; host discovery e.g. live host identification, address discovery; target identification e.g. authorised targets, scoped systems; network mapping e.g. network diagrams, asset mapping

Port and service identification: port scanning e.g. open port detection, closed port identification; service scanning e.g. service detection, version identification; vulnerability scanning e.g. known vulnerability detection, misconfiguration detection; scan results e.g. identified services, detected weaknesses.

*Enumeration and exploitation concepts:*

Information gathering techniques: enumeration e.g. information extraction, service enumeration; system information e.g. operating systems, system versions; service information e.g. running services, service configurations; user information e.g. authorised users, account details

Controlled exploitation activities: configuration weaknesses e.g. insecure settings, default configurations; access weaknesses e.g. weak credentials, excessive permissions; controlled exploits e.g. test exploits, simulated attacks; test environments e.g. isolated environments, controlled systems.

*Penetration testing outputs and reporting:*

Findings and evidence: test findings e.g. identified weaknesses, security gaps; identified vulnerabilities e.g. misconfigurations, unpatched systems; evidence records e.g. screenshots, logs; test logs e.g. activity logs, scan logs; result summaries e.g. summary reports, findings overviews

Risk evaluation and reporting: risk ratings e.g. high risk, low risk; reporting templates e.g. standard reports, structured templates; remediation notes e.g. suggested fixes, improvement actions; test conclusions e.g. security posture conclusions, effectiveness judgements.

LO4 **Review the effectiveness of the implemented network security strategy against the identified threat**

*Performance and effectiveness indicators:*

Security performance measures: security indicators e.g. alert volumes, incident frequency; performance measures e.g. system performance, network performance; detection rates e.g. threat detection rates, alert detection rates; response times e.g. incident response times, alert response times

Outcome and trend analysis: alert accuracy e.g. false positives, true positives; incident counts e.g. security incidents, access incidents; trend data e.g. historical trends, activity trends; effectiveness measures e.g. control effectiveness, response effectiveness

Alignment with requirements: requirement fulfilment e.g. security requirements met, compliance achieved; operational fit e.g. business needs supported, enterprise suitability.

*Incident response and review processes:*

Incident handling activities: incident identification e.g. event identification, alert identification; scope e.g. major, non-major; prioritisation e.g. incidents, actions; response actions e.g. containment actions, mitigation actions; containment steps e.g. system isolation, access restriction; recovery actions e.g. system restoration, service recovery

Evidence collection: evidence handling e.g. secure logging, timestamping, integrity protection, chain of custody, controlled access, secure storage for investigation use.

Incident support: forensic readiness e.g. audit trails, traceability, evidential integrity, analysis support, compliance with legal and organisational investigation procedures.

Post-incident review: response procedures e.g. incident procedures, escalation procedures; incident records e.g. incident reports, response logs; review findings e.g. root causes, response issues; corrective actions e.g. control changes, process improvements

Future threats: horizon scanning e.g. emerging threats, zero-day vulnerabilities, National Cyber Security Centre (NCSC) threat feeds; forward-looking analysis e.g. predicted threat evolution, emerging threat trends

*Compliance and optimisation considerations:*

Governance and compliance review: security policies e.g. organisational policies, network policies; compliance checks e.g. policy checks, control checks; procedural adherence e.g. approved procedures, operational procedures; control reviews e.g. security controls, access controls; current developments, knowledge currency, evolving practice

Improvement and optimisation actions: improvement actions e.g. configuration improvements, process improvements; review findings e.g. security findings, compliance findings; recommendation records e.g. improvement recommendations, review recommendations; improvement plans e.g. action plans, remediation plans

Security audits: activities e.g. participation in audit, evidence gathering, audit documentation; standards alignment e.g. ISO/IEC 27001, National Institute of Standards and Technology Risk Management Framework (NIST RMF), Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE); gap analysis e.g. current state vs target state, control deficiencies, compliance gaps.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explore the range of tools and techniques to secure an enterprise network</td><td rowspan="2">D1 Evaluate, using a selection of real world examples, the effectiveness of network security tools and techniques in securing an enterprise network against identified threats.</td></tr>
<tr><td>P1 Explore, using a selection of real-world examples, the network security tools and access control mechanisms used in the protection of an enterprise network.<br><br>P2 Review, using a selection of real-world examples, the network security tools and techniques used to protect data and manage traffic within an enterprise network.</td><td>M1 Analyse, using a selection of real-world examples, how different network security tools and techniques contribute to the protection of an enterprise network from identified threats.</td></tr>
<tr><td colspan="2">LO2 Design a network security strategy to secure a network against an identified threat</td><td rowspan="4">LO2 and LO3<br><br>D2 Justify the decisions made in the design and implementation of the secure network in being able to effectively protect against the identified threat.</td></tr>
<tr><td>P3 Design a network security strategy to secure a network against an identified threat.<br><br>P4 Review the network security strategy with a range of key stakeholders to identify strengths and weaknesses in securing the network against an identified threat.</td><td>M2 Optimise, using the results of the stakeholder feedback, the network security strategy so that it comprehensively addresses the risk of the identified threat.</td></tr>
<tr><td colspan="2">LO3 Implement the network security strategy to secure the network against an identified threat</td></tr>
<tr><td>P5 Implement the network security strategy to secure the network against an identified threat.<br><br>P6 Implement a selection of penetration tests to thoroughly assess the effectiveness of the network security strategy against an identified threat.</td><td>M3 Optimise, using the results of the penetration testing, the security of the network so that it comprehensively addresses the risk of the identified threat.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Review the effectiveness of the implemented network security strategy against the identified threat | | |
| **P7** Review, using data from penetration testing and performance indicators, the effectiveness of the implemented network security strategy against the identified threat.<br><br>**P8** Develop a regulatory-compliant incident response and review policy that will allow the organisation to deal with the current and future threats. | **M4** Analyse, using data from penetration testing and performance indicators, the effectiveness of the implemented network security strategy against the identified threat suggesting areas for future development. | **D3** Evaluate, using data from penetration testing and performance indicators, the effectiveness of the implemented network security strategy compared with the design in preventing the security risk from the identified threat. |

## Recommended Resources

### Textbooks

Halley, J. *et al*. (2025) *Zero Trust in Resilient Cloud and Network Architectures*. Indianapolis: Cisco Press.

Kim, D. and Solomon, M.G. (2021) *Fundamentals of Information Systems Security*. 4th edn. Burlington, MA: Jones & Bartlett Learning.

Shin, B. (2021) *A Practical Introduction to Enterprise Network and Security Management* (2nd edn.). Boca Raton, FL: Auerbach Publications (Taylor & Francis).

Stallings, W. (2021) *Network Security Essentials: Applications and Standards*, 6th edn, Harlow: Pearson Education.

Whitman, M.E. and Mattord, H.J. (2021) *Principles of Incident Response and Disaster Recovery*, 3rd edn, Boston, MA: Cengage Learning.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.cisa.gov/topics/cybersecurity-best-practices/zero-trust | Cybersecurity and Infrastructure Security Agency (CISA)<br>Zero trust guidance<br>(General reference) |
| www.netmaker.io/resources/threat-modeling | Netmaker<br>Threat Modeling for Company Networks: Step-By-Step Guide<br>(Article) |
| www.nist.gov/cyberframework | National Institute of Standards and Technology<br>Cybersecurity Framework (CSF 2.0)<br>(General reference) |
| www.cyberly.org/en/what-are-the-legal-and-ethical-considerations-in-penetration-testing/index.html | Cyberly<br>Legal And Ethical Considerations In Penetration Testing<br>(Article) |
| www.cloudflare.com/learning/access-management/how-to-implement-zero-trust/ | Cloudflare<br>How to Implement Zero Trust Security<br>(Article) |

| | |
|---|---|
| www.cyberriskinsight.com/cyber-incident/key-metrics-measure-effectiveness-incident/ | Cyber Risk Insight<br>Metrics Used to Measure the Effectiveness of Incident Response and Security Controls<br>(Article) |
| www.splunk.com/en_us/blog/learn/incident-response-metrics.html | Splunk<br>Incident response performance indicators including detection and response time metrics<br>(Blog) |

**Journals and articles**

Alhamed, M. and Rahman, M.H. (2023) 'A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions', *Applied Sciences*, 13(12), 6986. Available at: doi.org/10.3390/app13126986.

Ikpe, K. and Ashigwuike, E. (2025) 'Enterprise Network Design and Security Optimization', *Open Access Library Journal*, 12, e12489. Available at: doi.org/10.4236/oalib.1112489.

Mukherjee, P. *et al*. (2023) 'Detection and Hardening Strategies to Secure an Enterprise Network', *Lecture Notes in Computer Science*, 14424, pp. 91–108. Available at: doi.org/10.1007/978-3-031-49099-6_6.

Younus, Z.S. and Alanezi, M. (2023) 'A Survey on Network Security Monitoring: Tools and Functionalities', *Mustansiriyah Journal of Pure and Applied Sciences*, 1(2). Available at: doi.org/10.47831/mjpas.v1i2.33.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 403: Professional Practice*

*Unit 409: Core Cybersecurity*

*Unit 413: Core Networking.*

# Unit 416: Core Cloud Computing

**Unit code:** K/652/3268

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Cloud computing has become a fundamental part of modern digital systems, underpinning how organisations store data, run applications and deliver services. Rather than relying on physical infrastructure, organisations increasingly use cloud-based platforms to access shared computing resources over the internet. This approach supports more flexible and scalable ways of working, enabling businesses to respond effectively to changing demands while improving efficiency and service delivery.

The aim of this unit is to develop students' understanding of the core concepts and principles that underpin cloud computing. Students will explore how cloud environments are structured, including service models, deployment approaches and key architectural components. The unit also focuses on how cloud-based solutions can be selected and designed to meet defined organisational requirements in a range of business contexts.

The unit covers a range of essential topics that support a practical understanding of cloud computing. Students will examine core characteristics such as scalability, elasticity and virtualisation, along with cloud service models including infrastructure, platform and software services. The structure and function of cloud architectures are explored alongside deployment models such as public, private and hybrid cloud environments. The unit also introduces key considerations such as performance, security, cost and operational effectiveness, and leads to the design and review of a cloud-based solution for a given scenario.

On successful completion of this unit, students will be able to explain core cloud computing concepts, describe cloud service and deployment models, and design and evaluate a cloud-based architecture to meet specific organisational needs. These skills are relevant to a range of technical roles within the IT and digital sector, including Cloud Support Technician, Junior Cloud Engineer, Systems Administrator and Infrastructure Analyst.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Define the key concepts of cloud computing

LO2 Describe the major cloud service models and architectures

LO3 Design a basic cloud-based model for an identified business scenario

LO4 Review the effectiveness of the cloud-based model design for an identified scenario.

## Essential Content

LO1 **Define the key concepts of cloud computing**

*Core characteristics of cloud computing:*

Concept definition and structure: definition e.g. internet-based services; characteristics e.g. on-demand access, shared resources; service e.g. Software as a Service (SaaS), cloud applications; computing e.g. remote processing; infrastructure e.g. virtual servers; differences e.g. on-premise vs cloud

Access and provisioning: self-service e.g. cloud portals; access e.g. web access, Application Programming Interfaces (APIs); pooling e.g. multi-tenancy; management e.g. dashboards

Performance and scaling: elasticity e.g. automatic scaling; scalability e.g. storage expansion; flexibility e.g. resource configuration; availability e.g. uptime; drivers e.g. user demand.

*Virtualisation and abstraction:*

Virtualisation components: virtualisation e.g. hardware abstraction; machines e.g. virtual machines; hypervisors e.g. virtualisation software; systems e.g. data centres

Abstraction and isolation: abstraction e.g. service layers; isolation e.g. tenant separation; relationship e.g. virtual resource mapping

Efficiency and portability: utilisation e.g. shared resources; portability e.g. application transfer; systems e.g. container environments.

*Resource pooling, scalability and elasticity:*

Resource types: resources e.g. compute, storage, network; compute e.g. virtual CPU; storage e.g. object storage; network e.g. virtual networks

Allocation and demand: allocation e.g. dynamic provisioning; demand e.g. workload demand; workload e.g. applications; distribution e.g. load balancing

Scaling behaviour: scaling e.g. vertical, horizontal; scalability e.g. capacity growth; elasticity e.g. demand-based adjustment; distinction e.g. fixed vs dynamic; responsiveness e.g. rapid provisioning.

*Measured service and operational control:*

Monitoring and visibility: monitoring e.g. performance monitoring; visibility e.g. dashboards; dashboards e.g. management interfaces; logs e.g. activity logs; alerts e.g. notifications

Measurement and billing: metering e.g. usage tracking; billing e.g. pay-as-you-go; tracking e.g. cost monitoring; transparency e.g. usage reports

Operational optimisation: service e.g. application delivery; budgeting e.g. cost control; optimisation e.g. resource efficiency.

*Security:*

Core security principles: confidentiality e.g. data protection; integrity e.g. data accuracy; availability e.g. system access; governance e.g. policies; compliance e.g. regulations

Security techniques: threat modelling e.g. attack surface identification, risk scenarios; vulnerability scanning e.g. automated scans, security tools; dependency checking e.g. library validation, version control

Identity and access: identity and access management (IAM) e.g. identity management; authentication e.g. multi-factor authentication (MFA); authorisation e.g. role-based access; access control e.g. permissions

Data protection: encryption e.g. data encryption; data processing e.g. data handling; data storage e.g. cloud databases; data transmission e.g. secure transfer

Monitoring and risk: monitoring e.g. security monitoring; logging e.g. audit logs; auditing e.g. reviews; threats e.g. cyber threats; risks e.g. misconfiguration

Security architecture: architecture e.g. layered security; defence in depth e.g. multiple controls; responsibility e.g. shared responsibility

Cloud security concepts: Secure Access Service Edge (SASE), Zero Trust Networking (ZTN), change management processes

Cloud certification awareness: certification frameworks e.g. Amazon Web Services (AWS) Certified Cloud Practitioner, Microsoft Azure Fundamentals, Google Associate Cloud Engineer; requirements e.g. skills domains, assessment objectives, recertification expectations; alignment e.g. mapping cloud knowledge and practices to recognised industry certification standards.

LO2 **Describe the major cloud service models and architectures**

*Cloud service models:*

Service model types: infrastructure e.g. servers, storage, networks; platform e.g. application platforms, runtime environments, development tools; software e.g. web applications, email systems, business software

Model definitions and scope: service e.g. service delivery, hosted services, cloud provision; Infrastructure as a Service (IaaS) e.g. virtual machines, storage services, network services; Platform as a Service (PaaS) e.g. development platforms, application hosting, runtime services; SaaS e.g. web apps, cloud software, online systems; models e.g. service layers;

Roles and usage: provider e.g. cloud providers, service vendors, platform owners; customer e.g. users, organisations, clients; uses e.g. hosting, storage, applications; control e.g. system access, configuration, management; flexibility e.g. service options, resource choices; support e.g. managed services, updates, maintenance.

*Cloud deployment models:*

Deployment types: public e.g. shared cloud, internet services, hosted platforms; private e.g. dedicated cloud, internal systems, enterprise cloud; hybrid e.g. combined environments, mixed systems, linked platforms; community e.g. sector sharing, shared services, group platforms; multi-cloud e.g. multiple providers, different platforms

Ownership and management: ownership e.g. provider ownership, organisational ownership; management e.g. provider management, internal control; access e.g. internet access, restricted access; considerations e.g. security, privacy, control

Benefits and constraints: advantages e.g. scalability, flexibility, cost efficiency; limitations e.g. dependency, limited control, connectivity reliance; uses e.g. business systems, data storage, applications; deployment e.g. environment selection, service choice.

*Cloud architecture components:*

Core components: compute e.g. virtual machines, processors, instances; storage e.g. cloud storage, databases, file storage; networking e.g. virtual networks, connections, routing; databases e.g. relational databases, data services, storage systems

Management and security: identity e.g. users, accounts, identities; access e.g. permissions, roles, controls; management e.g. dashboards, tools, interfaces; security e.g. protection, controls, safeguards

Structure and integration: interfaces e.g. web interfaces, cloud portals, admin consoles; orchestration e.g. service co-ordination, resource scheduling, process control; relationships e.g. component links, service interactions, data flow; frontend e.g. browsers, client apps; backend e.g. servers, databases, storage systems; services e.g. compute services, storage services, network services.

*Architecture characteristics:*

Performance and reliability: availability e.g. uptime, access, service continuity; fault tolerance e.g. redundancy, backup systems; resilience e.g. recovery, stability; scalability e.g. growth, expansion

Maintainability and security: maintainability e.g. updates, maintenance, support; security e.g. protection, data security, access control; decisions e.g. configuration, setup

Continuity and alignment: continuity e.g. service uptime, system availability; recovery e.g. backups, restoration; access e.g. user access, system entry; workload e.g. applications, processing tasks; distribution e.g. load balancing, traffic distribution.

LO3 **Design a basic cloud-based model for an identified business scenario**

*Requirements analysis:*

Requirements identification: interpretation e.g. requirement analysis, scenario understanding; scenarios e.g. e-commerce systems, file storage systems, web applications; identification e.g. user needs, system needs

System and data needs: access e.g. user access, remote access; applications e.g. web applications, business systems; data e.g. files, databases, records; availability e.g. uptime, system access; performance e.g. speed, response time; security e.g. access control, data protection

Constraints and resources: growth e.g. user growth, data growth; constraints e.g. budget limits, storage limits, capacity limits; capability e.g. system capacity, storage capacity, compute capacity; budget e.g. costs, spending; timescale e.g. deadlines, schedules; staffing e.g. support staff, IT staff; capability e.g. system capacity, resource limits.

*Model design:*

Model selection and structure: selection e.g. service model choice; service e.g. IaaS, PaaS, SaaS; deployment e.g. public cloud, private cloud; models e.g. service models; components e.g. compute, storage

Design specification: requirements e.g. system needs, user needs; diagrams e.g. architecture diagrams, system diagrams; descriptions e.g. system description, architecture outline; decisions e.g. configuration, selection

Technical components: compute e.g. virtual machines, processors; storage e.g. cloud storage, databases; networking e.g. networks, connections; identity e.g. users, accounts; management e.g. dashboards, tools

Change management considerations: processes e.g. change control procedures, approval workflows, implementation tracking; compliance considerations e.g. alignment with change policies, awareness of configuration validation; secure architecture application e.g. SASE, ZTN; integration e.g. relationship between approved changes and security architecture; verification e.g. review of changes, confirmation of system outputs.

*Capacity, resilience and continuity:*

Capacity planning: capacity e.g. storage size, compute capacity; planning e.g. forecasting, estimation; compute e.g. processing power, servers; storage e.g. data capacity; estimation e.g. usage estimates

Performance and utilisation: bandwidth e.g. network speed, connection speed; requirements e.g. system needs, user needs; utilisation e.g. resource use, system usage; forecasts e.g. demand forecasts

Resilience and recovery: resilience e.g. system stability, fault tolerance; redundancy e.g. backup systems, duplicate resources; replication e.g. data copies, data duplication; backup e.g. data backup, system backup; recovery e.g. data recovery, system restoration; downtime e.g. service interruption.

*Cost and operational considerations:*

Cost models and factors: expenditure e.g. costs, charges, usage costs; capital e.g. setup costs, infrastructure costs, initial configuration; operational e.g. running costs, service costs, usage costs; pricing e.g. pay-as-you-go, subscription pricing, usage pricing

Usage and estimation: consumption e.g. resource use, service usage; storage e.g. data storage, backups; transfer e.g. data transfer, data movement; estimation e.g. cost estimates, usage estimates

Control strategies: regulatory controls e.g. policies, rules; shutdown e.g. resource stopping, service stopping; monitoring e.g. usage tracking, system monitoring.

*Optimisation opportunities:*

Opportunity identification: identification e.g. improvement areas, system gaps; opportunities e.g. efficiency gains, performance gains; utilisation e.g. resource use, system use

Performance improvement: resilience e.g. system strength, reliability; efficiency e.g. optimisation, improved use; effectiveness e.g. system performance

System refinement: automation e.g. automated processes, scheduled tasks; refinement e.g. updates, adjustments; monitoring e.g. tracking, observation; trade-offs e.g. cost vs performance, speed vs cost.

LO4 **Review the effectiveness of the cloud-based model design for an identified scenario**

*Performance and effectiveness:*

Evaluation measures: evaluation e.g. performance review, system review; performance e.g. response time, processing speed, system efficiency; requirements e.g. system needs, user needs; measures e.g. metrics, indicators

System responsiveness: responsiveness e.g. response time, system speed; throughput e.g. request rates, data transfer rates, processing capacity; availability e.g. uptime, access; reliability e.g. system stability, consistency

Quality and suitability: quality e.g. performance level, system quality; judgement e.g. evaluation, assessment; suitability e.g. system fit, appropriateness; context e.g. business use, user needs.

*Metrics and key performance indicators:*

Measurement metrics: metrics e.g. performance data, usage data, monitoring data; key performance indicators (KPIs) e.g. uptime targets, response targets, latency targets; KPIs e.g. targets, goals; uptime e.g. availability, service time; latency e.g. delay, response time

Performance indicators: throughput e.g. processing capacity, output rate; utilisation e.g. resource usage, system use; recovery e.g. restore time, recovery rate

Evaluation and interpretation: efficiency e.g. resource use, system performance; indicators e.g. metrics, data; interpretation e.g. analysis, understanding; value e.g. benefit, usefulness.

*Risks, limitations and sustainability:*

Risk factors: risks e.g. data loss, security breaches, service outages; security e.g. unauthorised access, data exposure, vulnerabilities; security e.g. threats, vulnerabilities; compliance e.g. regulations, policies; dependency e.g. provider reliance

Constraints and complexity: lock-in e.g. vendor dependency, provider reliance; complexity e.g. system complexity, integration difficulty; limitations e.g. system limits, resource limits

Sustainability and capability: maintainability e.g. system upkeep, updates; adaptability e.g. flexibility, change; growth e.g. scalability, expansion; capability e.g. system capacity

Costs evaluation and control: cost considerations e.g. operational costs, usage costs, scaling costs; cost trade-offs e.g. cost vs performance, cost vs availability; monitoring e.g. usage tracking, billing dashboards.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Define the key concepts of cloud computing | | **LO1 and LO2**<br>**D1** Evaluate, using a selection of real-world examples, the extent to which cloud architecture designs support organisational effectiveness and service quality. |
| **P1** Define, using a selection of real-world examples, the core characteristics of cloud computing.<br>**P2** Explain, using a selection of real-world examples, how virtualisation, resource pooling, scalability, elasticity and security support cloud service delivery. | **M1** Analyse, using a selection of real-world examples, how cloud computing concepts support flexible, scalable and efficient computing environments. | |
| **LO2** Describe the major cloud service models and architectures | | |
| **P3** Describe, using a selection of real-world examples, the cloud service models used in identified organisational contexts.<br>**P4** Outline, using a selection of real-world examples, the deployment models and core architecture components used in cloud environments. | **M2** Assess, using a selection of real-world examples, the suitability of cloud service models, deployment models and architecture components for a defined organisational context. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a basic cloud-based model for an identified business scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design of the cloud-based model that allows the comprehensive delivery of the requirements for an identified scenario. |
| **P5** Design a cloud-based model that addresses defined functional and non-functional requirements for an identified business scenario.<br>**P6** Review the design of the cloud-based model with a range of identified internal stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the stakeholder feedback, the design of the cloud-based model for an identified business scenario. | |
| **LO4** Review the effectiveness of the cloud-based model design for an identified scenario | | |
| **P7** Review the effectiveness of the cloud-based model design for an identified scenario.<br>**P8** Identify risks, limitations and improvement opportunities associated with the cloud-based model design for an identified scenario. | **M4** Evaluate the effectiveness of the cloud-based model design in meeting the functional and non-functional requirements for an identified scenario. | |

## Recommended Resources

### Textbooks

Erl, T. and Monroy, E. (2023) *Cloud Computing: Concepts, Technology, Security and Architecture*. 2nd edn. Boston: Pearson.

Furht, B. and Escalante, A. (2024) *Handbook of Cloud Computing*. 2nd edn. Cham: Springer.

Mishra, P., Pilli, E.S. and Joshi, R.C. (2022) *Cloud Security: Attacks, Techniques, Tools, and Challenges*. Boca Raton: CRC Press.

Ruparelia, N.B. (2023) *Cloud Computing*. Revised edn. Cambridge, MA: MIT Press.

Sehgal, N.K., Bhatt, P.C.P. and Acken, J.M. (2023) *Cloud Computing with Security and Scalability: Concepts and Practices*. 3rd edn. Cham: Springer.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| aws.amazon.com/what-is-cloud-computing/ | Amazon Web Services (AWS)<br>Cloud computing concepts, service models and architectural overview<br>(General reference) |
| cloud.google.com/learn/what-is-cloud-computing | Google Cloud<br>Cloud deployment models and cloud architecture fundamentals<br>(General reference) |
| www.geeksforgeeks.org/cloud-computing/ | GeeksforGeeks<br>Cloud computing concepts, scalability and elasticity explanations<br>(Tutorials) |
| www.ibm.com/topics/cloud-computing | IBM Cloud<br>Cloud service models, hybrid cloud concepts and enterprise use cases<br>(General reference) |
| learn.microsoft.com/en-us/azure/cloud-adoption-framework/ | Microsoft Azure<br>Cloud architecture principles and adoption frameworks<br>(Training) |

| | |
|---|---|
| www.nist.gov | National Institute of Standards and Technology (NIST)<br>Authoritative cloud computing definitions and characteristics<br>(General reference) |
| www.redhat.com/en/topics/cloud-computing | Red Hat<br>Cloud architectures, virtualisation and hybrid cloud principles<br>(General reference) |
| learn.microsoft.com/en-us/training/modules/describe-cloud-service-types/ | Microsoft Learn<br>Overview of cloud service models including IaaS, PaaS and SaaS with explanations of use cases<br>(Tutorial) |
| www.datacamp.com/blog/cloud-service-models | DataCamp<br>Detailed breakdown of cloud service models and how they support scalability and application delivery<br>(Article) |
| docs.aws.amazon.com/wellarchitected/latest/cost-optimization-pillar/welcome.html | Amazon Web Services (AWS)<br>Guidance on monitoring, optimisation and cost management in cloud architectures<br>(General reference) |
| www.cloudflare.com/learning/access-management/how-to-implement-zero-trust/ | Cloudflare<br>Introduction to Zero Trust security model including authentication, monitoring and access control in cloud systems<br>(Tutorial) |

**Journals and articles**

Ali, K.A., Fadare, O.A. and Al-Turjman, F. (2025) 'Predictive Analytics and Optimization of Cloud Computing Performance Using Deep Learning', in Al-Turjman, F. (ed.) *Smart Infrastructures in the IoT Era*. Cham: Springer Nature Switzerland, pp. 1051–1063. Available at: doi.org/10.1007/978-3-031-72509-8_86.

Lehrig, S., Eikerling, H. and Becker, S. (2015) 'Scalability, Elasticity, and Efficiency in Cloud Computing: a Systematic Literature Review of Definitions and Metrics, in *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA '15)*, Montréal, QC, Canada, 4–8 May 2015. New York: ACM, pp. 83–92. Available at: doi.org/10.1145/2737182.2737185.

Stamenkov, G. (2024) 'Cloud service models, business continuity and disaster recovery plans, and responsibilities', *International Journal of Organizational Analysis*, 33(3), pp. 437–451. Available at: doi.org/10.1108/IJOA-12-2023-4127.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 413: Core Networking*

*Unit 414: Applied Networking*

*Unit 417: Cloud DevOps Automation*

*Unit 418: Cloud Networking*

*Unit 419: Cloud Security.*

# Unit 417: Cloud DevOps Automation

**Unit code:** L/652/3269

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Cloud computing and DevOps practices have become central to the way modern digital services are developed and maintained. Organisations increasingly rely on cloud platforms to deliver scalable, reliable and high-performing applications, while DevOps approaches support closer collaboration between development and operations teams. Together, these approaches enable faster delivery cycles, improved system resilience and the ability to respond effectively to changing organisational and user requirements.

The aim of this unit is to develop students' understanding of how DevOps operates within cloud environments, focusing on the integration of automated processes throughout the software development life cycle. Students will explore key concepts such as continuous integration, continuous delivery and workflow automation, and will examine how these practices support efficient and reliable application deployment. The unit is designed to link technical knowledge with practical application in realistic development scenarios.

This unit covers a range of topics essential to cloud-based DevOps practices. Students will examine the principles of DevOps and how they are applied within cloud environments, before exploring the structure and function of DevOps pipelines and automation processes. The role of key toolsets, including version control, infrastructure as code and monitoring tools, is also investigated. The unit culminates in the design and configuration of an automated cloud-based development workflow, bringing together pipeline stages, tool integration, security considerations and performance requirements.

On successful completion of this unit, students will be able to explain DevOps concepts in cloud environments, examine DevOps pipelines, assess the contribution of DevOps toolsets to integrated development practices, and design an automated workflow to support cloud-based software delivery. The unit supports progression to further study in cloud computing, software engineering, infrastructure automation and DevOps-related areas, and provides a foundation for entry-level roles such as Junior DevOps Engineer, Cloud Support Technician, Platform Support Assistant, Software Development Support Technician and IT Operations Technician.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the concepts of DevOps development in the cloud

LO2 Discuss the core DevOps pipeline components and their role in improving software delivery

LO3 Explore the role of a range of DevOps toolsets in supporting integrated development

LO4 Design an automated cloud-based development workflow for a given scenario.

## Essential Content

LO1 **Explain the concepts of DevOps development in the cloud**

*DevOps principles:*

Purpose: principles e.g. standards, frameworks; purpose e.g. goals, objectives; collaboration e.g. teamwork, co-ordination; responsibility e.g. ownership, accountability

Delivery: delivery e.g. pipelines, releases; support e.g. services, operations; reliability e.g. uptime, stability; improvement e.g. optimisation, enhancement

Feedback and automation: feedback e.g. metrics, logs; loops e.g. iterations, cycles; automation e.g. pipelines, scripts.

*Cloud-enabled DevOps:*

Infrastructure and environments: cloud e.g. Amazon Web Services (AWS), Azure; environments e.g. cloud environments, staging, production; infrastructure e.g. servers, containers

Scalability and resilience: scalability e.g. scaling, capacity; elasticity e.g. resources, allocation; availability e.g. uptime, access; resilience e.g. redundancy, recovery

Development and delivery: abstraction e.g. services, layers; practices e.g. workflows, methods; development e.g. applications, microservices; delivery e.g. pipelines, releases.

*Continuous integration and delivery practices:*

Code and control: integration e.g. pipelines, workflows; code e.g. scripts, applications; source e.g. repositories, Git; control e.g. versioning, configuration

Build and deployment: builds e.g. artefacts, container images; deployment e.g. releases, rollouts; release e.g. versions, updates; delivery e.g. pipelines, distribution

Testing and flow: testing e.g. validation, checks; stages e.g. phases, steps; flow e.g. sequence, pipelines.

*Working practices and accountability:*

Communication and workflow: working e.g. tasks, operations; communication e.g. messaging, alerts; workflow e.g. pipelines, processes; visibility e.g. dashboards, tracking

Ownership and responsibility: ownership e.g. control, stewardship; responsibility e.g. accountability, roles; outcomes e.g. results, outputs

Performance and continuity: performance e.g. efficiency, throughput; incidents e.g. failures, outages; continuity e.g. stability, maintenance

Collaborative development techniques: pair programming e.g. driver-navigator roles, real-time code review; mob programming e.g. whole-team coding, shared decision-making; application contexts e.g. pair programming for complex feature development, debugging, onboarding and knowledge transfer; mob programming for high-risk changes, critical system components, architectural decisions, rapid alignment across the team; code techniques e.g. merging, branching by abstraction, feature toggling.

LO2 **Discuss the core DevOps pipeline components and their role in improving software delivery**

*Pipeline stages:*

Structure and flow: structure e.g. pipelines, workflows; source e.g. cloud repositories, Git; build e.g. artefacts, container images; test e.g. unit tests, integration tests; release e.g. versions, registries; deployment e.g. rollouts, clusters; monitoring e.g. cloud monitoring, metrics

Dependencies and consistency: artefacts e.g. images, templates; flow e.g. sequencing; dependencies e.g. services, Application Programming Interfaces (APIs); stages e.g. phases, steps; consistency e.g. standards, parity.

*Pipeline automation:*

Automation processes: automation e.g. continuous integration/continuous deployment (CI/CD); processes e.g. workflows, jobs

Efficiency and repeatability: reduction e.g. manual work, errors; intervention e.g. input, actions; repeatability e.g. reuse; activities e.g. tasks, operations; optimisation e.g. reduced rework, elimination of repetitive tasks; decision factors e.g. effort versus benefit, cost and resource use

Error and drift control: reduction e.g. errors, failures; drift e.g. configuration drift, environment inconsistency; control e.g. validation, enforcement, reconciliation.

*Testing and validation:*

Testing processes: testing e.g. unit tests, API tests; integration e.g. services, systems; system e.g. applications, platforms; execution e.g. runs, jobs

Quality and reporting: reporting e.g. logs, reports; failure e.g. errors, faults; handling e.g. alerts, responses; validation e.g. checks, verification

Quality assurance: quality e.g. standards, metrics; defects e.g. bugs, issues.

*Release and deployment approaches:*

Delivery models: delivery e.g. pipelines, releases; deployment e.g. rollouts, updates; models e.g. blue-green, canary; approaches e.g. strategies, methods; mechanisms e.g. tools, platforms

Environment progression: promotion e.g. stages, transitions; environments e.g. cloud environments, staging, production; consistency e.g. standards, parity

Life cycle environments: development e.g. environments, sandboxes; testing e.g. stages; production e.g. services, systems.

LO3 **Explore the role of a range of DevOps toolsets in supporting integrated development**

*Version control systems:*

Repositories and structures: version e.g. versioning, revisions; control e.g. management, configuration; systems e.g. platforms, tools; repositories e.g. Git, cloud repositories; structures e.g. branches, directories

Branching and workflows: branching e.g. branches, forks; strategies e.g. workflows, models; processes e.g. merges, updates; workflows e.g. pipelines, processes

Traceability and changes: requests e.g. pull requests, merge requests; reviews e.g. code reviews, peer reviews; traceability e.g. tracking, audit; changes e.g. commits, updates; history e.g. logs, records.

*Infrastructure as Code (IaC) and provisioning tools:*

Configuration e.g. templates, settings; control e.g. versioning, management; repositories e.g. Git, cloud repositories; structures e.g. templates, files; specification e.g. defining infrastructure using domain-specific languages, resource configurations, declarative infrastructure definitions

Provisioning strategies: strategies e.g. provisioning, deployment; approaches e.g. methods, techniques; processes e.g. workflows, pipelines; workflows e.g. deployment pipelines

Infrastructure tracking: requests e.g. pull requests, updates; reviews e.g. reviews, approvals; traceability e.g. tracking, audit; changes e.g. commits, updates; history e.g. logs, records.

*Build, integration and deployment tools:*

Build and integration systems: build e.g. artefacts, container images; integration e.g. pipelines, workflows; deployment e.g. releases, rollouts; tools e.g. platforms, utilities; automation e.g. pipelines, scripts

Execution and management: servers e.g. runners, agents; management e.g. control, scheduling; tools e.g. utilities, platforms; packaging e.g. artefacts, images; artefacts e.g. container images, packages

Repositories and execution: repositories e.g. Git, cloud repositories; integration e.g. pipelines, workflows; tools e.g. platforms, utilities; execution e.g. runs, jobs.

*Monitoring, logging and observability tools:*

Monitoring systems: monitoring e.g. logs, metrics, dashboards; tools e.g. platforms, systems; data e.g. logs, metrics

Performance and alerting: monitoring e.g. tracking, monitoring; performance e.g. efficiency, latency; infrastructure e.g. servers, containers; alerting e.g. alerts, notifications; notification e.g. messages, alerts

Visibility and response: systems e.g. services, platforms; dashboards e.g. panels, dashboards; visibility e.g. tracking, reporting; data e.g. metrics, logs; response e.g. actions, updates; improvement e.g. optimisation, tuning.

*Real-world Examples:*

A range of DevOps tool examples e.g.

Version control systems e.g. Git, GitHub, GitLab, Bitbucket

IaC tools e.g. Terraform, AWS CloudFormation, Azure Bicep, Azure Resource Manager (ARM) Templates.

Build and deployment tools e.g. Jenkins, GitHub Actions, GitLab CI/CD, Azure Pipelines (Azure DevOps)

Monitoring tools e.g. Prometheus, Grafana, Datadog, Splunk.

LO4 **Design an automated cloud-based development workflow for a given scenario**

*Workflow design and structure:*

Definition and structure: workflow e.g. pipelines, workflows; design e.g. structures, layouts; structure e.g. architectures, frameworks; definition e.g. specifications, outlines

Sequencing and life cycle: stages e.g. phases, steps; sequencing e.g. order, flow; life cycle e.g. phases, stages; activities e.g. tasks, operations

Inputs and dependencies: identification e.g. inputs, outputs; inputs e.g. data, triggers; outputs e.g. artefacts, results; dependencies e.g. services, APIs

Mapping and representation: mapping e.g. diagrams, mappings; diagrams e.g. flowcharts, schematics.

*Toolchain integration and orchestration:*

Governance and control: selection e.g. criteria, options; control e.g. policies, rules; management e.g. processes, functions; scheduling e.g. timelines, intervals; triggers e.g. events, conditions

Development and delivery: build e.g. artefacts, container images; testing e.g. unit tests, integration tests; deployment e.g. releases, rollouts; tools e.g. platforms, systems

Execution and operations: orchestration e.g. co-ordination of stages, automation; execution e.g. runs, jobs; monitoring e.g. logs, metrics, dashboards; release automation e.g. build, test and deployment stages, promotion across environments; delivery flow e.g. source control to deployment pipelines, continuous delivery to target environments and end users

Service navigation and troubleshooting: system navigation e.g. tracing requests across services, following workflows across stages; distributed systems e.g. service dependencies, stateful services, data persistence across components; issue identification e.g. log correlation, metrics analysis, cross-service fault isolation, end-to-end service diagnosis

Immutable infrastructure: immutable models e.g. replace-not-modify infrastructure, stateless deployments; refresh approaches e.g. redeployment, rebuild; updates e.g. operating system refresh, container image rebuilds, security patching through replacement; infrastructure deployment e.g. provisioning using infrastructure as code, pipeline-triggered deployments, automated instance rollouts.

*Security and access control within workflows:*

Security and control: security e.g. protection, encryption; access e.g. permissions, roles; control e.g. policies, rules, access control lists (ACL)

Identity and management: identity e.g. users, roles; management e.g. administration, governance

Security in pipelines: threat modelling, vulnerability scanning, dependency checking; automated CI/CD validation; attack surface reduction

Credentials and validation: roles e.g. permissions, roles; storage e.g. vaults; credentials e.g. keys, tokens; secrets e.g. tokens; checks e.g. validation, policies; dependencies e.g. services, libraries; scanning e.g. scans, tools; configuration e.g. settings, parameters; validation e.g. verification, checks.

*Performance, scalability and efficiency:*

Performance and scalability: performance e.g. efficiency, throughput; scalability e.g. scaling, capacity; efficiency e.g. optimisation, utilisation

Design and workflows: design e.g. architectures, structures

Processing and optimisation: workloads e.g. tasks, processes; allocation e.g. resources, capacity; processes e.g. operations; optimisation e.g. tuning, improvement; processing e.g. handling, execution.

*Cloud execution models and deployment targets:*

Execution models: models e.g. containers, serverless

Environments and execution: environments e.g. cloud environments, staging, production; execution e.g. instances

Services and selection: platforms e.g. cloud services; workflows e.g. processes; selection e.g. options, choices; requirements e.g. needs, specifications.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the concepts of DevOps development in the cloud</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Evaluate, using a selection of real-world examples, the effectiveness of DevOps principles, pipeline components and toolsets in supporting reliable and scalable cloud-based software delivery.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the concepts of DevOps development in the cloud.<br><br>P2 Explain, using a selection of real-world examples, how continuous integration, delivery and working practices operate within cloud-based development environments.</td><td>M1 Assess, using a selection of real-world examples, how DevOps practices allow organisations to deliver robust and secure applications continuously and quickly.</td></tr>
<tr><td colspan="2">LO2 Discuss the core DevOps pipeline components and their role in improving software delivery</td></tr>
<tr><td>P3 Explain, using a selection of real-world examples, the core components of the DevOps pipeline.<br><br>P4 Discuss, using a selection of real-world examples, how DevOps pipeline components help improve software delivery.</td><td>M2 Analyse, using a selection of real-world examples, how DevOps pipeline components contribute to consistency, efficiency and reliability in software delivery.</td></tr>
<tr><td colspan="2">LO3 Explore the role of a range of DevOps toolsets in supporting integrated development</td></tr>
<tr><td>P5 Review, using a selection of real-world examples, the DevOps toolsets used for version control, infrastructure automation and monitoring.<br><br>P6 Explore, using a selection of real-world examples, how DevOps toolsets support integrated development workflows in cloud environments.</td><td>M3 Analyse, using a selection of real-world examples, how DevOps toolsets support integration, automation and operational visibility across development processes.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Design an automated cloud-based development workflow for a given scenario | | |
| **P7** Design an automated cloud-based development workflow for a given scenario.<br><br>**P8** Configure a workflow that integrates DevOps tools, pipeline stages and cloud services including security controls and performance considerations. | **M4** Evaluate the effectiveness of the workflow design in supporting the identified requirements of the given scenario. | **D2** Justify the decisions made in the design and configuration of the automated cloud-based workflow in how comprehensively it meets the requirements of the given scenario. |

## Recommended Resources

### Textbooks

Humble, J. and Farley, D. (2010) *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Boston, MA: Addison-Wesley.

Kim, G., Humble, J. and Willis, J. (2023) *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organisations*. Updated edn. Portland: IT Revolution Press.

Forsgren, N., Humble, J. and Kim, G. (2018) *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. Portland: IT Revolution Press.

Arundel, J. and Domingus, J. (2019) *Cloud Native DevOps with Kubernetes*. Sebastopol: O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.atlassian.com/devops/continuous-delivery-tutorials | Atlassian<br>Continuous integration, delivery and deployment (CI/CD) concepts and practices in DevOps<br>(Tutorial) |
| octopus.com/devops/ci-cd/ci-cd-tutorial/ | Octopus Deploy<br>Step-by-step introduction to building CI/CD pipelines and automated deployment workflows<br>(Tutorial) |
| learn.microsoft.com/en-us/azure/devops/pipelines/architectures/devops-pipelines-baseline-architecture?view=azure-devops | Microsoft Learn<br>Overview of DevOps pipeline architecture including integration, testing and deployment stages<br>(General reference) |
| www.datacamp.com/tutorial/azure-pipelines | DataCamp<br>Guide to Creating and Managing Cloud-Based Pipelines with Automated Build, Test and Deployment<br>(Tutorial) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.igmguru.com/blog/microsoft-azure-devops-tutorial | igmGuru<br>Introduction to Azure DevOps tools including version control, pipelines and workflow automation<br>(Blog) |
| developer.hashicorp.com/terraform/tutorials/aws-get-started/infrastructure-as-code | HashiCorp Developer<br>Introduction to Infrastructure as Code using Terraform, including provisioning and life cycle workflows<br>(Article) |
| learn.microsoft.com/en-us/training/modules/terraform-introduction-to-infrastructure-as-code/ | Microsoft Learn<br>Fundamentals of Infrastructure as Code concepts and terraform usage in cloud environments<br>(Training) |
| code2deploy.com/blog/complete-devops-logging-and-monitoring-guide-observability-in-the-real-world/ | Code2Deploy<br>Overview of monitoring, logging and observability in DevOps environments with practical tool examples<br>(Article) |
| dev.to/cliffordisaboke/monitoring-logging-and-observability-in-devops-jn5 | DEV Community<br>Explanation of monitoring, logging and observability concepts and their role in maintaining system reliability<br>(Article) |

## Journals and articles

Karlovs-Karlovskis, U. *et al*. (2025) 'From software architecture models to pipelines: a conceptual framework for model transformation in DevOps', *Frontiers in Computer Science, 7*. Available at: doi.org/10.3389/fcomp.2025.1714197.

Pastrana, M. *et al*. (2025) 'Best practices evidenced for software development based on DevOps and Scrum', *Applied Sciences*, 15(10), 5421. Available at: doi.org/10.3390/app15105421.

Rifa'i, M.H. and Tiaharyadini, R. (2023) 'DevOps, continuous integration and continuous deployment methods for software deployment automation', *Jurnal Informatika dan Sains*, 6(2), pp. 116–123. Available at: doi.org/10.31326/jisa.v6i2.1751.

Saxena, A., Singh, S., Prakash, S., Yang, T. and Rathore, R.S. (2025) 'DevOps Automation Pipeline Deployment with IaC (Infrastructure as Code)', *in 2024 IEEE Silchar Subsection Conference (SILCON 2024)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers (IEEE), pp. 1–6. Available at: doi.org/10.1109/SILCON63976.2024.10910699.

Tanzil, M.H., Sarker, M., Uddin, G. and Iqbal, A. (2024) 'A mixed method study of DevOps challenges', *arXiv*. doi.org/10.48550/arXiv.2403.16436.

## Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 416: Core Cloud Computing*

*Unit 418: Cloud Networking*

*Unit 419: Cloud Security.*

 © Pearson Education Limited 2026

# Unit 418: Cloud Networking

**Unit code:** **T/652/3270**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

Cloud networking underpins the delivery of modern digital services, enabling organisations to connect applications, users and data across distributed environments. As organisations migrate from traditional on-premises infrastructure to cloud-based platforms, networking models have evolved to support scalable, resilient and secure connectivity. Cloud networking now forms a critical part of infrastructure design across sectors including finance, healthcare, retail, education and digital services, where reliable and performant network architectures are essential to service delivery.

The aim of this unit is to develop students' understanding of the principles and practical applications of cloud networking. It focuses on how networking concepts are adapted within cloud environments, including the use of virtual networks, service models and different approaches to connectivity. Students will explore how cloud networking supports organisational requirements, and how design decisions are influenced by factors such as performance, security and scalability.

The unit provides a structured overview of key cloud networking concepts and practices. Students will examine network architectures such as hub-and-spoke, mesh and hybrid models, along with common connectivity methods including virtual private networks (VPNs), peering and private links. They will also explore how networks are designed to meet defined requirements, including addressing, routing, segmentation and security controls.

The unit culminates in the design, development, monitoring and testing of a cloud network infrastructure solution, enabling students to apply their knowledge in a practical, scenario-based context.

On successful completion of this unit, students will be able to explain cloud networking concepts, analyse network architectures, design cloud network solutions and evaluate their effectiveness in meeting organisational requirements. This provides a foundation for progression to advanced study in cloud infrastructure, network security and distributed systems, and supports entry-level roles such as Cloud Support Technician, Network Technician, Infrastructure Support Engineer and Junior Cloud Engineer.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the fundamental concepts of cloud networking

LO2 Explore a range of cloud-based network architectures

LO3 Design a cloud network infrastructure solution for an identified scenario

LO4 Develop a cloud network infrastructure solution for an identified scenario.

## Essential Content

LO1 **Explain the fundamental concepts of cloud networking**

*Cloud networking models and responsibilities:*

Service delivery models: cloud networking e.g. network services, cloud infrastructure; service delivery e.g. Software as a Service (SaaS) platforms, application services; data delivery e.g. data transfer, storage services

Deployment models: public deployment e.g. public cloud, shared platforms; private deployment e.g. private cloud, enterprise systems; hybrid deployment e.g. hybrid cloud, integrated environments

Network control functions: addressing e.g. IP addresses, subnetting, routing tables; routing e.g. route paths, traffic direction; security e.g. network protection, access control.

*Virtual network architecture and segmentation:*

Network constructs: virtual networks e.g. virtual private clouds (VPCs), Azure Virtual Networks (VNets); VPC constructs e.g. subnets, routing domains; VNet constructs e.g. address spaces, subnet groups

Subnet structures: subnets e.g. IP segments, address partitions; public subnets e.g. internet access, exposed services; private subnets e.g. internal systems, restricted access

Isolation methods: regional isolation e.g. geographic regions, location separation; network segmentation e.g. subnet isolation, traffic separation, segment control; workload isolation e.g. application isolation, service isolation.

*IP addressing and naming:*

Addressing standards: IPv4 addressing e.g. IP addresses, dotted notation; Classless Inter-Domain Routing (CIDR) notation e.g. network prefixes, subnet masks; private address ranges e.g. internal IPs, reserved ranges

Address planning: subnet sizing e.g. subnet capacity, host allocation; address allocation e.g. IP assignment, address distribution; address planning e.g. network design, allocation strategies

Integration and resolution: Domain Name System (DNS) integration e.g. domain systems, DNS records; internal DNS resolution e.g. service lookup, name resolution; hybrid integration addressing e.g. hybrid networks, cloud integration.

*Routing and traffic behaviour:*

Routing configuration: routing tables e.g. route entries, path selection; default routes e.g. gateway routes, fallback paths; custom routes e.g. user-defined routes, routing rules

Traffic flows: traffic flow e.g. data movement, packet flow; north-south traffic e.g. internet traffic, inbound connections; east –west traffic e.g. internal communication, service traffic

Connectivity components: gateways e.g. network gateways, connection points; Network Address Translation (NAT) gateway e.g. outbound access, address translation; peering e.g. direct links, network connections.

*Network security mechanisms:*

Access management: security groups e.g. access rules, security policies; access control lists (ACLs) e.g. filtering rules, control lists; identity and access management (IAM) roles e.g. identity roles, access roles

Filtering controls: stateful filtering e.g. connection tracking, session control; stateless filtering e.g. packet filtering, rule matching; ingress control e.g. inbound access, entry filtering

Security zoning: network zones e.g. network segments, security areas; perimeter zones e.g. boundary networks, edge zones; restricted zones e.g. secure areas, protected environments

Protection mechanisms: Distributed Denial of Service (DDoS) protection e.g. attack mitigation, traffic filtering; exposure mitigation e.g. risk reduction, vulnerability control; identity-aware networking e.g. authentication systems, user context.

LO2 **Explore a range of cloud-based network architectures**

*Cloud network architecture patterns:*

Architecture models: architectural patterns e.g. hub-and-spoke topology, mesh topology, hybrid topology; cloud networking e.g. virtual networks, subnets, routing paths; network structures e.g. centralised networking, distributed networking, multi-region deployment

Deployment approaches: single-region deployment e.g. single region, localised deployment; multi-region deployment e.g. multiple regions, geographic distribution; hub-and-spoke topology e.g. central hub, spoke networks

Network organisation: mesh connectivity e.g. full connectivity, peer links; centralised networking e.g. central control, shared services; distributed networking e.g. decentralised design, service distribution.

*Connectivity models:*

Connection types: site-to-site virtual private network (VPN) e.g. secure tunnels, remote networks; client-to-site VPN e.g. user connections, remote access; dedicated private connectivity e.g. leased lines, private links

Access methods: internet-based access e.g. public access, web access; hybrid networking support e.g. hybrid access, mixed connectivity; cloud service access e.g. platform access, service endpoints

Performance factors: latency e.g. response time, delay; bandwidth predictability e.g. consistent throughput, stable capacity; cost e.g. pricing models, usage costs.

*Hybrid and inter-network integration:*

Integration models: cloud integration e.g. cloud platforms, service linking; on-premises integration e.g. local systems, enterprise networks; inter-network communication e.g. network links, system communication

Connectivity design: peering e.g. direct connections, network links; hub-based transit e.g. central routing, hub networks; cross-region links e.g. region connections, geographic links

Resilience planning: failover e.g. backup systems, system switching; redundancy e.g. duplicate systems, backup resources; service continuity e.g. continuous service, uptime support.

*Architectural characteristics and trade-offs:*

Performance characteristics: scalability e.g. resource scaling, system growth; availability e.g. uptime, service access; latency e.g. response time, delay

Reliability factors: resilience e.g. fault tolerance, system recovery; throughput e.g. data rate, transfer capacity; performance e.g. system efficiency, speed

Design trade-offs: cost e.g. pricing, resource usage; flexibility e.g. adaptable systems, configuration options; operational complexity e.g. system management, administrative effort.

LO3 **Design a cloud network infrastructure solution for an identified scenario**

*Requirements and design constraints:*

Requirements identification: business requirements e.g. organisational needs, service requirements; technical requirements e.g. system specifications, network constraints; user requirements e.g. user access, service needs; functional requirements e.g. system functions, service capabilities; non-functional requirements e.g. performance needs, reliability requirements

Operational considerations: availability e.g. uptime, service access; security e.g. data protection, access control; scalability e.g. system growth, resource scaling.

*Network topology and segmentation design:*

Topology design: network topology e.g. topology models, network layouts; topology selection e.g. design choices, topology options; segmented layouts e.g. network segments, structured layouts

Segmentation approaches: network zones e.g. zone divisions, security areas; public-facing zones e.g. exposed services, internet access; internal zones e.g. internal networks, protected areas

Workload separation: workload separation e.g. service isolation, application separation; functional separation e.g. service grouping, role separation; sensitivity-based separation e.g. data sensitivity, access levels.

*Addressing, subnetting and routing design:*

Address planning: address space planning e.g. address schemes, network design; CIDR notation e.g. network prefixes, subnet masks; subnet allocation e.g. IP segments, resource allocation

Routing design: routing paths e.g. traffic routes, path selection; internal traffic routing e.g. internal communication, service traffic; internet-bound routing e.g. external traffic, outbound connections

Traffic control: gateways placement e.g. gateway positioning, network access; route associations e.g. routing links, route mappings; communication flows e.g. data movement, traffic flow.

*Connectivity and service integration design:*

Connectivity methods: VPN e.g. secure tunnels, private connections; peering e.g. network links, direct connections; dedicated private links e.g. leased lines, private links

Service integration: service integration e.g. service linking, system integration; shared services e.g. central services, common resources; DNS services e.g. naming systems, domain services

Communication design: communication design e.g. system communication, service interaction; user endpoints e.g. devices, client systems; external systems e.g. third-party systems, remote services.

*Security and control design:*

Security implementation: security controls e.g. access control, protection measures; rule-based filtering e.g. filtering rules, access rules

Access control: administrative access restrictions e.g. admin access, privileged accounts; ingress control e.g. inbound access, entry filtering; egress control e.g. outbound access, exit filtering

Operational controls: monitoring e.g. system monitoring, activity tracking; logging e.g. log records, audit logs; audit support e.g. compliance checks, audit trails.

*Design representation:*

Design artefacts: logical diagrams e.g. network diagrams, topology diagrams; subnet plans e.g. address plans, subnet layouts; route mappings e.g. routing maps, path diagrams

Design documentation: component descriptions e.g. system components, network elements; network boundaries e.g. boundary definitions, segment limits; connectivity paths e.g. network links, communication paths

Design justification: addressing decisions e.g. addressing choices, allocation decisions; security zones e.g. network zones, protection areas; requirements alignment e.g. requirement matching, design alignment.

## LO4 **Develop a cloud network infrastructure solution for an identified scenario**

*Provisioning and configuration:*

Provisioning: provisioning e.g. resource creation, service setup; virtual networks e.g. VPCs, VNets; subnets e.g. IP segments, subnet groups

Configuration management: configuration e.g. system settings, network setup; routing configuration e.g. route tables, traffic routing; connectivity associations e.g. network links, connection setup

Network structuring: network segments e.g. subnet divisions, network zones; environments e.g. development environments, production environments; component dependencies e.g. service links, system dependencies.

*Validation of connectivity and routing:*

Connectivity validation: connectivity validation e.g. connection checks, network tests; network segments e.g. subnet zones, traffic areas; workloads e.g. applications, services

Routing validation: route verification e.g. route checks, path validation; route behaviour e.g. traffic direction, routing flow; gateway function e.g. gateway roles, connection points

Issue identification: configuration errors e.g. setup errors, misconfigurations; routing issues e.g. route failures, path errors; unintended exposure e.g. security gaps, access issues.

*Monitoring and operational visibility:*

Monitoring systems: network monitoring e.g. traffic monitoring, network tracking; activity monitoring e.g. system activity, usage tracking; metrics e.g. performance data, measurement values; installation and management e.g. alert configuration, threshold configuration, triggering; troubleshooting e.g. agent set up, fault diagnosis, resolution

Logging and records: logs e.g. system logs, event logs; flow logs e.g. traffic logs, connection logs; connection records e.g. session data, connection data; analysis e.g. pattern identification, unexpected behaviours; decision-making e.g. routing changes, operational changes

Operational insight: dashboards e.g. visual displays, monitoring panels; alerts e.g. warning signals, notifications; performance monitoring e.g. system performance, resource tracking.

*Performance and reliability evaluation:*

Performance measurement: performance evaluation e.g. system performance, service efficiency; responsiveness e.g. response time, system speed; throughput e.g. data transfer, processing rate

Reliability factors: availability e.g. uptime, system access; service continuity e.g. continuous service, operational stability; resilience measures e.g. fault tolerance, recovery mechanisms

Issue analysis: bottlenecks e.g. performance issues, traffic delays; inefficient routes e.g. poor routing, suboptimal paths; single points of failure e.g. failure risks, critical components.

*Security and operational effectiveness:*

Security evaluation: security effectiveness e.g. system security, protection levels; segmentation effectiveness e.g. network segmentation, isolation control; filtering effectiveness e.g. traffic filtering, access control

Operational control: access control effectiveness e.g. permission control, user access; service protection e.g. resource protection, system security; communication control e.g. traffic control, data flow

Operational management: monitoring coverage e.g. monitoring scope, visibility; troubleshooting e.g. issue resolution, fault finding; risks e.g. operational risks, security risks.

*Optimisation and improvement:*

Performance optimisation: performance improvement e.g. system tuning, efficiency gains; scalability improvement e.g. system scaling, resource expansion; operational efficiency e.g. process efficiency, system optimisation

Infrastructure refinement: route refinement e.g. route tuning, path optimisation; subnet restructuring e.g. subnet redesign, network layout changes; redundancy increase e.g. backup systems, duplicate resources

Change management: connectivity changes e.g. connection updates, network adjustments; security improvement e.g. enhanced protection, security updates; improvement justification e.g. change reasoning, decision support.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the fundamental concepts of cloud networking</td><td rowspan="4">LO1 and LO2<br><br>D1 Evaluate, using a selection of real world examples, the effectiveness of cloud-based network architectures and concepts in delivering high performance modern business networks.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the concepts of cloud networking, virtual architectures and hybrid models in organisational contexts.<br><br>P2 Discuss, using a selection of real-world examples, the principles of addressing, routing and network security controls in cloud environments.</td><td>M1 Analyse, using a selection of real-world examples, the importance of cloud networking concepts in delivering effective organisational networks.</td></tr>
<tr><td colspan="2">LO2 Explore a range of cloud-based network architectures</td></tr>
<tr><td>P3 Explore, using a selection of real-world examples, the application of cloud-based network architectures and models in different organisational contexts, including hybrids.<br><br>P4 Describe, using a selection of real-world examples, the performance characteristics and trade-offs of a range of network architectures and models in different organisational contexts.</td><td>M2 Assess, using a selection of real-world examples, the suitability of cloud network architecture patterns and connectivity models for different organisational contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a cloud network infrastructure solution for an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of a cloud network infrastructure solution to allow the comprehensive delivery of the requirements for an identified scenario. |
| **P5** Design a cloud network infrastructure design that meets the routing, segmentation and security requirements of a defined scenario.<br>**P6** Review the design of the cloud network infrastructure solution with a range of identified internal stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the stakeholder feedback, the design of the cloud–based model and architecture for an identified business scenario. | |
| **LO4** Develop a cloud network infrastructure solution for an identified scenario | | |
| **P7** Develop a cloud network infrastructure solution that meets the requirements of an identified scenario.<br>**P8** Review, using a developed test plan and range of monitoring tools, the connectivity and performance of the developed cloud network infrastructure solution identifying areas for optimisation and improvement. | **M4** Evaluate the effectiveness of the cloud network infrastructure solution in meeting the requirements for an identified scenario. | |

## Recommended Resources

### Textbooks

Dutt, D., Hilt, V. and Maldonado, A. (2023) *Cloud Native Data Center Networking: Architecture, Protocols, and Tools*. Sebastopol, CA: O'Reilly Media.

Rice, L. (2024) Learning eBPF: *Programming the Linux Kernel for Enhanced Observability, Networking, and Security*. Sebastopol, CA: O'Reilly Media.

Goniwada, S.R. (2022) *Cloud Native Architecture and Design*. Berkeley, CA: Apress.

Gupta, B.B. *et al.* (2024) SDN and NFV: *A New Dimension to Virtualization*. Singapore: World Scientific.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| docs.aws.amazon.com/vpc/latest/userguide/how-it-works.html | Amazon Web Services (AWS)<br>Explanation of How Virtual Private Cloud Networking Works, Including Subnets, Routing and Internet Access<br>(General reference) |
| www.edureka.co/blog/azure-virtual-network-tutorial/ | Edureka<br>Beginner-Friendly Guide to Azure Virtual Networks, Including Subnets, Security Groups and Cloud Communication Models<br>(Blog) |
| learn.microsoft.com/en-us/azure/virtual-network/quickstart-create-virtual-network | Microsoft Learn<br>Step-by-step tutorial on creating and testing a virtual network with routing and connectivity in Azure<br>(Tutorial) |
| www.geeksforgeeks.org/devops/google-cloud-vpc-virtual-private-cloud/ | GeeksforGeeks<br>Overview of Google Cloud VPC architecture including subnets, routing, firewall rules and connectivity components<br>(General reference) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.devopsschool.com/tutorials/google-cloud-virtual-private-cloud-vpc-tutorial-architecture-pricing-use-cases-and-hands-on-guide-for-networking/ | DevOpsSchool<br>Comprehensive tutorial on Google Cloud VPC architecture, routing, security controls and hybrid connectivity<br>(Tutorial) |
| cloudwebschool.com/docs/gcp/networking/vpc-networks-explained/ | CloudWebSchool<br>Explanation of VPC networking concepts including IP addressing, segmentation and traffic routing in cloud environments<br>(General reference) |
| learn.microsoft.com/en-us/azure/virtual-wan/virtual-wan-site-to-site-portal | Microsoft Learn<br>Tutorial on Configuring Site-to-Site VPN Connectivity and Routing Between Networks in Cloud Environments<br>(Tutorial) |
| zeronetworks.com/blog/cloud-network-segmentation-for-enhanced-security-guide | Zero Networks<br>Article Explaining Cloud Network Segmentation and Its Role in Security, Isolation and Performance Optimisation<br>(Article) |

### Journals and articles

Farchi, E. and Route, S. (2023) 'Quality engineering for Agile and DevOps on the cloud and edge', *arXiv*. Available at: doi.org/10.48550/arXiv.2302.03651.

Ma, T., Chen, Y., Wang, H., Xiong, X. and Zhao, J. (2023) 'Research on evaluation index method of cloud-network convergence capability', *Scientific Reports*, 13, Article 20286. Available at: doi.org/10.1038/s41598-023-47626-3.

Rosa, L., Foschini, L. and Corradi, A. (2024) 'Empowering cloud computing with network acceleration: A survey', *IEEE Communications Surveys & Tutorials*, 26(4), pp. 2729–2768. Available at: doi.org/10.1109/COMST.2024.3377531.

Silva, F.A., Trinta, F.A.M., Bonfim, M.S., de Macedo, J.A.F., Rego, P.A.L. and Lagrota, V. (2025) 'Performance evaluation of cloud native applications: A systematic mapping study', *Journal of Network and Systems Management*, 33, Article 78. Available at: doi.org/10.1007/s10922-025-09937-w.

### Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 416: Core Cloud Computing*

*Unit 417: Cloud DevOps Automation*

*Unit 419: Cloud Security.*

 © Pearson Education Limited 2026

# Unit 419: Cloud Security

**Unit code:** **Y/652/3271**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

Cloud computing plays a central role in modern IT systems, supporting the delivery of services, storage of data and operation of applications across a wide range of sectors. As organisations increasingly adopt cloud-based solutions, the need to protect systems, networks and information within these environments has become essential. Cloud security focuses on safeguarding digital assets while maintaining reliable and trusted services in environments that are scalable, shared and constantly evolving.

The aim of this unit is to develop students' understanding of how cloud environments are secured in practice. Students will explore the key principles that underpin cloud security, including shared responsibility models, identity and access management and data protection. The unit is designed to develop both theoretical knowledge and practical awareness of how security controls are applied to manage risks and meet organisational and regulatory requirements.

The unit covers a range of topics that reflect the core areas of cloud security. Students will examine how data is protected both at rest and in transit, alongside the technologies used to support encryption, monitoring and threat detection. The role of secure networking, infrastructure controls and access management will also be explored. In addition, students will design and implement a secure cloud environment for a given scenario, bringing together their knowledge of security principles, compliance requirements and system design.

On successful completion of this unit, students will be able to explain key cloud security concepts, evaluate security techniques and technologies, and design and implement a secure cloud-based solution. These skills support progression to further study in cloud computing, cybersecurity and IT infrastructure, as well as entry-level roles such as Cloud Support Technician, Junior Cloud Engineer, Security Operations Analyst, Infrastructure Technician and Cybersecurity Support Analyst.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the theoretical principles of cloud security

LO2 Review the techniques and technologies used to secure data stored within and transmitted between cloud environments

LO3 Design a secure, regulation-compliant cloud environment for an identified scenario

LO4 Implement a secure, regulation-compliant cloud environment for an identified scenario.

## Essential Content

LO1 **Explain the theoretical principles of cloud security**

*Cloud security principles and shared responsibility:*

Service models: Infrastructure as a Service (IaaS) e.g. virtual machines, storage services; Platform as a Service (PaaS) e.g. application platforms, development environments; Software as a Service (SaaS) e.g. web applications, email services

Responsibility models: provider-managed controls e.g. infrastructure management, platform maintenance; customer-managed controls e.g. data protection, access management; shared security responsibilities e.g. configuration, monitoring; accountability e.g. compliance ownership, risk ownership

Security domains and architecture concepts: infrastructure security e.g. physical security, virtual machines; platform protection e.g. runtime environments, middleware; application protection e.g. secure coding, vulnerabilities; data protection e.g. cryptographic protection, access control; multi-tenancy e.g. tenant isolation, shared resources; shared infrastructure e.g. data centres, network resources.

*Identity and access management:*

Authentication: authentication mechanisms e.g. passwords, tokens; multi-factor authentication (MFA) e.g. SMS verification, authenticator apps; identity federation e.g. single sign-on, external identity providers

Access control: role-based access control e.g. user roles, permission sets; privilege management e.g. admin rights, access levels; identity life cycle management e.g. user provisioning, deprovisioning

Cloud context: cloud environments e.g. public cloud, private cloud; identity life cycle management e.g. onboarding, offboarding; role-based access control e.g. grouped permissions, defined roles.

*Data protection concepts:*

Data states: stored data e.g. files, databases; transmitted data e.g. network traffic, data transfer; active data processing e.g. in-memory operations

Data governance: classification e.g. confidential, public; integrity e.g. accuracy, consistency; availability e.g. uptime, recovery readiness

Cryptographic protection: methods e.g. symmetric and asymmetric encryption; key life cycle management e.g. rotation, secure storage; safeguarded storage practices e.g. encrypted repositories, controlled access.

*Network and infrastructure security:*

Traffic control: network filtering e.g. packet inspection, rule-based filtering; service exposure restriction e.g. port and service limitations

Segmentation and protection: network partitioning e.g. subnet separation, zones; workload isolation e.g. container and virtual machine isolation; cloud firewall controls e.g. rule sets, traffic inspection, threat prevention

Infrastructure protection: security groups e.g. rule sets, access policies; virtual firewalls e.g. traffic inspection, threat prevention; exposure control of services within cloud networks e.g. controlled access, limited exposure.

*Advanced cloud security:*

Secure Access Service Edge (SASE): core components e.g. Software-Defined WAN (SD-WAN), Cloud Access Security Broker (CASB), Secure Web Gateway (SWG), Firewall-as-a-Service (FWaaS); benefits and drawbacks

Zero-Trust Networking (ZTN): core principles e.g. least privilege access, micro-segmentation, continuous verification, Identity Access Management (IAM); technologies e.g. identity providers, Zero Trust Network Access (ZTNA) Gateways, endpoint compliance tools, micro-segmentation platforms

Integration of SASE and ZTN with change management e.g. ZTNA access rules, SD-WAN routing policies, micro-segmentation zone changes, new identity provider settings, endpoint compliance requirements, FWaaS rules.

LO2 **Review the techniques and technologies used to secure data stored within and transmitted between cloud environments**

*Data classification and handling:*

Data classification: data classification e.g. data labels, cloud categories; sensitivity levels e.g. confidential levels, restricted levels, internal levels;

Data governance: regulatory requirements e.g. General Data Protection Regulation (GDPR) rules, data protection laws; data ownership e.g. data owners, cloud account owners, system owners;

Data categories: public data e.g. public websites, open data; confidential data e.g. restricted files, private records, sensitive documents.

*Protection of data stored in cloud environments:*

Encryption: data-at-rest encryption e.g. encrypted storage, encrypted disks; database encryption e.g. encrypted databases, protected tables, secure records;

Storage security: object storage security e.g. secure buckets, bucket permissions; secure backup storage e.g. cloud backups, backup storage, recovery storage;

Access and configuration: storage access restrictions e.g. access permissions, role restrictions; secure configuration e.g. secure settings, configuration controls, system settings.

*Protection of data transmitted between cloud environments:*

Encryption protocols: data-in-transit encryption e.g. secure transfer, encrypted traffic; Hypertext Transfer Protocol Secure (HTTPS) e.g. secure websites, encrypted web traffic, safe browsing;

Secure connectivity: secure Application Programming Interfaces (APIs) e.g. API security, authenticated APIs; VPN technologies e.g. VPN tunnels, private networks, secure connections;

Data transfer control: data transfer protection e.g. protected transfers, secure movement; external systems communication e.g. third-party connections, external integrations, partner systems.

*Encryption and key management technologies:*

Encryption types: symmetric encryption e.g. shared key encryption, simple encryption; asymmetric encryption e.g. public key encryption, private key encryption, key-based security

Key infrastructure: certificate management e.g. certificate storage, certificate updates; cloud key management services e.g. managed keys, cloud key storage, key services

Key control: customer-managed keys e.g. user-managed keys, private keys; provider-managed keys e.g. cloud keys, system keys, managed keys.

*Data masking, tokenisation and loss prevention:*

Data transformation: data masking e.g. hidden values, masked data; tokenisation e.g. token replacement, generated tokens, secure tokens

Loss prevention: data loss prevention (DLP) tools e.g. DLP tools, monitoring tools; unauthorised data movement e.g. data leaks, unauthorised transfers, data exposure

Protection controls: exposure reduction controls e.g. limited access, restricted visibility; sensitive data protection e.g. protected data, secured information, controlled access.

*Backup, recovery and data retention controls:*

Backup strategies: backup strategies e.g. full backups, incremental backups; replication e.g. data copies, cloud replication, system copies

Recovery planning: recovery testing e.g. test restores, system checks; recovery time considerations e.g. recovery targets, downtime limits, service recovery

Retention management: retention policies e.g. data rules, storage periods; archive storage e.g. long-term storage, archive systems, storage archives.

*Access control and monitoring for data protection:*

Access control: storage access policies e.g. access rules, permission settings; identity-based access e.g. user identities, access roles, role assignments;

Monitoring: audit trails e.g. activity logs, access logs; alerts e.g. security alerts, warning notifications, system alerts

Threat detection: failed access attempts e.g. login failures, access denials; unauthorised changes e.g. unauthorised edits, configuration changes, system changes.

*Compliance controls for stored and transmitted data:*

Regulatory requirements: data protection regulations e.g. GDPR, data protection laws; data residency e.g. data locations, regional storage, location rules; knowledge currency, innovation, changing practice

Control requirements: encryption requirements e.g. required encryption, security policies; access review requirements e.g. access checks, permission reviews, user audits

Protection scope: stored data protection e.g. secure storage, protected data; transmitted data protection e.g. secure transfer, protected communication, encrypted transmission.

LO3 **Design a secure, regulation-compliant cloud environment for an identified scenario**

*Security requirements and constraints:*

Security requirements: functional security requirements e.g. access needs, data protection needs; non-functional security requirements e.g. availability targets, performance limits, reliability needs

Core principles: confidentiality e.g. restricted data access, private information; integrity e.g. accurate records, protected data; availability e.g. system uptime, service access

Compliance context: compliance requirements e.g. regulations, standards; organisational policies e.g. security policies, access policies.

*Identity and access design:*

Access structure: user roles e.g. admin roles, user roles; access control structures e.g. role assignments, permission groups, access levels

Authentication design: authentication mechanisms e.g. passwords, tokens; identity provider integration e.g. identity services, cloud identity providers

Access principles: least privilege implementation e.g. minimal access, restricted permissions; secure access pathways e.g. secure login, controlled access routes.

*Data security design:*

Data planning: data classification e.g. data labels, sensitivity levels; data handling e.g. access rules, storage rules

Protection planning: encryption strategies e.g. encryption use, encryption selection; key management approaches e.g. key control, key storage

Storage design: secure storage design e.g. protected storage, secure data locations; sensitive data protection e.g. restricted data, protected records.

*Network security design:*

Segmentation design: network segmentation e.g. network zones, subnet separation; firewalls placement e.g. boundary controls, network points

Traffic control design: inbound traffic control e.g. incoming rules, filtering rules; outbound traffic control e.g. outgoing rules, data restrictions

Exposure design: service exposure control e.g. limited services, restricted endpoints; external threat protection e.g. threat protection, network defence.

*Security architecture integration:*

Security integration: identity security integration e.g. integrated access control, identity systems; network security integration e.g. network controls, layered protection

Data integration: data security integration e.g. protection layers, data controls; unified security design e.g. combined controls, integrated systems

System co-ordination: service model alignment e.g. cloud models, service layers; system components integration e.g. connected services, system components.

*Design documentation:*

Documentation: security architecture diagrams e.g. architecture diagrams, network diagrams; control mapping e.g. control placement, control alignment

Policy definition: policies e.g. security policies, access policies; design justification e.g. design decisions, security choices

Requirements alignment: alignment e.g. requirement matching, control alignment.

LO4 **Implement a secure, regulation-compliant cloud environment for an identified scenario**

*Implementation and configuration review:*

Configuration: configuration e.g. system settings, service setup; identity security controls e.g. user controls, access settings, login controls

Security controls: network security controls e.g. network rules, firewall settings; data security controls e.g. data protection, encryption settings, storage security

Verification: implementation verification e.g. system checks, setup validation; configuration weaknesses e.g. weak settings, misconfigurations.

*Security validation and testing:*

Testing activities: access testing e.g. user login tests, permission checks; vulnerability identification e.g. security gaps, weak points, exposed areas; penetration testing e.g. basic security testing, ethical hacking, simulated attacks

Validation: encryption validation e.g. encryption checks, secure data validation; protection mechanisms e.g. security controls, defence measures, protection tools

Control testing: control verification e.g. control checks, system validation; control effectiveness e.g. effective controls, working protections.

*Monitoring and threat detection:*

Monitoring systems: monitoring systems e.g. monitoring tools, system monitoring; logs e.g. system logs, activity logs, access logs

Monitoring operations: installation e.g. tool setup, agent installation; configuration e.g. alert settings, monitoring rules, system configuration;

Monitoring analysis: interpretation e.g. log review, alert understanding; analysis e.g. pattern identification, issue investigation, trend analysis

Alerting: alerts e.g. warning alerts, security alerts; thresholds e.g. limit settings, alert levels

Threat detection: threat detection e.g. threat identification, risk detection; security events e.g. security incidents, system events, alert events.

*Performance and resilience evaluation:*

Performance: performance impact e.g. system impact, response times; system performance e.g. system speed, service performance, processing speed

Resilience: resilience evaluation e.g. system resilience, service stability; attack resilience e.g. attack resistance, system protection, defence strength

Recovery: recovery mechanisms effectiveness e.g. recovery processes, restore systems.

*Risk and compliance evaluation:*

Risk assessment: residual risk e.g. remaining risk, unmanaged risk; compliance assessment e.g. compliance checks, standards review, policy checks

Policy and standards: policies e.g. security policies, access policies; standards e.g. security standards, compliance standards, organisational standards

Security gaps: security gaps e.g. missing controls, weak protections; weaknesses e.g. system weaknesses, security issues.

*Improvement and optimisation:*

Security improvement: security posture e.g. overall security, system security; control enhancement e.g. improved controls, stronger protections, updated settings

Monitoring improvement: monitoring improvement e.g. better monitoring, improved visibility; response improvement e.g. faster response, improved actions

Alignment: optimisation e.g. system optimisation, resource optimisation; organisational requirements alignment e.g. business needs, security requirements, system goals.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the theoretical principles of cloud security</td><td rowspan="4">LO1 and LO2<br><br>D1 Evaluate, using a selection of real world examples, the effectiveness of cloud security principles and technologies in protecting cloud environments and data-at-rest and in-motion for an identified organisational context.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the theoretical principles of cloud security and access management.<br><br>P2 Discuss, using a selection of real-world examples, the principles of data and infrastructure security for cloud-based systems.</td><td>M1 Analyse, using a selection of real-world examples, the effectiveness of cloud security principles in the protection of organisational data and cloud infrastructure for a range of organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Review the techniques and technologies used to secure data stored within and transmitted between cloud environments</td></tr>
<tr><td>P3 Review, using a selection of real-world examples, the techniques used to protect data stored within cloud environments.<br><br>P4 Explore, using a selection of real-world examples, the technologies used to secure data transmitted between cloud environments.</td><td>M2 Assess, using a selection of real-world examples, the effectiveness of security technologies used to protect stored and transmitted cloud data.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a secure, regulation-compliant cloud environment for an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of a secure cloud environment that enables it to comprehensively meet the requirements and regulations for an identified scenario. |
| **P5** Design a secure, regulation-compliant cloud environment for an identified scenario.<br>**P6** Review the design of the secure, regulation-compliant cloud environment with a range of identified stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the stakeholder feedback, the design of the secure, regulation-compliant cloud environment for an identified scenario. | |
| **LO4** Implement a secure, regulation-compliant cloud environment for an identified scenario | | |
| **P7** Implement a secure, regulation-compliant cloud environment for an identified scenario.<br>**P8** Review, using a developed test plan and range of threat monitoring tools, the security, performance and resilience of the developed secure cloud environment identifying areas for optimisation and improvement. | **M4** Evaluate the effectiveness of the secure cloud environment in meeting the requirements for an identified scenario and complying with required regulations. | |

## Recommended Resources

### Textbooks

Easttom, W. (2023) *Computer Security Fundamentals*. 5th edn. Indianapolis: Pearson IT Certification.

Estrin, E. (2022) *Cloud Security Handbook*. Birmingham: Packt Publishing.

Kim, D. and Solomon, M.G. (2021) *Fundamentals of Information Systems Security*. 4th edn. Burlington, MA: Jones & Bartlett Learning.

Pfleeger, C.P., Pfleeger, S.L. and Coles-Kemp, L. (2023) *Security in Computing*. 6th edn. Boston, MA: Addison-Wesley.

Stallings, W. (2023) *Cryptography and Network Security: Principles and Practice*. 8th edn. Harlow: Pearson.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/devops/identity-and-access-management/ | GeeksforGeeks<br>Identity and Access Management Concepts, Authentication and Authorisation in Cloud Systems<br>(Tutorials) |
| www.datacamp.com/tutorial/aws-identity-and-access-management-iam-guide | DataCamp<br>AWS IAM Overview, Roles, Policies and Access Control Mechanisms in Cloud Environments<br>(Tutorials) |
| www.cloudflare.com/learning/security/glossary/what-is-zero-trust/ | Cloudflare<br>Zero Trust Security Model, Principles of Least Privilege and Continuous Verification<br>(General reference) |
| phoenixnap.com/blog/data-in-transit-encryption | phoenixNAP<br>Encryption in Transit, Risks to Moving Data and Protection Using Secure Protocols<br>(Article) |

 © Pearson Education Limited 2026

| | |
|---|---|
| learn.microsoft.com/en-us/compliance/assurance/assurance-encryption-in-transit | Microsoft Learn<br>Encryption Technologies for Protecting Cloud Data in Transit Using TLS and IPsec<br>(General reference) |
| www.digitalocean.com/resources/articles/cloud-architecture | DigitalOcean<br>Cloud Architecture Fundamentals Including Front-End and Back-End Components and Design Concepts<br>(Article) |
| www.geeksforgeeks.org/cloud-computing/architecture-of-cloud-computing/ | GeeksforGeeks<br>Structure of Cloud Computing Architecture Including Infrastructure, Networking and Security Layers<br>(Tutorials) |
| clouddefenseauthority.com/cloud-siem-and-logging | Cloud Defense Authority<br>Cloud SIEM, logging and monitoring practices for threat detection and security operations<br>(General reference) |
| www.databank.com/resources/blogs/what-is-cloud-backup-disaster-recovery-best-practices-solutions/ | DataBank<br>Cloud Backup and Disaster Recovery Concepts Including RTO, RPO and Business Continuity<br>(Article) |

**Journals and articles**

Ali, M., Khan, S.U. and Vasilakos, A.V. (2015) 'Security in cloud computing: Opportunities and challenges', *Information Sciences*, 305, pp. 357–383. Available at: doi.org/10.1016/j.ins.2015.01.025.

Zissis, D. and Lekkas, D. (2012) 'Addressing cloud computing security issues', *Future Generation Computer Systems*, 28(3), pp. 583–592. Available at: doi.org/10.1016/j.future.2010.12.006.

Almorsy, M., Grundy, J. and Müller, I. (2010) 'An analysis of the cloud computing security problem', in *Proceedings of the APSEC 2010 Cloud Workshop*, Sydney, Australia, 30 November. Available at: doi.org/10.48550/arXiv.1609.01107.

Hashizume, K., Rosado, D.G., Fernández-Medina, E. and Fernandez, E.B. (2013) 'An analysis of security issues for cloud computing', *Journal of Internet Services and Applications*, 4(1), Article 5. Available at: doi.org/10.1186/1869-0238-4-5.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 416: Core Cloud Computing*

*Unit 417: Cloud DevOps Automation*

*Unit 418: Cloud Networking.*

# Unit 420: Core AI

**Unit code:** **A/652/3272**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

Artificial Intelligence (AI) has become a central part of modern digital systems, influencing how organisations analyse data, automate processes and support decision-making. From everyday software tools to large-scale business applications, AI technologies are increasingly embedded in products and services across many sectors. Understanding the foundations of AI is now an essential part of working in computing and digital technologies, as students need to recognise how these systems are designed, how they operate and where they can be effectively used.

The aim of this unit is to give students a clear and practical introduction to the core ideas that underpin AI. It focuses on the key terminology, concepts and processes that form the basis of intelligent systems and machine learning models. The unit is designed to build confidence at an introductory level, helping students understand how AI approaches differ from traditional solutions and why organisations choose AI-enabled methods for particular tasks.

The unit covers a broad range of essential content. Students will explore the principles behind intelligent agents, problem-solving methods and heuristic search, as well as the main categories of machine learning and how data is prepared, trained and evaluated. They will also learn how simple AI-driven solutions can be developed to meet defined requirements, and how these can be compared to non-AI alternatives to judge their effectiveness and limitations.

On successful completion of this unit, students will be able to explain core AI concepts, describe machine learning categories and processes, develop a basic AI-driven solution and evaluate how well it performs in relation to a traditional approach. The knowledge gained in this unit supports progression into more specialist areas of computing and prepares students for early-stage roles such as Data Technician, Machine Learning Assistant, AI Support Specialist, Junior Data Analyst, Automation Technician or Software Testing Assistant.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Define core terminology and principles used in the field of Artificial Intelligence (AI)

LO2 Describe the fundamental machine learning categories and processes

LO3 Develop a simple AI-driven data solution using appropriate models and techniques for a specified scenario

LO4 Review the effectiveness of the AI-driven solution for a specified scenario against a non-AI solution.

## Essential Content

LO1 **Define core terminology and principles used in the field of Artificial Intelligence (AI)**

*AI terminology and definitions:*

AI core concepts: artificial intelligence e.g. computer intelligence, automated behaviour; narrow AI e.g. focused tasks, specific goals; intelligent systems e.g. rule systems, simple learning systems

Agent fundamentals: agent e.g. software unit, autonomous entity; environment e.g. surroundings, system context; state e.g. current condition, situation data

Behaviour and interaction: perception e.g. data input, sensory signals; action e.g. response execution, task step; goal e.g. target outcome, expected state;

Knowledge and representation: knowledge e.g. stored information, learned patterns; representations e.g. symbols, structured data.

*Rational agents and problem formulation:*

Agent types and structure: rationality e.g. sensible choices, goal-directed behaviour; simple reflex agents e.g. condition–action rules, direct response; model-based agents e.g. stored information, system model

Problem modelling basics: state space e.g. possible states, configuration set; initial state e.g. starting position, baseline condition; goal test e.g. success check, outcome match

Problem-solving actions: operators e.g. allowed actions, transformations; tasks e.g. objectives, requirements; transitions e.g. movements, changes.

*Search methods and heuristic concepts:*

Search approaches: uninformed search e.g. blind exploration, step-by-step checking; informed search e.g. guided search, estimated direction; tree search e.g. branching paths, hierarchical choices

Search states and movement: search nodes e.g. positions, state markers; search paths e.g. routes, state sequences; solution steps e.g. actions, moves

Search tracking: frontier e.g. open list, active nodes; explored set e.g. visited states, checked nodes

Heuristic principles: heuristic function e.g. score value, simple estimate; guidance value e.g. direction cue, priority hint

*Real-world examples:*

A range of real-world AI systems e.g.

AI assistants: Siri e.g. voice queries, reminders; Alexa e.g. smart home control, music; Copilot e.g. document support, coding help

Intelligent agents: Nest Thermostat e.g. automatic temperature control; Roomba e.g. autonomous cleaning; Google Assistant e.g. task automation, scheduling

Navigation systems: Google Maps e.g. route planning, traffic updates; Waze e.g. crowd-sourced routes, hazard alerts; Apple Maps e.g. navigation, route guidance

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Describe the fundamental machine learning categories and processes**

*Machine learning categories:*

Learning types: supervised learning e.g. labelled data, training examples; unsupervised learning e.g. pattern groups, data clusters, hidden structure; reinforcement learning e.g. rewards, actions, outcomes

Task categories: classification e.g. category labels, class outputs; regression e.g. value prediction, numeric output, simple estimates; clustering e.g. group patterns, data segments.

*Data sets and representation:*

Data types: structured data e.g. tables, fields; unstructured data e.g. text, images, documents; mixed data e.g. combined sources, varied formats

Data splits: training data e.g. learning examples, labelled sets; validation data e.g. tuning sets, performance checks; test data e.g. unseen samples, evaluation sets;

Data elements: features e.g. attributes, input variables; labels e.g. class names, target categories; target variables e.g. outcome values, result fields

Data quality: missing data e.g. gaps, absent values; inconsistency e.g. mismatched entries, irregular values; noise e.g. random variation, unwanted signals.

*Model development processes:*

Model setup: problem definition e.g. task goal, required output; baseline models e.g. simple predictor, starting approach; model selection e.g. best fit, suitable choice

Training and validation: training e.g. pattern learning, model adjustment, repeated examples; validation e.g. tuning checks, performance comparison; overfitting e.g. memorised patterns, poor generalisation

Performance assessment: accuracy e.g. correct predictions, match rate; precision e.g. correct positives, relevant hits; recall e.g. captured cases, found positives

*Examples of real-world machine learning applications:*

A range of real-world ML tools e.g.

Amazon Recommendations e.g. personalised suggestions, purchase prediction; Google News grouping e.g. article clustering, topic grouping; DeepMind Atari Agents e.g. game learning, decision making

Machine learning platforms: Google Vertex AI e.g. model training, deployment; Azure Machine Learning e.g. data preparation, model evaluation; Amazon SageMaker e.g. model training, data labelling

Model development tools: Google Colab e.g. coding notebooks, experimentation; Azure ML Studio e.g. visual models, pipeline setup; SageMaker Studio e.g. development environment, model testing

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Develop a simple AI-driven data solution using appropriate models and techniques for a specified scenario**

*Scenario requirements and solution scope:*

Scenario elements: scenario context e.g. problem setting, user situation; inputs e.g. data sources, required information; outputs e.g. results, predictions, classifications

Constraints and assumptions: constraints e.g. limits, conditions; assumptions e.g. expected conditions, basic expectations; acceptance criteria e.g. success measures, required outcomes

User and stakeholder needs: stakeholders e.g. users, clients; user needs e.g. goals, requirements; success measures e.g. performance targets, outcome goals.

*Data processing techniques and quality considerations:*

Data sources and types: data sources e.g. files, simple datasets; data types e.g. numerical values, text data; data quality e.g. completeness, accuracy

Preparation and cleaning: data collection e.g. gathering records, capturing samples; data cleaning e.g. removing errors, fixing missing values; preprocessing e.g. formatting data, preparing inputs

Feature handling: features e.g. attributes, variables; feature selection e.g. important fields, chosen inputs; labelling e.g. category names, target values.

*Model and workflow design elements:*

Basic model design: model selection e.g. suitable approach, simple model choice; algorithm suitability e.g. task match, appropriate method; workflow setup e.g. steps, process order

Training and testing: training process e.g. pattern learning, repeated examples; validation approach e.g. checking performance, tuning checks; testing approach e.g. final evaluation, unseen data

Evaluation and outputs: performance measurement e.g. accuracy, correct results; documentation e.g. notes, records; outputs e.g. results, interpretations

*Examples of real-world AI applications:*

A range of real-world AI tools e.g.

AI applications: Salesforce Einstein e.g. case routing, predictions; Zendesk AI e.g. ticket sorting, chatbots; Freshdesk Freddy e.g. query handling, automation

Data preparation tools: AWS DataBrew e.g. data cleaning, transformation; Google Dataprep e.g. data formatting, anomaly detection; Azure Data Wrangler e.g. data shaping, feature preparation

Model training tools: Vertex AI Training e.g. model training, pipelines; Azure ML Pipelines e.g. workflow automation, model management; Kaggle Notebooks e.g. experimentation, dataset exploration

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Review the effectiveness of the AI-driven solution for a specified scenario against a non-AI solution**

*Non-AI solution baselines:*

Rule-based approaches: manual decision rules e.g. fixed logic, predefined actions; threshold methods e.g. cut-off points, decision limits; simple statistics e.g. averages, totals

Traditional methods: spreadsheet methods e.g. tables, formulas; qualitative judgement e.g. expert opinion, human assessment.

*Comparative effectiveness measures:*

Performance measures: performance metrics e.g. accuracy values, correct outputs; error rates e.g. mistakes, incorrect results; prediction quality e.g. reliable outputs, consistent results

Operational qualities: consistency e.g. stable behaviour, repeatable results; interpretability e.g. clear meaning, understandable output; maintainability e.g. updates, changes, adjustments.

*Test plan elements:*

Testing components: test cases e.g. sample inputs, trial examples; test data e.g. evaluation samples, scenario sets; expected results e.g. desired outputs, target answers

Verification and validation: validation checks e.g. outcome review, result confirmation; evaluation criteria e.g. scoring rules, success measures; test procedures e.g. step sequences, testing actions.

*Advantages, limitations, and contextual fit:*

Solution strengths: strengths e.g. benefits, useful features; suitability e.g. match to needs, correct fit; efficiency e.g. reduced effort, faster tasks

Solution limitations: limitations e.g. constraints, weaknesses; risks e.g. incorrect outputs, errors; data dependence e.g. data quality needs, input requirements

User factors: usability e.g. ease of use, clarity; human oversight e.g. supervision, review.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Define core terminology and principles used in the field of Artificial Intelligence (AI)</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a range of real-world examples where appropriate, the effectiveness of core Artificial Intelligence and Machine Learning principles and processes in the development of AI systems.</td></tr>
<tr><td>P1 Define, using a range of real-world examples where appropriate, the core Artificial Intelligence terminology and principles used in AI systems.<br><br>P2 Review, using a range of real-world examples where appropriate, the concepts of rational agents, problem formulations, heuristics and search methods used in AI systems.</td><td>M1 Analyse, using a range of real-world examples where appropriate, how core Artificial Intelligence terminology and principles are applied in AI systems.</td></tr>
<tr><td colspan="2">LO2 Describe the fundamental machine learning categories and processes</td></tr>
<tr><td>P3 Describe, using a range of real-world examples where appropriate, the fundamental machine learning categories used in AI systems.<br><br>P4 Discuss, using a range of real-world examples where appropriate, the use of datasets and model development processes used in AI systems.</td><td>M2 Analyse, using a range of real-world examples where appropriate, the application of machine learning categories and processes in the development of AI systems.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Develop a simple AI-driven data solution using appropriate models and techniques for a specified scenario | | **LO3 and LO4**<br>**D2** Justify, using the results of your testing and your observations of real-world examples, the selection of an AI-driven solution compared to a non-AI solution for a given scenario. |
| **P5** Design a simple AI-driven data solution using appropriate models and techniques for a specified scenario.<br>**P6** Develop a simple AI-driven data solution using appropriate models and techniques for a specified scenario. | **M3** Analyse the effectiveness of the models and techniques used within the AI-driven data solution. | |
| **LO4** Review the effectiveness of the AI-driven solution for a specified scenario against a non-AI solution | | |
| **P7** Implement a test plan to compare the effectiveness of an AI-driven solution against a non-AI solution for a specified scenario.<br>**P8** Review, using the results of the test plan, the effectiveness of an AI-driven solution against a non-AI solution for a specified scenario. | **M4** Evaluate the advantages and limitations of using an AI-driven solution compared with a non-AI solution for a specified scenario. | |

## Recommended Resources

### Textbooks

Alpaydin, E. (2020) *Introduction to Machine Learning*. 4th edn. Cambridge, MA: MIT Press.

Burkov, A. (2023). *The Hundred-Page Machine Learning Book*. 2nd edn. Andriy Burkov.

Géron, A. (2022) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd edn. Sebastopol, CA: O'Reilly Media.

Jung, A. (2022) *Machine Learning: The Basics*. Cham: Springer.

Mueller, J. P., Massaron, L., & Diamond, S. (2024). *Artificial Intelligence For Dummies*. Wiley.

Russell, S. and Norvig, P. (2021) *Artificial Intelligence: A Modern Approach*. 4th edn. Harlow: Pearson.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| ai.google/learn-ai-skills/ | Google AI<br>Learn essential AI Skills<br>(General Reference) |
| openlearning.mit.edu/news/13-foundational-ai-courses-resources-mit | MIT Open<br>13 foundational AI courses, resources from MIT<br>(General Reference) |
| www.elementsofai.com/ | Elements of AI<br>Welcome to the Elements of AI free online course!<br>(Online Course) |
| machinelearningmastery.com/start-here/ | Machine Learning Mastery<br>The 2026 guide to Machine Learning<br>(General Resource) |
| dev.to/william_roberts_fc2bfc1dc/how-to-build-an-ai-model-from-scratch-a-beginners-guide-33nh | William Roberts<br>How to Build an AI Model from Scratch: A Beginner's Guide<br>(Blog) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.netguru.com/blog/how-to-make-an-ai-model | NetGuru<br>How to create an AI model: a step-by-step guide for beginners<br>(Blog) |
| it.purdue.edu/ai/evaluating-ai-tools/ | Purdue University<br>Evaluating AI tools<br>(Article) |
| techcommunity.microsoft.com/blog/azure-ai-foundry-blog/genaiops-and-evals-best-practices/4409415 | Microsoft<br>GenAIOps and Evals Best Practices<br>(Blog) |

**Journals and articles**

Almatrafi, O., Johri, A. and Lee, H. (2024) 'A systematic review of AI literacy conceptualization, constructs, and implementation and assessment efforts (2019–2023)', *Computers and Education Open*, 6, 100173. Available at: doi.org/10.1016/j.caeo.2024.100173.

Huang, G., & Wang, S. (2023) 'Is artificial intelligence more persuasive than humans? A meta-analysis', *Journal of Communication*, 73(6), 552–562. Available at: doi.org/10.1093/joc/jqad024.

Shaveta. (2023) 'A review on machine learning ', *International Journal of Science and Research Archive*, 9(1), 281–285. Available at: doi.org/10.30574/ijsra.2023.9.1.0410

Steidl, M., Felderer, M., and Ramler, R. (2023) 'The pipeline for the continuous development of artificial intelligence models: Current state of research and practice ', *Journal of Systems and Software*, 199, 111615. Available at: doi.org/10.1016/j.jss.2023.111615.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science.*

# Unit 421: AI Automation and Digital Productivity

**Unit code:** D/652/3273

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Artificial Intelligence (AI) and automation are now embedded in many aspects of modern business operations. Organisations across sectors increasingly rely on digital tools to streamline routine tasks, interpret data, and support informed decision-making. As these technologies continue to evolve, they are reshaping how teams work, how services are delivered, and how businesses respond to new opportunities and challenges.

The aim of this unit is to give students a solid understanding of how AI and automation can improve productivity within a business environment. It introduces the core principles behind these technologies and explains how they can be applied in practical, operational contexts. Students will explore the potential benefits of AI-enabled systems as well as the considerations required to adopt them responsibly and effectively.

The unit covers several key areas that help students build a rounded view of AI-driven business productivity. These include the fundamentals of AI concepts, common automation tools used across organisations, and the ways AI supports efficiency, decision-making, and customer-focused services. Students will also investigate organisational risks, governance factors, and the steps involved in designing an AI-enabled solution for a specific business need.

On successful completion of this unit, students will be able to evaluate how AI tools contribute to business performance, assess opportunities for automation, and outline suitable AI-based solutions to real-world problems. These skills are valuable for a range of roles such as Business Analyst, Digital Transformation Assistant, Operations Co-ordinator, Automation Technician, Junior Data Analyst, and IT Support Technician, as well as further study within business, computing, or digital technologies.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core AI concepts relevant to business contexts

LO2 Discuss how organisations apply AI tools to enhance operational efficiency

LO3 Explain the organisational risks of AI adoption in business

LO4 Design an AI-enabled business solution for an identified scenario.

## Essential Content

LO1 **Explain the core AI concepts relevant to business contexts**

*Artificial Intelligence concepts:*

AI Definitions: artificial intelligence e.g. narrow AI, enterprise AI, generative AI; intelligent systems e.g. decision engines, knowledge systems, advisory systems; business intelligence e.g. reporting dashboards, analytics suites, insight reports;

Automation concepts: automation e.g. workflow automation, task automation, rule automation; intelligent business systems e.g. optimisation engines, recommendation engines, diagnostic systems; decision support systems e.g. BI dashboards, scenario models, scorecards, analytical frameworks

Automation opportunities: opportunity identification e.g. process inefficiencies, repetitive tasks, workflow gaps; automation potential e.g. suitability for AI or automation tools, value to organisation; decision support e.g. advising leaders, presenting options; ethical considerations e.g. fairness, transparency; workforce impact e.g. staff concerns, role changes, user needs.

*Machine learning approaches:*

Analytical models: predictive analytics e.g. forecasting models, trend patterns, demand curves; segmentation modelling e.g. customer clusters, behavioural groups, persona segments; optimisation models e.g. resource models, scheduling matrices, cost models

Performance concepts: anomaly detection e.g. fraud signals, deviation patterns, outlier markers; performance metrics e.g. accuracy scores, precision measures, F1 scores; model drift e.g. data drift, concept drift, feature drift.

*Data and algorithms:*

Enterprise data types: structured data e.g. transaction tables, sales records, product inventories; unstructured data e.g. emails, customer comments, text corpora; data governance e.g. quality rules, ownership structures, stewardship roles

Algorithmic elements: predictive algorithms e.g. scoring models, classification models, regression models; algorithmic pipelines e.g. preprocessing stages, feature pipelines, inference chains; business rules engines e.g. logic flows, policy rules, compliance checks.

*Business relevance of AI:*

Operational value: decision-making e.g. forecasting dashboards, prioritisation matrices, decision scorecards; process improvement e.g. workflow maps, bottleneck diagnostics, efficiency audits; scalability e.g. elastic compute, cloud resources, distributed processing

Customer and workforce impact: customer personalisation e.g. preference profiles, behavioural segments, recommendation sets; employee productivity e.g. digital assistants, workload insights, automation tools; process automation e.g. automated tasks, self-service flows, routine triage.

*Examples of real-world AI business tools applications:*

*A range of real-world AI business tools e.g.*

AI systems tools: Google Vertex AI e.g. model training, deployment; IBM Watson Assistant e.g. chatbots, customer support; Microsoft Azure OpenAI Service e.g. text generation, language processing

Machine learning tools: Amazon Forecast e.g. demand prediction, trend analysis; SAS Viya e.g. predictive modelling, analytics; DataRobot AI Cloud e.g. automated models, deployment

Decision systems: H2O.ai Driverless AI e.g. automated machine learning, optimisation; TensorFlow Extended e.g. pipelines, model management; IBM Operational Decision Manager e.g. rule-based decisions, automation

Business intelligence tools: Power BI Copilot e.g. dashboards, insights; Tableau with Einstein Discovery e.g. predictions, analytics; Qlik Sense e.g. data exploration, reporting

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Discuss how organisations apply AI tools to enhance operational efficiency**

*AI applications in organisations:*

Automation tools: workflow automation e.g. approval flows, rule scripts; robotic process automation e.g. task bots, automation scripts; intelligent scheduling e.g. rota planners, resource allocators

Analytical tools: predictive analytics e.g. forecasting models, demand curves; recommendation systems e.g. ranking algorithms, preference models; anomaly detection e.g. fraud flags, deviation markers

Customer-facing tools: chatbots e.g. FAQ responders, support bots; virtual assistants e.g. voice interfaces, task assistants; self-service engines e.g. account portals, diagnostic modules

Organisational leadership: leadership roles e.g. oversight, direction, policies; organisational values e.g. ethics, responsible AI culture, accountability; strategy alignment e.g. long-term planning, sustainability, organisational objectives

Business case for ethical AI: reputational risk e.g. brand trust, stakeholder confidence; staff morale e.g. confidence, engagement, change acceptance; long-term sustainability e.g. responsible growth, regulatory alignment, ethical resilience.

*Operational efficiency and performance impact:*

Efficiency: process optimisation e.g. workflow maps, bottleneck diagnostics, workload balancing; cost reduction e.g. savings reports, cost dashboards; time savings e.g. cycle-time reductions, task shortcuts; sustainability e.g. energy efficiency, resource conservation; contributions e.g. design support, process improvement

Performance measures: productivity improvement e.g. utilisation metrics, throughput rates; error reduction e.g. validation layers, error flags; performance measurement e.g. KPI dashboards, scorecards

Reporting and evaluation: reporting e.g. dashboards, summaries, insights for stakeholders; evaluation e.g. productivity gains, efficiency improvements, cost savings

Automation opportunities and limitations: opportunities e.g. further automation, process improvements; limitations e.g. reduced user experience, process inefficiencies, unsuitable use

Resource management: resource allocation e.g. staffing matrices, utilisation charts; capacity planning e.g. demand forecasts, load estimates; workload balancing e.g. queue distribution, task routing.

*Business functions using AI:*

Customer-focused functions: customer service e.g. support queues, enquiry records; marketing analytics e.g. campaign metrics, targeting profiles; service personalisation e.g. preference clusters, behaviour segments

Operational functions: operations management e.g. process KPIs, workflow trackers; supply chain management e.g. inventory datasets, logistics flows; logistics co-ordination e.g. routing plans, delivery schedules

Enterprise functions: finance e.g. audit trails, financial dashboards; human resources e.g. recruitment metrics, workforce data; administrative services e.g. scheduling systems, document repositories.

*Implementation considerations:*

Technical factors: integration with existing systems e.g. API connectors, middleware; data availability e.g. operational datasets, customer records; system compatibility e.g. platform standards

Human factors: workforce readiness e.g. digital literacy baselines; training requirements e.g. skills modules, user guidance; change management e.g. communication plans, adoption roadmaps

Organisational factors: organisational capability e.g. digital maturity, process ownership; governance alignment e.g. policy frameworks, oversight groups; process maturity e.g. workflow baselines

*Examples of real-world AI operation applications:*

Automation tools: UiPath Automation Cloud e.g. process automation, task bots; Blue Prism Intelligent Automation e.g. robotic process automation, workflows; Microsoft Power Automate e.g. workflow automation, task triggers

Process optimisation tools: Celonis Process Mining e.g. process discovery, bottlenecks; IBM Process Mining e.g. process tracking, performance insights; SAP Signavio e.g. process modelling, optimisation

Customer service tools: Salesforce Einstein e.g. customer insights, predictions; Zendesk AI e.g. ticket classification, chatbots; HubSpot Marketing Hub AI e.g. campaign automation, segmentation

Integration tools: MuleSoft Anypoint Platform e.g. API integration, connectivity; Workato e.g. workflow automation, data sync; Apache Kafka e.g. data streaming, event processing

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Explain the organisational risks of AI adoption in business**

*Ethical and social risks:*

Fairness factors: algorithmic bias e.g. skewed datasets, sampling imbalance, demographic distortion; fairness concerns e.g. protected-group disparities, unequal outcomes, representation gaps; transparency needs e.g. explainable outputs, interpretability layers

Accountability elements: accountability structures e.g. oversight committees, audit groups, review panels; social impact e.g. stakeholder concerns, community effects; ethical expectations e.g. responsible-use principles, ethical frameworks.

*Legal and regulatory risks:*

Compliance requirements: data protection e.g. privacy rules, consent conditions, retention policies; regulatory compliance e.g. governance policies, audit controls; intellectual property e.g. ownership rights, asset licensing, data rights

Standards and frameworks: governance frameworks e.g. risk models, assurance frameworks; industry standards e.g. compliance controls, quality benchmarks; privacy requirements e.g. consent rules, disclosure obligations.

*Operational and strategic risks:*

System vulnerabilities: system failure e.g. outage events, model crashes, dependency failures; data quality issues e.g. missing records, duplicate entries, inconsistent formats; cybersecurity threats e.g. intrusion risks, data breaches, adversarial attacks

Business exposure: reputational risk e.g. trust erosion, negative perception; over-reliance on automation e.g. single-point systems, unsupported overrides; strategic misalignment e.g. misaligned objectives, incongruent use cases.

*Risk mitigation awareness:*

Control mechanisms: human oversight e.g. review checkpoints, escalation routes; governance controls e.g. approval workflows, audit trails; policy frameworks e.g. ethical guidelines, compliance policies

Monitoring elements: compliance monitoring e.g. monitoring dashboards, audit logs; life cycle monitoring e.g. model reviews, performance checks; ethical guidelines e.g. behaviour codes, fairness criteria

*Examples of real-world AI business applications:*

A range of real-world AI business tools e.g.

Governance tools: OneTrust e.g. data privacy, consent tracking; TrustArc e.g. compliance checks, risk assessment; Microsoft Purview e.g. policy control, data governance

Security tools: Darktrace Cyber AI e.g. threat detection, anomaly alerts; CrowdStrike Falcon e.g. endpoint security, threat monitoring; IBM Security QRadar e.g. security analysis, event monitoring

Risk management tools: Azure ML Responsible AI Dashboard e.g. bias detection, model evaluation; IBM GRC e.g. risk assessment, audit control; SAP GRC e.g. compliance monitoring, access control

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Design an AI-enabled business solution for an identified scenario**

*AI-enabled business solutions:*

Solution design elements: AI tools selection e.g. analytics engines, automation platforms, predictive models; business problem alignment e.g. operational gaps, value drivers, performance targets; system functionality e.g. feature sets, data workflows, integration layers

Automation tools: workflow configuration e.g. triggers, actions, conditional logic; integration setup e.g. API connectors, data flows, system linking; solution adaptation e.g. modifying workflows to meet organisational processes, user requirements, compliance needs

Solution objectives: business value e.g. efficiency gains, quality improvements, cost reductions; ROI e.g. cost–benefit indicators, investment metrics; performance metrics e.g. success criteria, output measures, productivity indicators.

*Scenario analysis:*

Context factors: organisational context e.g. structure, culture, capabilities; business needs e.g. priority areas, value opportunities, service demands; stakeholders e.g. users, managers, customers

Environmental considerations: constraints e.g. budgets, timelines, resources; operational environment e.g. workflow conditions, system dependencies; adoption factors e.g. readiness levels, training requirements.

*Feasibility considerations:*

Technical feasibility: data availability e.g. dataset readiness, data coverage, quality baselines; model suitability e.g. fit indicators, performance thresholds; integration constraints e.g. legacy systems, API limitations

Organisational feasibility: organisational readiness e.g. skills baseline, resource levels; cost considerations e.g. licensing, infrastructure costs; implementation constraints e.g. deployment windows, staffing limits.

*Designing AI systems:*

Architecture elements: requirements identification e.g. functional needs, data needs; data pipeline design e.g. ingestion flows, transformation layers, storage approaches; architecture planning e.g. cloud layouts, hybrid models, enterprise environments

Interaction and monitoring: human-AI interaction e.g. user interfaces, decision support screens; system integration e.g. ERP links, CRM connectors; life cycle management e.g. monitoring dashboards, model reviews, update cycles

System design documentation: technical specifications e.g. data dictionaries, model specifications, interface descriptions; design blueprints e.g. architecture diagrams, workflow designs, integration diagrams; configuration documentation e.g. parameter settings, environment profiles, version records; governance documentation e.g. policy documents, approval logs; tool selection e.g. low-code/no-code platforms, integration tools, automation environments; integration methods e.g. API connectors, webhooks, middleware, data pipelines; system integration e.g. linking applications, synchronising data, enabling end-to-end automation;

System testing: testing procedures e.g. workflow testing, integration testing, user acceptance testing; validation checks e.g. expected outputs, data accuracy, process reliability; error handling e.g. exception paths, failure responses, retry logic; refinement e.g. debugging workflows, optimising integrations, iterative improvement

Implementation planning: practices e.g. deploying workflows, testing outputs, validating performance; organisational context e.g. aligning automation with business processes, data governance requirements, user roles, and operational constraints; optimisation e.g. refining workflows for efficiency, reliability, and scalability.

*Examples of real-world AI applications:*

A range of real-world AI tools e.g.

AI solution platforms: AWS SageMaker e.g. model development, training, deployment; Microsoft Azure Machine Learning e.g. model building, life cycle management

Automation tools: Microsoft Power Automate e.g. workflow automation, task triggers; Zapier e.g. app integration, task automation

Business systems: Salesforce Service Cloud e.g. customer service, case management; HubSpot CRM e.g. customer tracking, sales management

Data platforms: Snowflake Data Cloud e.g. data storage, processing; Apache Airflow e.g. workflow scheduling, pipeline automation

Infrastructure tools: AWS Lambda e.g. serverless computing, event processing; Google BigQuery e.g. data analysis, data warehousing

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the core AI concepts relevant to business contexts</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Evaluate using a range of real-world examples, the effectiveness of the use of AI concepts and tools in enhancing operational efficiency across an organisation considering the legal, ethical and organisational risks.</td></tr>
<tr><td>P1 Explain, using a range of real-world examples, the core AI concepts relevant to business contexts.<br><br>P2 Discuss, using a range of real-world examples, how these concepts are used within organisations.</td><td>M1 Analyse how core AI concepts support business decision-making in organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Discuss how organisations apply AI tools to enhance operational efficiency</td></tr>
<tr><td>P3 Discuss, using a range of real-world examples, how organisations apply a range of AI tools to enhance operational efficiency across the business.<br><br>P4 Explore, using a range of real-world examples, the impact of implementing these tools across a business.</td><td>M2 Assess, using a range of real-world examples, the effects of implementing a range of AI tools on operational efficiency.</td></tr>
<tr><td colspan="2">LO3 Explain the organisational risks of AI adoption in business</td></tr>
<tr><td>P5 Explain, using a range of real-world examples, the business risks associated with AI adoption in business contexts.<br><br>P6 Explore, using a range of real-world examples, the legal and ethical risks associated with AI adoption in business contexts.</td><td>M3 Assess, using a range of real-world examples, the legal, ethical and organisational risks associated with AI adoption in an identified business context.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Design an AI-enabled business solution for an identified scenario | | |
| **P7** Design an AI-enabled business solution for an identified organisational scenario.<br><br>**P8** Review the design of an AI-enabled business solution with a group of identified stakeholders to obtain meaningful feedback and identify improvements. | **M4** Evaluate, using the feedback obtained, the effectiveness of the designed AI-enabled solution in meeting the requirements of the identified organisational scenario. | **D2** Justify the decisions made in the design of the AI-enabled business solution for an identified organisational scenario, commenting on its strengths, weaknesses and areas for future development. |

 © Pearson Education Limited 2026

## Recommended Resources

### Textbooks


Agrawal, A., Gans, J. and Goldfarb, A. (2022) *Power and Prediction: The Disruptive Economics of Artificial Intelligence*. Boston, MA: Harvard Business Review Press.

Cummings, M. (2022) *Artificial Intelligence in Context: The Regulatory, Ethical and Societal Landscape*. Oxford: Oxford University Press.

Davenport, T.H. and Mittal, N. (2023) *All-in On AI: How Smart Companies Win Big with Artificial Intelligence*. Harvard Business Press.

Géron, A. (2022) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd edn. Sebastopol, CA: O'Reilly Media.

Hemachandran, K. and Rodriguez, R.V. (eds.) (2023) *Artificial Intelligence for Business: An Implementation Guide Containing Practical and Industry-Specific Case Studies*. Boca Raton: Productivity Press.

Lardi, K. (2025) *Artificial Intelligence for Business: Harness AI for Value, Growth and Innovation*. London: Kogan Page.

Panda, S.K., *et al*. (eds.) (2021) *Artificial Intelligence and Machine Learning in Business Management: Concepts, Challenges, and Case Studies.*

Russell, S. and Norvig, P. (2021) *Artificial Intelligence: A Modern Approach*. 4th edn. Harlow: Pearson.


### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.deloitte.com/global/en/issues/generative-ai/ai-roi-the-paradox-of-rising-investment-and-elusive-returns.html | Deloitte<br>Generative AI research<br>(Report) |
| artificialintelligenceact.eu/the-act/ | EU Artificial Intelligence Act,<br>The Act Texts<br>(General reference) |
| nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf | NIST<br>Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile<br>(Report) |

| | |
|---|---|
| www.workato.com/work-automation-index | Workato<br>Work Automation Index 2024<br>(Report) |
| post.parliament.uk/artificial-intelligence-ethics-governance-and-regulation/ | UK Parliament POST<br>Horizon scanning. Artificial intelligence: ethics, governance and regulation<br>(General reference) |
| cloud.google.com/transform/101-real-world-generative-ai-use-cases-from-industry-leaders | Google<br>1,302 real-world gen AI use cases from the world's leading organizations<br>(General Resource) |
| ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/guidance-on-ai-and-data-protection/ | ICO<br>Guidance on AI and data protection<br>(General Reference) |
| docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/machine-learning-lens.html | AWS<br>Machine Learning Lens – AWS Well-Architected Framework<br>(Framework) |

### Journals and articles

Batool, A., Zowghi, D. and Bano, M. (2025) 'AI governance: a systematic literature review', *AI and Ethics*, 5, pp. 3265–3279. Available at: doi.org/10.1007/s43681-024-00653-w.

Machucho, R. and Ortiz, D. (2025) 'The Impacts of Artificial Intelligence on Business Innovation: A Comprehensive Review of Applications, Organizational Challenges, and Ethical Considerations', *Systems*, 13(4), 264. Available at: doi.org/10.3390/systems13040264.

Patrício, L., Varela, L. and Silveira, Z. (2024) 'Integration of Artificial Intelligence and Robotic Process Automation: Literature Review and Proposal for a Sustainable Model', *Applied Sciences*, 14(21), 9648. Available at: doi.org/10.3390/app14219648.

Zebec, A. and Indihar Štemberger, M. (2024) 'Creating AI business value through BPM capabilities', *Business Process Management Journal*, 30(8). Available at: doi.org/10.1108/BPMJ-07-2023-0566.

### Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 420: Core AI.*

# Unit 422: Ethics and AI

**Unit code:** F/652/3274

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Artificial Intelligence (AI) now plays a routine role in how organisations make decisions, deliver services, and manage information. From automated recommendations to data-driven risk assessments, AI systems influence many everyday processes, often in ways that are not immediately visible. As these technologies become more embedded across workplaces and public services, understanding their ethical implications is increasingly important for anyone entering a digital or technology-enabled profession.

The aim of this unit is to help students develop a clear and practical understanding of the ethical issues that arise when AI systems are designed, deployed, and used. The unit focuses on the principles that guide responsible practice, including fairness, transparency, accountability, privacy and responsible decision-making, giving students the foundation needed to recognise the ethical expectations placed on modern organisations.

Students will explore how AI is applied ethically and unethically across a range of real examples. They will examine the risks that can emerge – such as bias, lack of transparency, or loss of human oversight – and consider how ethical frameworks can support better choices in development and governance. The unit also encourages students to think critically about real-world scenarios, looking at how organisations respond to challenges and what good practice looks like in everyday use.

On successful completion of the unit, students will be able to discuss ethical principles confidently, apply ethical frameworks to practical situations and evaluate the impact of AI systems in different contexts. These skills support a range of early-career job roles where ethical awareness is essential, including Data Technician, Junior Business Analyst, AI Operations Assistant, Digital Support Technician, Cybersecurity Technician, and Technology Governance Assistant.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Describe the core ethical principles in the development and use of AI systems

LO2 Discuss the ethical and unethical use of AI applications

LO3 Explore the application of ethical frameworks in AI development

LO4 Review the ethical application of AI in an identified organisational scenario.

## Essential Content

LO1 **Describe the core ethical principles in the development and use of AI systems**

*Core ethical principles in artificial intelligence:*

Core values: fairness e.g. bias, inequality; transparency e.g. clarity, explanation, access; accountability e.g. responsibility, ownership, review; privacy e.g. consent, data protection; responsibility e.g. duty, standards, care

Governance: oversight e.g. supervision, checks; trust e.g. reliability, confidence, consistency.

*Fairness and bias:*

Bias sources: bias e.g. sampling bias, measurement bias; data quality e.g. missing data, outdated data, noisy data; discrimination risks e.g. exclusion, unfair outcomes

Outcome quality: fairness of outcomes e.g. equal access, balanced results, parity

Design considerations: inclusive design e.g. accessibility, user diversity; diverse data e.g. representation, variation, coverage.

*Transparency and explainability:*

Communication: clear information e.g. summaries, guidance; open communication e.g. disclosure, reporting, openness

Understanding: understandable decisions e.g. clarity, simplicity; simple explanations e.g. examples, descriptions, notes

Documentation: accessible documentation e.g. guides, instructions; visible decision steps e.g. workflow, stages, reasoning.

*Accountability and responsibility:*

Roles: clear roles e.g. responsibilities, ownership; decision ownership e.g. authority, control, assignment

Practices: responsible use e.g. correct use, safe use; organisational responsibility e.g. policies, guidelines, procedures, knowledge currency

Oversight: ethical judgement e.g. evaluation, considerations; oversight processes e.g. checks, reviews, audits.

*Privacy and data protection:*

Data Control: personal data e.g. records, identifiers; consent e.g. permission, agreement

Security: data security e.g. safeguards, controls; anonymisation e.g. masking, removal, generalisation

Rights: user rights e.g. access rights, correction rights; information protection e.g. confidentiality, compliance, storage

Handling: safe data handling e.g. storage, transmission.

*Examples of real-world ethical AI applications:*

A range of real-world ethical AI tools e.g.

Bias and fairness tools: IBM AI Fairness 360 e.g. bias checks, fairness testing; Google What-If Tool e.g. model testing, bias analysis; Fiddler AI e.g. model monitoring, fairness insights

Explainability tools: Google Explainable AI e.g. model interpretation, feature impact; LIME e.g. local explanations; SHAP e.g. prediction explanations

Safety and governance tools: AWS Bedrock Guardrails e.g. content control, safety filters; Vertex AI Safety e.g. risk controls; Azure AI Safety Tools e.g. safety testing

Data protection tools: Google Cloud DLP e.g. data masking, data discovery; AWS Macie e.g. data security, sensitive data detection; Microsoft Purview DLP e.g. compliance control, data protection

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Discuss the ethical and unethical use of AI applications**

*Ethical use of AI applications:*

Positive outcomes: beneficial outcomes e.g. improved services, public value; social good e.g. community benefit, wider support, inclusion; efficiency gains e.g. time savings, resource savings

Human-centred values: human-centred design e.g. user needs, accessibility; accessibility e.g. readability, usability, reach; fairness of use e.g. equal access, balanced outcomes, parity;

Decision support: support for decision-making e.g. recommendations, guidance.

*Unethical use of AI applications:*

Misuse risks: surveillance misuse e.g. tracking, monitoring; manipulation e.g. persuasion, influence, pressure; misinformation e.g. false content, misleading information

Harmful automation: harmful automation e.g. job displacement, errors; exclusion e.g. service denial, unfair omission, access barriers

Privacy issues: lack of consent e.g. unauthorised use, hidden collection; privacy risks e.g. exposure, breaches, leaks; unfair treatment e.g. biased decisions, inconsistent results, unequal outcomes.

*Ethical risks associated with AI use:*

Risk factors: bias risks e.g. sampling bias, unequal impact; unclear decisions e.g. opacity, uncertainty, unexplained outputs; privacy concerns e.g. data misuse, insecurity

User impact risks: harm to users e.g. confusion, disadvantage; loss of control e.g. dependency, unpredictability, reduced input; unequal impacts e.g. varied outcomes, imbalance

Human involvement risks: reduced human involvement e.g. automation, dependence; unintended consequences e.g. errors, side effects, surprises.

*Contextual application considerations:*

Operational context: sector context e.g. healthcare settings, business operations; organisational practices e.g. policies, workflows, procedures; stakeholder needs e.g. users, customers, communities

Risk context: level of risk e.g. sensitive tasks, critical actions; scale of use e.g. small systems, large deployments, platforms

Purpose and impact: intended purpose e.g. goals, functions; user impact e.g. experience, outcomes, effects; environmental factors e.g. conditions, surroundings, external factors

*Examples of real-world AI applications:*

A range of real-world AI tools e.g.

Ethical AI applications: Microsoft Copilot e.g. task support, content generation; Google Maps e.g. navigation, route planning; IBM Watson Assistant e.g. chatbots, customer support

Unethical AI applications: Clearview AI e.g. facial recognition, identity tracking; Hikvision Analytics e.g. surveillance monitoring, behaviour tracking; DeepFace e.g. facial recognition, identity detection

AI risk examples: COMPAS e.g. risk scoring, decision bias; YouTube Recommendations e.g. content suggestions, algorithm bias;

Facial Recognition Systems e.g. identification, privacy concerns

Organisational AI use cases: NHS AI triage tools e.g. patient prioritisation, decision support; government chatbots e.g. public queries, service access; banking fraud detection systems e.g. transaction monitoring, anomaly detection

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Explore the application of ethical frameworks in AI development**

*Ethical frameworks for AI:*

Principles and values: principles e.g. guidelines, standards; rights e.g. permissions, access; values e.g. fairness, safety, trust

Fairness and transparency: fairness e.g. equal treatment, unbiased outcomes; transparency e.g. clear information, simple explanations, openness; accountability e.g. responsibility, oversight

Human oversight: human oversight e.g. monitoring, supervision; risk awareness e.g. warnings, indicators; responsible practice e.g. safe behaviour, ethical conduct.

*Application of ethical frameworks in AI development:*

Design considerations: design choices e.g. layouts, workflows; data selection e.g. datasets, filtering; documentation e.g. notes, records

Development processes: basic testing e.g. checks, trials, cases; decision steps e.g. approvals, reviews; ethical checks e.g. fairness checks, safety checks

User-focused guidance: user needs e.g. preferences, requirements; review processes e.g. quick reviews, brief checks, assessments.

*Benefits of ethical frameworks for AI development:*

Clarity and consistency: clarity e.g. clear steps, clear purpose; consistency e.g. stable processes, predictable approaches; guidance e.g. instructions, direction

Trust and accountability: trust e.g. confidence, reliability; accountability e.g. responsibility, ownership, answerability; responsible practice e.g. ethical practice, safe behaviour

Risk management: risk reduction e.g. fewer issues, fewer harms; safer outcomes e.g. protection, reliability, safety; fairness e.g. equal chances, balanced outcomes.

*Limitations of ethical frameworks in AI development:*

Practical constraints: constraints e.g. time limits, resource limits; changing situations e.g. new conditions, updated requirements; incomplete information e.g. missing data, partial data, unclear data

Interpretation challenges: interpretation e.g. differing views, alternative readings; unclear guidance e.g. vague rules, incomplete instructions; differing contexts e.g. environments, use cases, settings

Human factors: human error e.g. mistakes, oversights.

*Examples of real-world AI applications:*

A range of real-world AI tools e.g.

Ethical frameworks and tools: ChatGPT e.g. conversational AI, user interaction; Meta AI Fairness Tools e.g. bias analysis, fairness checks; Amazon Rekognition e.g. image recognition, identity detection

Ethical development tools: Amazon SageMaker Clarify e.g. bias detection, model evaluation; Google Explainable AI e.g. model interpretation, transparency; Azure Model Evaluation Tools e.g. performance checks, validation

Risk management tools: NIST AI Risk Management Tools e.g. risk assessment, governance guidance; Darktrace Cyber AI e.g. threat detection, anomaly alerts; Mastercard Decision Intelligence e.g. fraud detection, risk analysis

Limitations and real-world systems: YouTube Recommendation AI e.g. content suggestions, algorithm bias; TikTok Algorithm e.g. content ranking, engagement optimisation; Netflix Personalisation AI e.g. viewing recommendations, user profiling

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Review the ethical application of AI in an identified organisational scenario**

*Practical AI application scenarios:*

Sector applications: business systems e.g. scheduling tools, retail platforms; public services e.g. council portals, transport systems, service platforms; customer services e.g. support bots, helpdesk systems, response systems

Technology applications: healthcare tools e.g. diagnostic tools, symptom-checkers; finance tools e.g. budgeting tools, fraud-flags, scoring dashboards; digital platforms e.g. content systems, sharing tools

Operational applications: education systems e.g. grading tools, learning portals; workplace automation e.g. task systems, workflow boards, monitoring dashboards.

*Ethical considerations in real-world scenarios:*

Core considerations: impact on people e.g. well-being, inclusion; fairness e.g. equal chances, non-bias; accuracy of decisions e.g. correct outputs, reliable scoring, consistent results

Privacy and control: privacy e.g. data protection, information control, safe handling; consent e.g. user agreement, permission records; safe use e.g. safeguards, protections

Transparency and responsibility: transparency e.g. clear information, open details, plain communication; organisational responsibility e.g. duty roles, reporting channels.

*Ethical outcomes of AI use:*

Positive outcomes: positive effects e.g. improved services, better access; risk reduction e.g. fewer issues, fewer mistakes, fewer harms; trust e.g. confidence, reliability

Negative outcomes: negative effects e.g. errors, confusion; unintended results e.g. mismatches, misclassification, side effects; harm prevention e.g. safety checks, mitigations

Service outcomes: user satisfaction e.g. helpful experience, positive feedback; improvement to services e.g. smoother access, quicker processes, better support.

*Stakeholder impact and outcome variation:*

Direct impacts: user impact e.g. convenience, experience; organisational impact e.g. efficiency, workload; community effects e.g. public benefit, local impact

Outcome variation: unequal outcomes e.g. imbalances, disparities; benefits and risks e.g. gains, drawbacks, trade-offs; fairness differences e.g. varied access, uneven results

Performance variation: confidence levels e.g. assurance, certainty; accessibility e.g. availability, usability, reach.

*Comparative ethical application:*

Contextual variation: different contexts e.g. workplaces, platforms; variation in outcomes e.g. changes, differences; decision differences e.g. alternative routes, approaches

Environmental influences: environment influence e.g. settings, conditions; levels of oversight e.g. checks, supervision; changing needs e.g. updates, requirements

Purpose and scale: purpose of use e.g. goals, intentions; scale of deployment e.g. small scale, large scale, phased use.

*Examples of real-world AI applications:*

A range of real-world AI tools e.g.

AI applications: Watson Assistant e.g. support bots, customer queries; Salesforce Einstein e.g. customer insights, service automation; Webex Contact Centre AI e.g. call routing, customer interaction

Data and analysis tools: Google Vision AI e.g. image recognition, data extraction; Watson Discovery e.g. document search, information retrieval; Amazon Comprehend e.g. text analysis, sentiment detection

Healthcare tools: NHS Triage AI e.g. patient prioritisation, decision support; Google Health AI e.g. predictions, diagnostics; Tempus AI e.g. clinical analysis, data insights

Logistics tools: UPS ORION e.g. route optimisation, delivery planning; Google Maps AI e.g. route planning, traffic updates; DHL AI e.g. warehouse automation, inventory management

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Describe the core ethical principles in the development and use of AI systems</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a range of real-world examples, the effectiveness of the application of core ethical principles in the development of AI systems, considering their ethical and unethical uses.</td></tr>
<tr><td>P1 Discuss, using a range of real-world examples, the core ethical principles, including fairness and transparency, in the development and use of AI systems.<br><br>P2 Explore, using a range of real-world examples, how organisations ensure accountability and privacy in the development and use of AI systems.</td><td>M1 Analyse, using a range of real-world examples, how the application of core ethical principles can influence the development and use of AI systems.</td></tr>
<tr><td colspan="2">LO2 Discuss the ethical and unethical use of AI applications</td></tr>
<tr><td>P3 Explore, using a range of real-world examples, the ethical and unethical uses of AI applications in a range of different contexts.<br><br>P4 Discuss, using a range of real-world examples, the ethical risks of the use of AI applications in a range of different contexts.</td><td>M2 Assess using a range of real-world examples, the impact of the ethical and unethical use of AI applications on individuals, organisations, and society.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Explore the application of ethical frameworks in AI development | | **LO3 and LO4**<br>**D2** Evaluate the effectiveness of ethical frameworks in the development and use of an AI application for an identified organisational scenario. |
| **P5** Explore, using a range of real-world examples, the application of the different ethical frameworks in the development of AI systems.<br>**P6** Review, using a range of real-world examples, the benefits and limitations to an organisation of the application of ethical frameworks in the development of AI systems. | **M3** Analyse, using a range of real-world examples, the impact of the application of ethical frameworks on AI system development. | |
| **LO4** Review the ethical application of AI in an identified organisational scenario | | |
| **P7** Review the ethical considerations of the use of AI applications in an identified organisational scenario.<br>**P8** Discuss the ethical outcomes of the use of AI applications in an identified organisational scenario. | **M4** Analyse the ethical application of AI in an identified organisational scenario, considering the outcomes for the organisation and its stakeholders. | |

## Recommended Resources

### Textbooks

Bullock, J.B. *et al.* (eds.) (2024) *The Oxford Handbook of AI Governance*. New York: Oxford University Press.

Coeckelbergh, M. (2025) *AI Ethics*. 2nd edn. Cambridge, MA: MIT Press.

Floridi, L. (2024) *Ethics of Artificial Intelligence*. 2nd edn. Oxford: Oxford University Press.

O'Regan, G. (2024) *Ethical and Legal Aspects of Computing: A Professional Perspective from Software Engineering.* Cham: Springer.

Russell, S. and Norvig, P. (2021) *Artificial Intelligence: A Modern Approach*. 4th edn. Harlow: Pearson.

Smuha, N.A. (ed.) (2025) *The Cambridge Handbook of the Law, Ethics and Policy of Artificial Intelligence*. Cambridge: Cambridge University Press

Véliz, C. (2025) *Privacy Is Power: Why and How You Should Take Back Control of Your Data*. 2nd edn. London: Bantam Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.gov.uk/data-ethics-guidance/understanding-artificial-intelligence-ethics-and-safety | UK Government<br>Understanding artificial intelligence ethics and safety<br>(General reference) |
| www.gov.uk/government/publications/data-ethics-framework | UK Government<br>Data and AI Ethics Framework<br>(Guidance) |
| www.oecd.org/en/topics/sub-issues/ai-principles.html | Organisation for Economic Co-operation And Development OECD<br>AI principles and policy guidance for trustworthy AI<br>(General reference) |
| www.unesco.org/en/articles/recommendation-ethics-artificial-intelligence | UNESCO<br>Ethics of artificial intelligence recommendations and global guidance<br>(General reference) |

| | |
|---|---|
| www.iso.org/standard/42001 | International Organization for Standardization<br>AI governance and AI management system standards (ISO/IEC 42001)<br>(General reference) |
| ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/guidance-on-ai-and-data-protection/ | Information Commissioner's Office<br>Guidance on AI, data protection, and privacy compliance<br>(General reference) |
| www.nist.gov/itl/ai-risk-management-framework | National Institute of Standards and Technology<br>AI risk management framework and responsible AI resources<br>(Framework) |
| aiethicsjournal.org | AI Ethics Journal<br>Applied ethics case studies and AI governance discussions<br>(Resource) |

### Journals and articles

Floridi, L. *et al*. (2018) 'AI4People—An Ethical Framework for a Good AI Society : Opportunities, Risks, Principles, and Recommendations', *Minds and Machines*, 28(4), pp. 689–707. Available at: doi.org/10.1007/s11023-018-9482-5.

Jobin, A., Ienca, M. and Vayena, E. (2019) 'The global landscape of AI ethics guidelines', *Nature Machine Intelligence*, 1, pp. 389–399. Available at: doi.org/10.1038/s42256-019-0088-2.

Leslie, D. (2019) 'Understanding artificial intelligence ethics and safety: A guide for the responsible design and implementation of AI systems in the public sector', *The Alan Turing Institute*. Available at: doi.org/10.5281/zenodo.3240529.

Mittelstadt, B.D. *et al*. (2016) 'The ethics of algorithms: Mapping the debate', *Big Data & Society*, 3(2). Available at: doi.org/10.1177/2053951716679679.

### Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 420: Core AI.*

# Unit 423: Agentic AI

**Unit code:** H/652/3275

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Agentic AI is becoming an important part of today's digital tools, helping systems do more than simply follow fixed instructions. Instead of waiting for a user to tell them what to do at every step, these systems can set goals, plan tasks and respond to changes as they happen. Agentic AI is currently being used in areas such as customer support, data processing, software tools and automated business services. As more organisations rely on technology that can act with some level of independence, it is becoming essential for students to understand how these systems work and what they can do.

The aim of this unit is to give students a clear foundation in how agentic AI works, focusing on the principles that guide autonomous systems and how these systems are structured. Students will explore the key ideas that shape agent-based behaviour, the different ways agents can be designed, and the factors that influence why certain models are chosen. This helps students develop a practical understanding of how AI systems are planned and built.

The unit introduces the main components of agentic AI, including autonomy, goals, environments and behaviour models. Students will examine common agent architectures, such as reactive, deliberative and hybrid designs, and review the range of different agentic AI models and how these models are used to support organisational decision-making. The unit then moves on to designing a simple agent workflow, including goal setting, task breakdown and tool selection, before looking at how such a workflow can be implemented in practice.

On successful completion of this unit, students will be able to explain the core principles of agentic AI, design suitable workflows and understand how autonomous systems behave in different situations. These skills support progression into roles that increasingly rely on AI-enabled tools, including Junior AI Technician, Digital Support Specialist, Automation Assistant, Data Technician, Software Support Analyst and Technical Operations Assistant.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core principles of agentic AI systems

LO2 Review the architectures and models used in designing agentic AI systems

LO3 Design a simple agent workflow to accomplish a defined task for an identified scenario

LO4 Implement a simple agent workflow to accomplish a defined task for an identified scenario.

## Essential Content

LO1 **Explain the core principles of agentic AI systems**

*Agentic AI fundamentals:*

Autonomy and goals: autonomy e.g. independence, control, self-direction; goals e.g. aims, targets, intentions

Perception and environment: perception e.g. sensing, detection, intake; action e.g. movement, response, output; environment e.g. setting, context, surroundings; interaction e.g. engagement, contact, connection

Decision processes: decisions e.g. choices, options, selections; reasoning e.g. logic, explanation, analysis

Adaptation & responses: adaptation e.g. adjustment, change, modification; responses e.g. reactions, outputs, behaviours; selection e.g. choice, picking, option.

*Goals and environments:*

Goal types: goals e.g. objectives, targets, outcomes; specification e.g. requirements, details, criteria

Environment types: environments e.g. settings, contexts, conditions; conditions e.g. factors, limits, requirements

Observability: environments e.g. visible cues, available data, observable states

Environment dynamics: environments e.g. stable states, variable states, changing conditions; assumptions e.g. expectations, constraints, premises.

*Agent behaviours:*

Behaviour types: behaviour e.g. patterns, actions, reactions; triggers e.g. signals, cues, prompts; rules e.g. conditions, instructions, guidelines

Reasoning and planning: reasoning e.g. logic, thinking, analysis; planning e.g. preparation, structuring, organising; evaluation e.g. assessment, review, checking.

Utility and goals: utility e.g. value, benefit, priority; goals e.g. intentions, aims, targets

Feedback mechanisms: feedback e.g. responses, signals, indicators; mechanisms e.g. systems, processes, structures.

*Examples of real-world agentic AI applications:*

A range of agentic AI software examples e.g.

Decision processing e.g. IBM Watson decision engine, Google DeepMind reasoning models, Microsoft Azure Decision Service

Environments e.g. Apple HomeKit context triggers, Google Home routines, Samsung SmartThings environment rules

Reasoning and planning e.g. DeepMind AlphaZero planning, OpenAI planning agents, Google Task Planner AI

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Review the architectures and models used in designing agentic AI systems**

*Reactive agent architectures:*

Condition–action mechanisms: condition–action rules e.g. rule matching, action mapping; stimulus–response mappings e.g. simple input, basic reaction; triggers e.g. event signal, condition check

Internal state factors: internal state e.g. stored value, memory item; signals e.g. input signal, output signal; actions e.g. chosen action, system response; responsiveness e.g. reaction speed, system adjustment.

*Deliberative agent architectures:*

Model representation: world models e.g. environment map, stored representation; representations e.g. symbolic model, simple form; state descriptions e.g. current state, updated state

Planning structures: planning mechanisms e.g. action plan, step sequence; reasoning processes e.g. logical step, option choice; decision trees e.g. branch path, option route; evaluations e.g. outcome check, simple rating.

*Hybrid agent architectures:*

Architecture components: reactive components e.g. quick response, simple rule; deliberative components e.g. planning module, reasoning block; layered architectures e.g. top layer, middle layer

Integration mechanisms: arbitration mechanisms e.g. priority rule, selection method; trade-offs e.g. speed balance, accuracy balance; integration e.g. combined flow, connected steps; workflows e.g. task flow, action route.

*Agentic AI models:*

Model types: goal-based models e.g. simple goals, clear tasks; reactive models e.g. quick responses, basic rules; deliberative models e.g. planning steps, option choice; hybrid models e.g. mixed methods, combined steps

Decision approaches: utility-based models e.g. value choice, best option selection; rule-based models e.g. if-then rules, action rules; behaviour-based models e.g. action patterns, repeated behaviours

Learning and environment influences: learning-augmented models e.g. updated responses, simple adjustments; environment-driven models e.g. input changes, external conditions.

*Model selection considerations:*

Design factors: task complexity e.g. number of steps, difficulty level; uncertainty e.g. missing data, changing information; requirements e.g. task need, system expectation; scalability e.g. system growth, capacity level

Performance constraints: transparency e.g. clear logic, visible process; computational cost e.g. processing time, resource load; performance factors e.g. speed level, accuracy level; design constraints e.g. limits, boundaries, conditions

*Examples of real-world agentic AI architecture applications:*

A range of agentic AI architecture examples e.g.

Reactive agents e.g. Thermostat auto-adjustment, Roborock vacuum obstacle response, Philips Hue motion-trigger lighting

Deliberative agents e.g. SAP supply-chain planners, OpenAI planning models, IBM robotic planning systems

Hybrid agents e.g. ROS robot arbitration, Microsoft Autonomous Systems platform, Honda ASIMO decision integration

Agentic models e.g. AutoGPT, BabyAGI, Microsoft Autogen agents

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Design a simple agent workflow to accomplish a defined task for an identified scenario**

*Agent workflow structure:*

Workflow components: workflow stages e.g. input stage, process stage; input–output flow e.g. data input, result output, transformation path; decision points e.g. branching point, choice node

Control flow: sequencing e.g. step order, action order; conceptual workflow representation e.g. flow diagram, workflow map, process sketch.

*Goal specification and constraints:*

Goal design: primary goals e.g. main objective, required outcome, target condition; secondary goals e.g. optimisation aim, efficiency target; success criteria e.g. completion metric, correctness measure

Constraint factors: operational constraints e.g. timing constraint, rule constraint; resource limitations e.g. memory limit, processing limit, bandwidth limit; environmental assumptions e.g. system condition, context requirement.

*Task decomposition:*

Task structure: sub-tasks e.g. component task, smaller task, dependent task; dependencies e.g. prerequisite link, task order; ordering e.g. sequence rule, execution order, priority chain

Modularity: modular components e.g. processing module, action unit; prioritisation e.g. priority rule, ranking order, task weight.

*Tool and capability selection:*

Tooling: data sources e.g. dataset input, sensor feed; interfaces e.g. user interface, system interface

Capabilities: integration points e.g. connection point, link module; capability boundaries e.g. function limit, system limit, tool restriction.

*Design documentation:*

Design artefacts: workflow diagram e.g. flowchart sketch, block diagram; requirement summary e.g. task brief, goal outline; specification sheet e.g. system spec, component spec

Documentation standards: version control e.g. revision number, update record; annotation elements e.g. comment marker, label note

Traceability records: change log e.g. update entry, revision detail; dependency record e.g. link table, requirement trace

*Examples of real-world agentic AI workflow applications:*

A range of agentic AI workflow software examples e.g.

Control flow e.g. Jira automation sequence, Monday.com workflow steps, Airtable automations

Task structure e.g. Asana task decomposition, GitHub issue breakdown, Trello linked-card flows

Tools and services e.g. GitHub Copilot, Microsoft Copilot Studio, Google AI Agents

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Implement a simple agent workflow to accomplish a defined task for an identified scenario**

*Workflow parameterisation:*

Parameter types: configuration parameters e.g. threshold value, input variable; input variables e.g. data value, user parameter, system input; constraints handling e.g. limit rule, boundary condition

Execution settings: defaults e.g. default value, preset setting; threshold settings e.g. minimum threshold, upper limit, trigger value.

*Execution loop:*

Execution stages: initialisation e.g. start state, setup step; action selection e.g. chosen action, selected step, planned output; termination conditions e.g. stop rule, end state

State updates: state update e.g. updated value, new condition; environment interaction e.g. system feedback, external signal, input change.

*Error handling and resilience:*

Error types: failure states e.g. invalid state, error condition; exception handling e.g. exception rule, error routine, fallback action

Recovery behaviours: recovery mechanisms e.g. restart action, correction step; fallback behaviours e.g. backup option, alternative path, safe mode.

*Monitoring and logging:*

Monitoring elements: event tracking e.g. event record, event marker, event trace; performance indicators e.g. timing metric, success rate; execution logs e.g. log entry, trace record

Decision traces: decision trace e.g. choice record, decision path; state logs e.g. state snapshot, recorded status, system note.

*Cost and latency awareness:*

Performance costs: resource usage e.g. CPU load, memory usage; response time e.g. action delay, output delay, wait duration

Efficiency trade-offs: efficiency considerations e.g. processing balance, resource trade; performance cost e.g. computation cost, overhead value, latency factor.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the core principles of agentic AI systems</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate the effectiveness of different agentic AI principles, architectures and models in supporting agent decision-making.</td></tr>
<tr><td>P1 Explain, using a range of real-world examples, the core principles of agentic AI systems in meeting goals and interacting with environments.<br><br>P2 Explore, using a range of real-world examples, the different agent behaviours used in agentic AI systems.</td><td>M1 Analyse how the core principles of agentic AI systems influence agent behaviour in different environments.</td></tr>
<tr><td colspan="2">LO2 Review the architectures and models used in designing agentic AI systems</td></tr>
<tr><td>P3 Review, using a range of real-world examples, the different architectures used in the design of agentic AI systems.<br><br>P4 Discuss, using a range of real-world examples, the different models used in the design of agentic AI systems and the factors that affect their selection.</td><td>M2 Analyse, using a range of real-world examples, the effectiveness of different agentic AI architectures and models for specific business contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a simple agent workflow to accomplish a defined task for an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and implementation of the agent workflow in relation to its effectiveness in meeting the requirements of the identified scenario. |
| **P5** Design a simple agent workflow to accomplish a defined task for an identified scenario.<br>**P6** Review the design of the agent workflow with yourself as well as a range of organisational stakeholders to identify improvements. | **M3** Optimise, using the results of the review, the design of the agent workflow to accomplish a defined task for an identified scenario. | |
| **LO4** Implement a simple agent workflow to accomplish a defined task for an identified scenario | | |
| **P7** Implement a simple agent workflow to accomplish a defined task for an identified scenario.<br>**P8** Review the implementation of the agent workflow with yourself as well as a range of organisational stakeholders against the identified scenario requirements. | **M4** Evaluate, using the results of the review, the effectiveness of the implemented agent workflow in the delivery of the identified scenario requirements. | |

## Recommended Resources

### Textbooks

Albrecht, S.V., Christianos, F. & Schäfer, L. (2024) *Multi-Agent Reinforcement Learning*. Cambridge: MIT Press

Barocas, S., Hardt, M. & Narayanan, A. (2023) *Fairness and Machine Learning*. Cambridge: MIT Press.

Poole, D.L. & Mackworth, A.K. (2023) *Artificial Intelligence: Foundations of Computational Agents*. 3rd edn. Cambridge: Cambridge University Press.

Russell, S. and Norvig, P. (2021) *Artificial Intelligence: A Modern Approach*. 4th edn. Harlow: Pearson.

Wooldridge, M. (2021) *A Brief History of Artificial Intelligence: What It Is, Where We Are, and Where We Are Going*. London: Penguin Random House.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| cloud.google.com/discover/what-are-ai-agents | Google AI<br>What is an AI agent?<br>(General reference) |
| aws.amazon.com/ai/frontier-agents/ | Amazon Web Services<br>Frontier agents<br>Autonomous, massively scalable AI agents that work for hours without requiring intervention<br>(Reference) |
| www.geeksforgeeks.org/artificial-intelligence/agentic-ai-architecture/ | GeeksforGeeks<br>Agentic AI Architecture<br>(Resource) |
| vectorize.io/blog/designing-agentic-ai-systems-part-1-agent-architectures | Vectorize<br>Designing Agentic AI Systems, Part 1: Agent Architectures<br>(Article) |

| | |
|---|---|
| fme.safe.com/guides/ai-agent-architecture/ | Safe Software<br>AI Agent Architecture: Tutorial & Examples<br>(Tutorial) |
| www.coursera.org/specializations/building-ai-agents-and-agentic-workflows | Coursera<br>Building AI Agents and Agentic Workflows Specialization<br>(Course) |
| harvard-edge.github.io/cs249r_fall2025/course/resources | Harvard University<br>Agentic AI for computer systems design<br>(Resource) |

**Journals and articles**

Amershi, S. *et al*. (2019) 'Guidelines for human-AI interaction', in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Available at: doi.org/10.1145/3290605.3300233.

Hu, S., Lu, C. and Clune, J. (2024) 'Automated Design of Agentic Systems', *arXiv preprint*. Available at: doi.org/10.48550/arXiv.2408.08435.

Wooldridge, M. and Jennings, N.R. (1995) 'Intelligent agents: theory and practice', *The Knowledge Engineering Review*, 10(2), pp. 115–152. Available at: doi.org/10.1017/S0269888900008122.

Wooldridge, M. and Jennings, N.R. (1996) 'Agent theories, architectures, and languages: A survey', *Lecture Notes in Computer Science*, Vol. 890. Springer. Available at: doi.org/10.1007/3-540-58855-8_1.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 420: Core AI*

*Unit 421: AI Automation and Digital Productivity.*

# Unit 424: Creative AI Coding

**Unit code:** J/652/3276

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Artificial Intelligence is becoming a routine part of modern software development, influencing how digital tools are designed, built and used. As creative and technical fields continue to overlap, AI-supported coding is now used across industries ranging from digital media to business applications. Understanding how these tools work, and how they can support or enhance traditional coding practices, is now an essential part of the skill set expected from emerging developers.

The aim of this unit is to give students a clear and practical understanding of how AI-assisted programming and creative coding techniques can be used together. The unit introduces the principles behind these approaches, helping students understand where AI can provide structure, automation and support, and where exploratory or expressive coding approaches may be more suitable.

The unit covers three key areas: the core concepts of AI-assisted programming and vibe coding; the principles of prompt engineering and how prompts influence generated outputs; and the development of a simple programmed solution using both structured and exploratory methods. Students also examine the benefits and limitations of these approaches and consider how accuracy, reliability and documentation vary across different workflows.

On successful completion of this unit, students will be able to apply AI-supported coding tools, design effective prompts and combine different programming approaches to meet the needs of a given task. These skills support progression into a range of roles, including Junior Software Developer, Creative Technologist, Digital Developer, AI Development Assistant, Interactive Media Technician and Automation Support Developer. Students will also gain valuable experience relevant to further study in computing, creative technologies and emerging digital disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core concepts of AI-assisted programming and vibe coding

LO2 Discuss the fundamental principles of prompt engineering

LO3 Develop a simple programmed solution for an identified requirement using both AI-assisted programming and vibe coding across the entire software life cycle

LO4 Compare the benefits and limitations of the programmed solution obtained through vibe coding and AI-assisted programming.

## Essential Content

### LO1 **Explain the core concepts of AI-assisted programming and vibe coding**

*AI-assisted programming concepts:*

Artificial intelligence in software development: artificial intelligence e.g. model inference, pattern processing, optimisation; automation e.g. repetitive tasks, code generation; augmentation e.g. assisted debugging, guided completion

Human-in-the-Loop (HITL) Coding: oversight e.g. manual review, human correction; responsibility e.g. output checking, risk awareness; limitation awareness e.g. accuracy limits, model gaps

Model-assisted problem solving: code suggestion e.g. function templates, logic scaffolds, syntax recommendations; completion tools e.g. autocompletion, inline hints; support features e.g. error detection, refactoring prompts

Risks and constraints: dependency risks e.g. over-reliance, reduced understanding; accuracy limitations e.g. hallucination, incorrect logic; security considerations e.g. data exposure, unsafe patterns.

*Creative and expressive coding concepts:*

Creative computation: creativity e.g. generative ideas, aesthetic choices; experimentation e.g. iterative trials, unconventional methods; examples

Exploratory coding e.g. trial-and-error, discovery phases;

Creative–technical balance: relationship considerations e.g. creativity vs. structure, innovation vs. precision; refinement e.g. iterative adjustment, improvement cycles; emergent outcomes e.g. unexpected solutions, novel patterns.

*Vibe coding principles and context:*

Informal approaches: intuitive development e.g. rapid edits, informal notes; creative flow e.g. spontaneous coding, real-time shaping; emergent problem-solving e.g. pattern discovery, adaptive logic

Rapid experimentation: experimentation e.g. quick trials, test sketches, small prototypes; refinement e.g. micro-adjustments, fast iteration; informal structure e.g. flexible formats, evolving logic

Application contexts: creative contexts e.g. digital art, interactive media; experimental systems e.g. generative visuals, reactive environments; contrast with structure e.g. rigid rulesets, formal design

*Examples of real-world AI coding applications:*

A range of real-world AI coding tools e.g.

AI-assisted programming tools: GitHub Copilot e.g. code suggestions, autocompletion; Amazon CodeWhisperer e.g. code generation, syntax support; Google Gemini Code Assist e.g. coding help, debugging support

Creative AI coding tools: Runway Gen-2 e.g. generative video, creative media; TouchDesigner AI Nodes e.g. interactive visuals, generative design; Google Magenta e.g. music generation, creative experimentation

Vibe coding and rapid development tools: CodePen AI e.g. rapid prototyping, front-end experiments; StackBlitz AI e.g. browser coding, development testing; Replit Ghostwriter e.g. coding assistance, live editing

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Discuss the fundamental principles of prompt engineering**

*Prompt structure and composition:*

Intent: intent e.g. task definition, coding goal; task framing e.g. problem outline, output need, constraint summary; contextual information e.g. user role, domain background; constraints e.g. formatting rules, data limits; instructions e.g. coding steps, processing actions, required checks

Clarity and specificity: clarity e.g. direct phrasing, concise wording; specificity e.g. parameter detail, output format; ambiguity reduction e.g. boundary conditions, exclusion notes; relevance e.g. problem alignment, domain fit, coding suitability

Structural components: structure e.g. ordered steps, code blocks; formatting e.g. indentation, layout rules; examples e.g. sample input, expected output, usage pattern.

*Prompt parameters and controls:*

Model controls: system prompts e.g. behaviour setup, rule definition; temperature e.g. randomness control, creativity level, variability; reproducibility e.g. stable prompts, unchanged parameters

Parameter settings: tokens e.g. token limits, output length; sampling settings e.g. top-p values, nucleus sampling, diversity tuning; response length e.g. short outputs, extended responses

Output regulation: output stability e.g. fixed structure, controlled variation; iteration consistency e.g. aligned tone, similar outputs; parameter limitations e.g. max tokens, restricted controls.

*Human interaction with AI systems:*

Iterative processes: prompting cycles e.g. revision steps, iteration rounds; refinement e.g. wording improvement, structure adjustment; testing e.g. validation runs, comparison outputs

Responsible use: ethics e.g. fairness, safety; accountability e.g. traceability, responsibility assignment, documented reasoning; bias awareness e.g. bias detection, mitigation strategy

Model awareness: model limitations e.g. hallucination, outdated knowledge; capability boundaries e.g. unsupported tasks, restricted functions; risk awareness e.g. incorrect logic, fabricated details

*Examples of real-world AI prompting applications:*

A range of real-world AI prompt tools e.g.

Prompt design tools: ChatGPT Prompt Generator e.g. prompt creation, task structuring; LearnPrompting tools e.g. prompt building, guidance support; Google Gemini prompt examples e.g. prompt formats, example outputs

Model control tools: OpenAI Playground e.g. prompt testing, parameter control; Anthropic Claude Console e.g. prompt tuning, response control; Google Vertex AI prompt controls e.g. parameter settings, output tuning

Prompt refinement tools: ChatGPT iterative prompting e.g. prompt revision, output refinement; Replit Prompt Refiner e.g. prompt improvement, testing; Microsoft Prompt Flow e.g. workflow prompts, iteration control

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Develop a simple programmed solution for an identified requirement using both AI-assisted programming and vibe coding across the entire software life cycle**

*General software life cycle:*

Life cycle Stages: design e.g. planning, structuring; development e.g. coding, assembling; testing e.g. validation checks, corrections

Documentation and maintenance: documentation e.g. summaries, notes; maintenance considerations e.g. updates, revisions; workflow comparison e.g. structured life cycle, exploratory life cycle.

*Programming requirement identification:*

Problem definition: problem definition e.g. user need, task goal; functional requirements e.g. inputs, outputs, behaviours; scope limitation e.g. boundaries, constraints; examples: Jira AI Issue Assistant, Azure DevOps AI Work Item Planner, Trello AI

Non-functional considerations: performance factors e.g. speed, efficiency; usability aspects e.g. clarity, accessibility; suitability e.g. task fit, context alignment.

*AI-assisted programming across the software life cycle:*

Design support: design guidance e.g. flow suggestions, structure patterns; planning elements e.g. modular outline, function grouping; logic scaffolding e.g. branching hints, iteration patterns

Code generation assistance: code generation e.g. snippets, templates; syntax patterns e.g. formatting rules, variable use; debugging support e.g. error prompts, correction hints

Testing and documentation: testing assistance e.g. test cases, validation checks; documentation support e.g. comments, summaries; maintenance awareness e.g. versioning points, update notes.

*Vibe coding across the software life cycle:*

Expressive development: exploratory flow e.g. rapid edits, improvisation; creative experimentation e.g. small prototypes, trial sequences

Iterative refinement: refinement e.g. micro-adjustments, continuous shaping; informal structure e.g. flexible formats, evolving logic; exploratory debugging e.g. intuition-based checks, ad-hoc fixes

Creative workflow context: creative contexts e.g. digital media, interactive art; experimental systems e.g. generative visuals, dynamic behaviours; workflow contrast e.g. structured vs. open-ended approaches

*Examples of real-world AI assisted programming applications:*

A range of real-world AI assisted programming tools e.g.

Development tools: GitHub Codespaces e.g. cloud coding, development environments; Visual Studio e.g. code editing, debugging; JetBrains AI Assist e.g. coding support, suggestions

AI coding tools: GitHub Copilot e.g. code generation, autocompletion; CodeWhisperer e.g. code suggestions, syntax help; Gemini Code Assist e.g. debugging, coding support

Creative coding tools: Replit Ghostwriter e.g. coding assistance, rapid iteration; CodePen AI e.g. prototyping, experiments; Cursor AI e.g. interactive coding, live editing

Project tools: Jira AI e.g. task tracking, requirements; Azure DevOps AI e.g. work planning, project tracking; Trello AI e.g. task management, workflow organisation

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Compare the benefits and limitations of the programmed solution obtained through vibe coding and AI-assisted programming**

*Benefits of AI-assisted programming approaches:*

Efficiency gains: efficiency e.g. rapid prototyping, fast generation; productivity e.g. workflow acceleration, reduced manual effort; consistency e.g. uniform patterns, stable syntax

Support features: error detection e.g. flagged issues, correction hints; accessibility e.g. novice support, guided structure; scalability e.g. larger tasks, expanded functionality.

*Limitations and risks of AI-assisted programming approaches:*

Accuracy and reliability issues: bias e.g. skewed outputs, training imbalance; hallucination risks e.g. invented details, incorrect logic; security considerations e.g. unsafe patterns, data exposure

Operational concerns: intellectual property e.g. unclear origins, licence uncertainty; accountability issues e.g. unclear reasoning, responsibility gaps; over-reliance e.g. reduced understanding, automation dependence.

*Benefits of vibe coding approaches:*

Creative flexibility: creativity e.g. unconventional solutions, expressive outcomes; innovation e.g. novel effects, unique structures; adaptability e.g. fluid logic, evolving patterns

Exploratory strengths: exploratory behaviour e.g. trial-and-error, discovery flow; flexibility e.g. open structure, rapid alterations; responsiveness e.g. real-time adjustments, emergent shaping.

*Limitations of vibe coding approaches:*

Structural weaknesses: reliability concerns e.g. inconsistent behaviour, unstable logic; maintainability issues e.g. unclear structure, fragmented code; validation limitations e.g. minimal testing, informal checks

Scalability and documentation: documentation weaknesses e.g. sparse notes, missing explanations; scalability constraints e.g. difficulty expanding, structural fragility; long-term issues e.g. hard updates, unclear evolution.

*Comparative validation and evaluation considerations:*

Quality assessment: accuracy checking e.g. test cases, comparison runs; reliability e.g. repeat behaviour, stable output; testing approaches e.g. validation checks, error review

Professional responsibility: documentation quality e.g. clarity, structure; contextual suitability e.g. task fit, requirement match; ethical considerations e.g. accountability, safe practice

*Examples of real-world vibe coding and AI-assisted coding applications:*

A range of real-world AI tools e.g.

AI coding and productivity tools: GitHub Copilot e.g. code suggestions, autocompletion; AWS CodeCatalyst AI e.g. development workflows, project automation; JetBrains AI Assistant e.g. coding support, refactoring

AI coding limitations and evaluation tools: OpenAI GPT-4o code mode e.g. code generation, logic support; Claude 3.5 Sonnet e.g. reasoning, code generation; Meta CodeLlama e.g. code completion, programming assistance

Creative AI tools: Runway Gen-2 e.g. generative video, media creation; TouchDesigner ML Toolkit e.g. interactive visuals, generative design; Hydra AI e.g. live coding, creative development

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explain the core concepts of AI-assisted programming and vibe coding | | **LO1 and LO2**<br>**D1** Evaluate the effectiveness of AI-assisted programming approaches and prompt engineering in supporting creative and technical coding practices. |
| **P1** Explain, using real-world examples, the core concepts of AI-assisted programming in relation to software development contexts.<br>**P2** Discuss, using real-world examples, the core concepts of vibe coding in relation to alternative creative coding practices. | **M1** Analyse the impact of AI-assisted programming and vibe coding in a software development environment. | |
| **LO2** Discuss the fundamental principles of prompt engineering | | |
| **P3** Discuss, using real-world examples, the principles of prompt structure and prompt parameters in prompt engineering.<br>**P4** Explain how humans can interact with AI systems by the use of engineered prompts. | **M2** Analyse how prompt engineering principles influence the reliability and reproducibility of AI-generated outputs. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Develop a simple programmed solution for an identified scenario using both AI-assisted programming and vibe coding across the entire software life cycle | | **LO3 and LO4**<br>**D2** Justify the selection of either vibe coding or AI assisted programming as a suitable methodology for delivering robust, reliable and secure programmed solutions across the software life cycle. |
| **P5** Develop a simple programmed solution that addresses the requirements of an identified scenario using AI-assisted programming.<br>**P6** Develop a simple programmed solution that addresses the requirements of an identified scenario using vibe coding techniques. | **M3** Analyse the effectiveness of AI-assisted programming and vibe coding approaches used in developing the solution for the requirements of the identified scenario. | |
| **LO4** Compare the benefits and limitations of the programmed solution obtained through vibe coding and AI-assisted programming | | |
| **P7** Review the differences in the code, documentation and testing delivered by the AI-assisted programming method and the vibe coding method.<br>**P8** Compare the benefits and limitations of the programmed solution obtained through vibe coding and AI-assisted programming in meeting the identified requirement across the software life cycle. | **M4** Evaluate the differences in reliability, validation, and documentation between AI-assisted programming and vibe coding solutions across the software life cycle. | |

## Recommended Resources

### Textbooks


Albrecht, S.V., Christianos, F. and Schäfer, L. (2024) *Multi-Agent Reinforcement Learning*. Cambridge, MA: MIT Press.

Barocas, S., Hardt, M. and Narayanan, A. (2023) *Fairness and Machine Learning*. Cambridge, MA: MIT Press.

Boden, M.A. (2024) *The Creative Mind: Myths and Mechanisms*. Updated edn. Abingdon: Routledge.

Geroimenko, V. (2025) *The Essential Guide to Prompt Engineering: Key Principles, Techniques, Challenges, and Security Risks.* Cham: Springer.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*. Cambridge, MA: MIT Press.

Mollick, E. (2024) *Co-Intelligence: Living and Working with AI*. London: Penguin Business.

O'Reilly, T. (2023) *Practical Prompt Engineering for Developers*. Sebastopol, CA: O'Reilly Media.

Subramani, B. (2025) *The AI-DLC Handbook: Reimagining Software Development with AI*. Notion Press.

Taulli, T. (2024) *AI-Assisted Programming: Better Planning, Coding, Testing, and Deployment.* Sebastopol, CA: O'Reilly Media.

White, J., Hays, S., Fu, Q., Spencer-Smith, J. and Schmidt, D.C. (2024) *Prompt Engineering for Generative AI*. Sebastopol, CA: O'Reilly Media.


### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| openai.com/research/index/ | OpenAI Research<br>AI-assisted programming, prompt engineering concepts and limitations<br>(Research Hub) |
| deepmind.google/research/ | Google DeepMind<br>Human-AI interaction, generative models and creative systems<br>(Research Hub) |

| | |
|---|---|
| www.freecodecamp.org/news/how-to-become-an-expert-in-ai-assisted-coding-a-handbook-for-developers/ | freeCodeCamp – AI-assisted coding<br>AI-assisted programming concepts, developer workflows, Human-in-the-Loop coding<br>(Tutorial/Article) |
| learn.microsoft.com/en-us/visualstudio/ide/ai-assisted-development-visual-studio | Microsoft Learn<br>AI-assisted development in Visual Studio<br>AI-assisted programming tools, code generation, debugging support<br>(Tutorial/Documentation) |
| learnprompting.org/docs/introduction | Learn Prompting<br>Prompt structure, clarity, iteration, and model interaction principles<br>(Learning guide) |
| github.com/resources/articles/tools-and-techniques-for-effective-code-documentation | GitHub – Code documentation techniques<br>Software life cycle documentation, testing and maintenance practices<br>(Article) |
| laravel.com/docs/12.x/ai | Laravel Documentation – AI-assisted development<br>AI-assisted programming across development stages, structured workflows<br>(Framework documentation) |
| aws.amazon.com/blogs/devops/ai-driven-development-life-cycle/ | AWS – AI-Driven Development Life cycle<br>Benefits and limitations of AI-assisted development across SDLC stages<br>(Blog) |
| www.ibm.com/think/topics/ai-life cycle | IBM – AI life cycle overview<br>Reliability, limitations, governance, and evaluation of AI systems<br>(Explainer) |
| cloud.google.com/discover/what-is-vibe-coding | Google Cloud<br>What is vibe coding?<br>(Article) |

## Journals and articles

Amershi, S. *et al*. (2019) 'Guidelines for Human-AI Interaction', *Communications of the ACM*, 62(10), pp. 52–61. Available at: doi.org/10.1145/3290605.3300233.

Liu, P. *et al*. (2023) 'Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing', *ACM Computing Surveys*, 55(9), Article 195. Available at: doi.org/10.1145/3560815.

Moruzzi, C. (2025) 'Artificial Intelligence and Creativity', *Philosophy Compass*, 20(3), Article e70030. Available at: doi.org/10.1111/phc3.70030.

Vaithilingam, P., Zhang, T. and Glassman, E.L. (2022) 'Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models', *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. Available at: doi.org/10.1145/3491101.3519665.

## Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 420: Core AI*

*Unit 421:  AI Automation and Digital Productivity*

*Unit 423: Agentic AI.*

# Unit 425: AI Programming

**Unit code:** K/652/3277

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Artificial Intelligence (AI) is becoming an essential part of modern digital systems, shaping how organisations analyse information, make decisions and deliver services. AI-enabled tools now appear in areas such as customer support, finance, healthcare, manufacturing and everyday applications on personal devices. As these technologies continue to grow, there is an increasing need for practitioners who understand how AI techniques work and how they can be applied responsibly within software systems.

The aim of this unit is to introduce students to the core principles and practical foundations of AI programming. The unit focuses on building a clear understanding of the techniques that underpin AI-based solutions, including algorithms, data structures, search methods and basic knowledge representation. Students will also explore how these principles support the development of simple, functional AI programs suitable for real-world problem-solving.

Students will learn how to implement a range of introductory AI techniques in code, including rule-based systems, decision-making models and basic machine learning approaches. The unit guides students through designing and developing a small AI program for a specific scenario, using appropriate tools, programming environments and development practices. Students will then review the performance of their program, considering its accuracy, efficiency and suitability for the intended purpose.

On successful completion of this unit, students will be able to explain key AI programming concepts, apply core techniques in code and develop basic AI solutions that meet defined requirements. These skills are relevant to a range of early-career roles in the digital and technology sector, including Junior Software Developer, AI Technician, Data Technician, Machine Learning Assistant, Junior Automation Developer and Trainee Systems Developer. The unit also provides a strong foundation for further study in artificial intelligence, data science and software engineering.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the foundational AI programming principles

LO2 Implement a range of core AI techniques in code

LO3 Develop a basic AI program using suitable programming tools and techniques for an identified scenario

LO4 Review the performance of the AI program developed for an identified scenario against key requirements.

## Essential Content

LO1 **Explain the foundational AI programming principles**

*Algorithms and problem-solving approaches:*

Algorithm design: deterministic algorithms e.g. fixed steps, predictable behaviour; stepwise logic e.g. ordered steps, logical flow; problem decomposition e.g. task breakdown, smaller sub-tasks

Algorithm structure: correctness e.g. valid outputs, expected results; instruction sequence e.g. step order, execution flow.

*Data structures for AI applications:*

Linear structures: arrays e.g. indexed data, fixed order; lists e.g. flexible items, ordered elements

Hierarchical structures: trees e.g. parent–child links, branching structure; graphs e.g. connected nodes, simple networks

Hashed structures: hash-based structures e.g. key–value pairs, fast lookup.

*Search strategies and techniques:*

Search representation: state–space representation e.g. possible states, move options; nodes e.g. positions, decision points.

Uninformed search: depth-first search e.g. deep exploration, backtracking; breadth-first search e.g. level exploration, ordered scanning

Exploration process: search paths e.g. explored routes, node sequences; transitions e.g. state changes, movement steps.

*Knowledge representation methods:*

Symbolic forms: rules e.g. if-then structures, logical conditions; facts e.g. known data, stored information

Logical forms: logical representations e.g. structured expressions, logic rules; statements e.g. declared conditions, expressed facts.

Structural elements: attributes e.g. item features, simple properties; relationships e.g. links, connections.

*Examples of real-world AI applications:*

A range of AI software examples e.g.

Algorithms e.g. Google Search ranking algorithms, IBM Watson Discovery, Microsoft Bing Web Indexer

Data structures e.g. NumPy arrays, PyTorch tensors, Java Arraylists

Uninformed searching e.g. Googlebot web crawler, file-system tree scanners, network vulnerability scanners

Symbolic forms e.g. MYCIN, CLIPS, Drools (AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Implement a range of core AI techniques in code**

*Rule-based systems:*

Condition rules: conditions e.g. comparisons, thresholds; action rules e.g. outputs, responses

Rule sets: rule sets e.g. grouped rules, knowledge bases; inference steps e.g. simple reasoning, step processing

Chaining methods: forward chaining e.g. data-driven steps, rule firing; backward chaining e.g. goal checking, target matching

Control flow: rule matching e.g. checking conditions, selecting rules; control flow e.g. rule order, execution sequence.

*Decision-making models:*

Decision trees: decision trees e.g. branching paths, tests; branches e.g. choices, next steps

Nodes: nodes e.g. decision points, test points; splitting criteria e.g. simple rules, key features

Outputs: classification outputs e.g. predicted class, category choice; decision structure e.g. logic outline, rule pattern.

*Introductory machine learning techniques:*

Supervised learning: supervised learning e.g. labelled examples, known answers; training data e.g. sample inputs, practice cases

Testing data: testing data e.g. unseen examples, evaluation set; model fitting e.g. adjusting values, learning patterns

Model outputs: prediction outputs e.g. expected result, suggested label; data samples e.g. small datasets, simple records; feature inputs e.g. basic attributes, input values.

*Implementation principles:*

Code structure: modules e.g. files, components; functions e.g. reusable blocks, methods; classes e.g. objects, templates.

Programming standards: syntax e.g. correct statements, expressions; conventions e.g. naming rules, layout

Code quality: readability e.g. clear naming, comments; maintainability e.g. modular design, updates; documentation e.g. comments, descriptions

Development practices: testing e.g. test cases, sample runs; debugging e.g. error checking, fault fixing

Library usage: numerical libraries e.g. NumPy, math tools; machine learning libraries e.g. scikit-learn, basic ML tools.

*Examples of real-world AI applications:*

A range of AI software examples e.g.

Rules e.g. Drools rules engine, CLIPS expert system, IBM operational decision manager

Decision trees e.g. scikit-learn DecisionTreeClassifier, RapidMiner decision tree, Spark ML decision tree

Supervised learning e.g. scikit-learn supervised models, TensorFlow Keras classifiers, PyTorch supervised training loops

Code structure e.g. Python module-based ML projects, Java class-based AI utilities, Node.js modular AI scripts

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Develop a basic AI program using suitable programming tools and techniques for an identified scenario**

*AI program architecture:*

Input handling: user input e.g. text entry, selections; data input e.g. files, datasets

Processing logic: decision logic e.g. rule evaluation, condition checks; model execution e.g. prediction steps, calculations; control flow e.g. sequencing, branching

Output generation: results e.g. predictions, classifications; feedback e.g. messages, indicators

Modular structure: modules e.g. separate components, reusable parts; integration steps e.g. linking modules, combining functions.

*Scenario-driven application design:*

Problem definition: objectives e.g. task goals, requirements; constraints e.g. limits, conditions

Contextual factors: user needs e.g. expected behaviour, usability; environment e.g. deployment setting, system context

Data characteristics: data type e.g. numerical values, categories; data size e.g. small datasets, sample data

Application purpose: expected outputs e.g. simple recommendation, classification result; task goals e.g. clear aim, specific problem.

*Programming tools and environments:*

Development environments: IDEs e.g. code editors, notebooks; execution tools e.g. interpreters, runtimes

Programming languages: scripting languages e.g. Python, simple scripts; general-purpose languages e.g. high-level languages, structured code

Libraries and frameworks: data libraries e.g. data handling tools, tables; AI libraries e.g. basic ML tools, model utilities

Tool integration: library imports e.g. packages, modules; workflow setup e.g. development pipelines, execution steps.

*Data preparation and handling:*

Data cleaning: error removal e.g. missing values, duplicates; data correction e.g. formatting fixes, value updates

Data splitting: training data e.g. learning sets, samples; testing data e.g. evaluation sets, unseen data

Feature handling: feature selection e.g. picking useful attributes, removing unnecessary ones; data representation e.g. tables, rows and columns.

*Examples of real-world AI applications:*

A range of AI software examples e.g.

Input handling: Google Forms data capture, Python argparse input handlers, Microsoft Power Apps input field

Libraries and frameworks: pandas, NumPy, scikit-learn

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Review the performance of the AI program developed for an identified scenario against key requirements**

*Key requirements:*

Functional requirements: expected behaviour e.g. correct outputs, task completion; system actions e.g. responses, processing steps

Performance expectations: speed e.g. response time, execution rate; accuracy e.g. correct results, valid predictions

Scenario constraints: data limits e.g. dataset size, input range; environment limits e.g. hardware capacity, software restrictions

Purpose alignment: intended use e.g. user needs, task goals; output relevance e.g. useful results, appropriate responses.

*Performance evaluation concepts:*

Correctness: valid outputs e.g. expected results, correct decisions; logic accuracy e.g. rule matching, condition handling;

Efficiency: execution time e.g. runtime speed, processing delay; resource usage e.g. memory use, processor load

Bias considerations: bias e.g. training data bias, sampling bias, representation bias; model behaviour e.g. skewed predictions, unfair outputs, systematic error patterns; fairness considerations e.g. equal representation, balanced datasets, ethical evaluation

Robustness: stability e.g. consistent behaviour, error handling; reliability e.g. repeatable results, predictable operation

Output consistency: result stability e.g. similar inputs, similar outputs; variation control e.g. limited fluctuation, controlled change.

*Validation and testing approaches:*

Testing datasets: sample data e.g. test records, example inputs; unseen data e.g. new cases, evaluation inputs

Validation techniques: result checking e.g. expected outputs, comparison; split testing e.g. training data, testing data

Error identification: faults e.g. logic errors, incorrect rules; anomalies e.g. unexpected outputs, inconsistencies

Debugging concepts: code tracing e.g. step execution, value tracking; issue correction e.g. fixes, adjustments.

*Evaluation metrics and improvement considerations:*

Accuracy measures: accuracy scores e.g. percentage correct, success rate; error rates e.g. misclassification rate, incorrect output

Metric suitability: metric selection e.g. task relevance, simplicity; problem fit e.g. classification tasks, decision tasks

Optimisation considerations: performance tuning e.g. code efficiency, logic refinement; model adjustment e.g. parameter changes, rule updates

Solution limitations: scope limits e.g. simple scenarios, small datasets; improvement areas e.g. feature expansion, logic enhancement.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the foundational AI programming principles</td><td rowspan="5">LO1 and LO2<br><br>D1 Justify the suitability of implemented AI techniques and programming principles for solving the identified problem context.</td></tr>
<tr><td>P1 Explain, using real-world code examples, the foundational algorithms and data structures used in the programming of AI systems.<br><br>P2 Review, using real-world code examples, the search strategies and knowledge representation methods used in the development of AI systems.</td><td>M1 Analyse how foundational AI programming principles support the development of AI systems.</td></tr>
<tr><td colspan="2">LO2 Implement a range of core AI techniques in code</td></tr>
<tr><td>P3 Implement the core AI techniques of decision-making and rule-based systems in code using professional implementation principles for an identified problem context.<br><br>P4 Develop a simple code solution using foundational machine learning techniques and professional implementation principles for an identified problem context.</td><td>M2 Evaluate the effectiveness of implemented AI techniques in meeting identified functional requirements.</td></tr>
</table>

 © Pearson Education Limited 2026

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Develop a basic AI program using suitable programming tools and techniques for an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of the AI program in relation to its effectiveness in meeting the requirements of the identified scenario. |
| **P5** Design a basic AI program, using suitable design tools and techniques, that will meet the requirements of an identified scenario.<br>**P6** Develop a basic AI program, using suitable programming tools and techniques, that addresses the requirements of an identified scenario. | **M3** Assess how programming tools and techniques contribute to the performance of the developed AI program. | |
| **LO4** Review the performance of the AI program developed for an identified scenario against key requirements | | |
| **P7** Implement a test plan to determine the performance of the developed AI program against key requirements.<br>**P8** Review the performance of the developed AI program with a range of organisational stakeholders against the identified scenario requirements. | **M4** Evaluate, using the results of the testing and review, the performance of the AI program, identifying strengths and limitations. | |

## Recommended Resources

### Textbooks

Deitel, P. and Deitel, H. (2025) *Java for Programmers: With Generative AI*. 5th edn. Pearson.

Géron, A. (2022) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd edn. Sebastopol, CA: O'Reilly Media.

Kubat, M. (2021) *An Introduction to Machine Learning*. 3rd edn. Springer.

Poole, D.L. and Mackworth, A.K. (2023) *Artificial Intelligence: Foundations of Computational Agents*. 3rd edn. Cambridge: Cambridge University Press.

Russell, S. and Norvig, P. (2022) *Artificial Intelligence: A Modern Approach*. 4th edn. Pearson.

Taulli, T. (2024) *AI-Assisted Programming: Better Planning, Coding, Testing, and Deployment*. Sebastopol, CA: O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/artificial-intelligence/ai-algorithms/ | GeeksforGeeks<br>Artificial Intelligence (AI) Algorithms<br>(General reference) |
| www.artint.info/ | Artificial Intelligence 3E<br>Foundations of Computational Agents<br>(General reference) |
| www.freecodecamp.org/learn/machine-learning-with-python/ | FreeCodeCamp<br>Machine Learning with Python<br>(Tutorial) |
| www.kaggle.com/ | Kaggle<br>Level up with the largest AI and ML community<br>(Datasets) |
| numpy.org/ | NumPy<br>The fundamental package for scientific computing with Python<br>(General reference) |

| | |
|---|---|
| www.codecademy.com/learn/learn-c-plus-plus | Codecademy<br>Learn C++<br>(Online Course) |
| pandas.pydata.org/ | pandas<br>pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.<br>(Resource Hub) |
| scikit-learn.org/stable/ | scikit-learn<br>Machine Learning in Python<br>(Resource Hub) |
| www.geeksforgeeks.org/machine-learning/machine-learning-with-java/ | GeeksforGeeks<br>Machine learning with Java<br>(General reference) |

### Journals and articles

Delgrande, J.P. *et al*. (2024) 'Current and Future Challenges in Knowledge Representation and Reasoning, *Dagstuhl Manifestos,* 10(1), pp. 1–61. Available at: doi.org/10.4230/DagMan.10.1.1.

Kokol, P. (2024) 'The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature', *Information*, 15(6), 354. Available at: doi.org/10.3390/info15060354.

Rainio, O., Teuho, J. and Klén, R. (2024) 'Evaluation metrics and statistical tests for machine learning', *Scientific Reports*, 14, 6086. Available at: doi.org/10.1038/s41598-024-56706-x.

Rizzo, L., Verda, D., Berretta, S. and Longo, L. (2024) 'A Novel Integration of Data-Driven Rule Generation and Computational Argumentation for Enhanced Explainable AI', *Machine Learning and Knowledge Extraction*, 6(3), pp. 2049–2073. Available at: doi.org/10.3390/make6030101.

### Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 420: Core AI*

*Unit 421:  AI Automation and Digital Productivity*

*Unit 423: Agentic AI.*

# Unit 426: Digital Project Management

**Unit code:** L/652/3278

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Digital projects play a central role in how organisations design, deliver, and improve products, services and systems across a wide range of sectors. From software development and digital services to business change and technology-enabled operations, projects increasingly rely on digital tools, platforms and collaborative working practices. As a result, there is growing demand for professionals who can plan, manage, and review projects effectively within digital environments.

The aim of this unit is to develop students' understanding of digital project management principles and how they are applied in real-world project contexts. Students will explore how projects are structured, governed and controlled, and how digital tools are used to support planning, co-ordination, monitoring, and communication throughout the project life cycle. The unit focuses on developing both knowledge and practical skills that reflect current industry expectations.

The unit covers key aspects of digital project management, including project life cycles, governance structures, documentation, scheduling, and stakeholder roles. Students will apply project planning techniques to an identified computational scenario, developing a structured project plan that considers requirements, tasks, resources, risks, and constraints. The unit also examines the use of digital tools to track progress, manage issues and risks, and communicate performance. Finally, students review project outcomes against agreed success criteria, considering quality, performance, and stakeholder feedback.

On successful completion of this unit, students will be able to contribute effectively to digital project activities within organisational settings. They will have the skills needed to support project planning, use digital tools to manage and monitor delivery, and evaluate project outcomes in a professional manner. This provides a foundation for progression into roles such as Project Co-ordinator, Junior Project Manager, Digital Operations Assistant, Business Analyst or Project Support Officer, as well as further study in project management and related digital disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core principles of digital project management

LO2 Apply digital project planning techniques to develop a project plan for an identified organisational scenario

LO3 Implement a range of digital tools to monitor and manage project progress for an identified organisational scenario

LO4 Review the project outcomes against identified success criteria.

## Essential Content

LO1 **Explain the core principles of digital project management**

*Digital project life cycle models:*

Methodologies: linear models e.g. construction projects, infrastructure upgrades; agile methodology e.g. app development, digital services; hybrid approach e.g. business change projects, enterprise systems

Life cycle stages: project initiation e.g. project approval, feasibility review; project planning e.g. schedule development, resource planning; project execution e.g. task delivery, team co-ordination; project closure e.g. final reporting, handover activities

Structure and sequencing: life cycle phases e.g. initiation phase, delivery phase; phase sequencing e.g. planned milestones, phase transitions; project monitoring e.g. progress tracking, status checking.

*Project governance structures:*

Decision and authority: decision frameworks e.g. approval hierarchies, decision models; governance roles e.g. project sponsor, steering group; stakeholder authority e.g. budget holders, senior managers

Control and accountability: responsibility allocation e.g. role assignment, task ownership; accountability structures e.g. reporting lines, performance accountability; escalation routes e.g. issue escalation, risk escalation

Compliance and alignment: approval processes e.g. stage-gate approval, sign-off processes; compliance requirements e.g. organisational policies, regulatory standards; organisational standards e.g. project frameworks, internal guidelines.

*Project documentation and controls:*

Core project records: project charters e.g. project purpose statements, scope outlines; business cases e.g. cost–benefit analysis, justification documents; project plans e.g. schedules, resource plans

Risk and change records: risk registers e.g. identified risks, mitigation actions; issue logs e.g. project issues, resolution actions; change control records e.g. change requests, approval records

Communication and compliance: communication plans e.g. stakeholder updates, reporting schedules; version control e.g. document versioning, revision tracking; audit records e.g. compliance checks, review evidence.

*Scheduling and stakeholder roles:*

Scheduling elements: milestones e.g. key deadlines, delivery points; task dependencies e.g. predecessor tasks, successor tasks; project timelines e.g. delivery schedules, phase timelines

Resource and co-ordination roles: project manager roles e.g. planning activities, progress oversight; team roles e.g. developers, analysts; responsibility ownership e.g. task owners, deliverable leads; team dynamics e.g. collaboration, communication, role interaction, conflict resolution, motivation and engagement within project teams

Stakeholder involvement: internal stakeholders e.g. project team, managers; external stakeholders e.g. clients, suppliers; sponsor roles e.g. funding approval, strategic oversight.

LO2 **Apply digital project planning techniques to develop a project plan for an identified organisational scenario**

*Requirements definition and scoping techniques:*

Requirements identification: business requirements e.g. operational needs, organisational goals; user needs e.g. end-user expectations, usability requirements; stakeholder needs e.g. sponsor priorities, client expectations

Requirements specification: functional requirements e.g. system features, process functions; non-functional requirements e.g. performance standards, security criteria; requirement prioritisation e.g. mandatory features, optional features

Scope control: scope definition e.g. project boundaries, deliverable limits; scope boundaries e.g. inclusions, exclusions; project assumptions e.g. resource availability, technical constraints; project constraints e.g. budget limits, time restrictions.

*Work breakdown and task structuring techniques:*

Task planning: work breakdown structure e.g. project phases, task hierarchies; task decomposition e.g. large tasks, smaller activities; activity definition e.g. work packages, task descriptions

Task sequencing: task sequencing e.g. ordered activities, logical flows; task dependencies e.g. finish-to-start links, start-to-start links; dependency mapping e.g. dependency diagrams, task relationships

Delivery organisation: deliverable identification e.g. project outputs, completion items; deliverable alignment e.g. task-to-output links, phase objectives; task grouping e.g. related tasks, functional grouping; workflow structure e.g. task flows, process order.

*Estimation and resource planning techniques:*

Estimation techniques: time estimation e.g. activity duration, schedule estimates; effort estimation e.g. workload estimates, staff effort; cost estimation e.g. labour costs, material costs

Resource management: resource availability e.g. staff availability, equipment access; resource allocation e.g. staff assignment, tool assignment; skills allocation e.g. specialist skills, technical competencies

Capacity planning: capacity planning e.g. workload limits, staffing levels; workload planning e.g. task loading, work balancing; scheduling assumptions e.g. working hours, availability periods; budget considerations e.g. cost limits, funding plans.

*Risk management techniques:*

Risk analysis: risk identification e.g. technical risks, schedule risks; planning risks e.g. estimation errors, dependency issues; project uncertainty e.g. requirement changes, external influences

Risk response: contingency planning e.g. backup resources, alternative actions; risk prioritisation e.g. high-impact risks, critical risks; impact assessment e.g. cost impact, schedule impact

Planning control: dependency risks e.g. supplier delays, task slippage; constraint management e.g. time limits, budget limits; planning tolerances e.g. allowed variance, performance thresholds; assumption validation e.g. assumption checks, review updates.

LO3 **Implement a range of digital tools to monitor and manage project progress for an identified organisational scenario**

*Digital scheduling and workflow tools:*

Scheduling tools: Gantt charts e.g. project timelines, milestone schedules; scheduling software e.g. task scheduling, timeline planning

Workflow management: Kanban boards e.g. visual task flow, work-in-progress tracking; digital workflows e.g. automated task flow, approval routing

Task organisation: sprint backlogs e.g. planned sprint tasks, prioritised activities; task management tools e.g. task assignment, workload tracking

Progress display: progress visualisation e.g. status charts, completion indicators; timeline views e.g. delivery timelines, schedule overviews.

*Monitoring and reporting mechanisms:*

Progress tracking: progress indicators e.g. completion percentages, traffic-light status; milestone tracking e.g. milestone completion, review checkpoints

Reporting outputs: status reports e.g. weekly reports, highlight summaries; exception reports e.g. deviation reports, issue summaries

Performance visibility: project dashboards e.g. management overviews, performance displays; performance metrics e.g. delivery metrics, productivity measures.

*Risk, issue, and communication tools:*

Risk management records: risk logs e.g. identified risks, mitigation actions; issue trackers e.g. issue recording, resolution tracking

Communication platforms: collaboration platforms e.g. team messaging, shared workspaces; communication channels e.g. project chat, update emails

Information sharing: document sharing systems e.g. central repositories, shared folders; feedback records e.g. review comments, stakeholder feedback.

*Data accuracy and control:*

Data integrity: data consistency e.g. aligned records, consistent updates; data validation e.g. accuracy checks, entry verification

Access control: access permissions e.g. user rights, editing controls; user roles e.g. administrator roles, contributor roles

Audit and tracking: audit trails e.g. activity history, change logs; version tracking e.g. document versions, update history

*Examples of real-world project management applications:*

A range of project management software examples e.g.

Project scheduling tools: Microsoft Project e.g. Gantt scheduling, milestone tracking; Smartsheet e.g. schedule tracking, dashboard reporting; Microsoft Planner e.g. task planning, progress tracking

Management tools: Jira Software e.g. backlog management, sprint tracking; Trello e.g. Kanban boards, task tracking; Asana e.g. work tracking, progress updates; Monday.com e.g. workflow management, project tracking

Collaboration and reporting platforms: Microsoft Teams e.g. project communication, file sharing; SharePoint e.g. document management, version control; ClickUp e.g. task tracking, reporting tools; Wrike e.g. performance tracking, collaboration.

LO4 **Review the project outcomes against identified success criteria**

*Success criteria and performance measures:*

Performance metrics: key performance indicators e.g. productivity measures, delivery measures, efficiency indicators; success criteria e.g. agreed objectives, acceptance thresholds; performance targets e.g. delivery targets, cost targets

Quality and acceptance: deliverable acceptance e.g. client sign-off, approval confirmation; quality benchmarks e.g. quality standards, specification compliance

Schedule and cost: schedule performance e.g. planned versus actual timelines, schedule variance; cost performance e.g. budget usage, cost variance; tolerance thresholds e.g. allowable variance, reporting limits.

*Outcome and quality review:*

Outcome comparison: planned outcomes e.g. intended deliverables, expected outputs; actual outcomes e.g. completed deliverables, achieved results; variance analysis e.g. performance gaps, delivery differences

Quality and compliance: quality assurance outcomes e.g. quality checks, inspection results; requirement compliance e.g. requirement fulfilment, specification adherence; acceptance results e.g. approval decisions, review outcomes

Review analysis: performance gaps e.g. unmet targets, shortfalls; deviation causes e.g. scheduling issues, resource constraints; review findings e.g. evaluation conclusions, review outcomes.

*Stakeholder perspectives and satisfaction:*

Expectations and perception: stakeholder expectations e.g. success standards, delivery expectations, key performance indicators (KPI) targets; perception alignment e.g. expectation alignment, outcome understanding

Feedback and engagement: stakeholder feedback e.g. review comments, evaluation feedback; feedback analysis e.g. feedback trends, response patterns; engagement quality e.g. participation levels, stakeholder involvement

Satisfaction and communication: satisfaction levels e.g. stakeholder satisfaction, approval ratings; communication effectiveness e.g. clarity of updates, reporting quality.

*Improvement identification:*

Learning capture: lessons learned e.g. project learnings, delivery insights; knowledge capture e.g. documented learning, shared knowledge

Improvement actions: improvement opportunities e.g. process improvements, delivery enhancements; corrective actions e.g. action plans, issue resolution; future recommendations e.g. improvement suggestions, best-practice guidance

Continuous improvement: process improvements e.g. workflow refinement, efficiency gains; performance refinement e.g. delivery optimisation, quality improvement.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explain the core principles of digital project management | | |
| **P1** Explain, using a selection of real-world examples, the core principles of digital project management including life cycles, documentation and stakeholders.<br><br>**P2** Discuss, using a selection of real-world examples, the types of governance structures and processes used in digital project management. | **M1** Analyse, using a selection of real-world examples, how core digital project management principles can impact on project planning, control and delivery in a range of organisational contexts. | **D1** Evaluate, using a selection of real-world examples, the effectiveness of digital project management principles in supporting successful project delivery for identified organisational contexts. |

| Pass | Merit | Distinction |
|---|---|---|
| **LO2** Apply digital project planning techniques to develop a project plan for an identified organisational scenario | | **LO2, LO3 and LO4**<br>**D2** Justify the decisions made in the development and implementation of the project plan in relation to how they impacted on the successful delivery of the project against the initial client requirements. |
| **P3** Apply digital project planning techniques to develop a project plan for an identified organisational scenario.<br>**P4** Review the developed project plan with a range of identified internal stakeholders to identify strengths and areas for improvement. | **M2** Optimise, using stakeholder feedback, the developed project plan so that it fully meets the delivery requirements and deadlines of the identified organisational scenario. | |
| **LO3** Implement a range of digital tools to monitor and manage project progress for an identified organisational scenario | | |
| **P5** Implement a range of digital tools to monitor and manage the project plan for an identified organisational scenario. | **M3** Evaluate the effectiveness of the chosen digital tools in supporting the tracking, management and communication of the digital project. | |
| **LO4** Review the project outcomes against identified success criteria | | |
| **P6** Review project outcomes against identified success criteria.<br>**P7** Present the results of the project outcomes against identified success criteria to a range of identified external stakeholders to identify strengths and areas for improvement. | **M4** Evaluate, using the results of the stakeholder feedback, the project outcomes in relation to performance, quality, and stakeholder expectations. | |

## Recommended Resources

### Textbooks

Brewer, J.L. and Dittman, K.C. (2022) *Methods of IT Project Management*. 4th edn. West Lafayette, IN: Purdue University Press.

Kerzner, H. (2022) *Project Management Metrics, KPIs, and Dashboards*. 4th edn. Hoboken, NJ: Wiley.

Kerzner, H. (2023) *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. 13th edn. Hoboken, NJ: Wiley.

Kerzner, H. (2023) *Project Management Best Practices: Achieving Global Excellence*. 5th edn. Hoboken, NJ: Wiley.

Schwalbe, K. (2024) *Information Technology Project Management*. 10th edn. Boston: Cengage Learning.

Wysocki, R.K. (2019) *Effective Project Management: Traditional, Agile, Extreme, Hybrid*. 8th edn. Indianapolis, IN: Wiley.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.6sigma.us/project-management/project-life-cycle/ | Six Sigma<br>The Ultimate Guide to Project Life Cycle: Mastering the 5 Phases for Project Success<br>(Article) |
| productive.io/blog/digital-project-management/ | Productive.io<br>What Is Digital Project Management? Detailed Guide + Tips<br>(Blog) |
| www.projectmanager.com/blog/what-does-project-governance-really-mean | ProjectManager<br>What Does Project Governance Really Mean?<br>(Article) |
| thedigitalprojectmanager.com/project-management/project-planning-techniques/ | The Digital Project Manager<br>11 Project Planning Techniques For Project Success in 2026<br>(Tutorial) |

| | |
|---|---|
| www.projectmanager.com/blog/how-to-create-a-work-breakdown-structure | ProjectManager<br>How to Create a Work Breakdown Structure (WBS) Step-by-Step<br>(Tutorial) |
| www.proofhub.com/articles/project-tracking | ProofHub<br>What is project tracking and how to track project progress effectively?<br>(Article) |
| www.celoxis.com/article/project-management-bi-dashboards | Celoxis<br>Introduces project dashboards and key performance indicators used to track progress and performance.<br>(Article) |
| www.mural.co/blog/the-definitive-guide-to-project-tracking | Mural<br>The definitive guide to project tracking<br>(Blog) |
| project-management.com/6-ways-of-measuring-project-success/ | Project Management.com<br>How to Measure Project Success: Key Metrics, Tools, & Best Practices<br>(Article) |
| www.float.com/resources/project-evaluation | Float<br>Project evaluation guide: how to measure the success of a project<br>(Article) |

**Journals and articles**

Bosch-Rekveldt, M. *et al*. (2023) 'Performance measurement in project management', *International Journal of Project Management*, 41, 102518. Available at: doi.org/10.1016/j.ijproman.2023.102518.

Chen, M. *et al*. (2025) 'Digital Transformation in Project Management: A Systematic Review and Research Agenda', *Systems*, 13(8), 625. Available at: doi.org/10.3390/systems13080625.

Gonçalves, M.L.A. *et al*. (2023) 'Technological tools for project management in the digital transformation context', *Concilium*, 23. Available at: doi.org/10.53660/CLM-1045-23C57.

Sommer, L. (2024) 'Project management approaches and their selection in the digital age: Overview, challenges and decision models', *Journal of Project Management*, 9(2), pp. 131–148. Available at: doi.org/10.5267/j.jpm.2024.1.001.

**Links**

This unit links to the following related units:

*Unit 402: Designing a Computing Project (Pearson-set)*

*Unit 403: Professional Practice*

*Unit 404: Digital Systems Analysis.*

# Unit 427: UI/UX Programming and Development

**Unit code:** M/652/3279

**Unit level:** 4

**Credit value:** 15

---

## Introduction

The design and development of User Interface/User Experience (UI/UX) play a central role in how people interact with digital systems across a wide range of contexts, including websites, mobile applications, business software and interactive services. Well-designed interfaces support ease of use, accessibility and efficiency, while poor design can create barriers for users and reduce the effectiveness of digital products. As organisations place increasing emphasis on usability and inclusive design, there is growing demand for computing practitioners who understand how UI/UX principles shape user interaction.

The aim of this unit is to develop students' understanding of the principles, methods and practices used in UI/UX design, and to support the application of this knowledge within practical, scenario-based contexts. The unit focuses on how design decisions influence usability, accessibility and user experience, and encourages students to consider their professional responsibility when creating interfaces intended for diverse users.

This unit covers the foundational concepts of UI/UX design, including design principles, heuristics and commonly used UI/UX methodologies. Students will explore accessible interface structure and layout theory, examining navigation models, visual hierarchy and recognised accessibility standards. The unit also enables students to design, implement and refine functional and accessible UI/UX solutions for identified scenarios, using usability testing methods and user feedback to support iterative improvement.

On successful completion of this unit, students will be able to explain and apply core UI/UX principles, develop accessible interface solutions, and evaluate and refine designs using usability evidence. These skills are directly applicable to a range of computing and digital development roles, including Junior UI/UX Designer, Front-End Developer, Interaction Designer, Digital Product Support Analyst, and Web Application Developer, and also provide a strong foundation for progression to further study in computing and software development.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core UI/UX design principles and methodologies

LO2 Discuss the principles of accessible interface structure and layout theory

LO3 Implement functional and accessible UI/UX solutions for an identified scenario

LO4 Refine the UI/UX solutions for an identified scenario using iterative methods.

## Essential Content

### LO1 Explain the core UI/UX design principles and methodologies

*UI and UX design foundations:*

Design disciplines: user interface design e.g. graphical interface, digital interface; user experience design e.g. user journey, experience flow; interaction design e.g. control interaction, gesture interaction

User focus: user-centred design e.g. user needs, user goals; user needs e.g. functional needs, accessibility needs; user goals e.g. task completion, efficiency improvement

Usability context: usability concepts e.g. effectiveness, efficiency; interface usability e.g. ease of use, learnability; human-computer interaction e.g. interaction models, system feedback.

*Core UI/UX design principles:*

Interface behaviour: consistency e.g. visual consistency, interaction consistency; visibility e.g. system status, navigation visibility

Interaction control: affordance e.g. clickable elements, control cues; error prevention e.g. validation controls, confirmation prompts; error recovery e.g. undo actions, error messages

Usability qualities: simplicity e.g. minimal design, content reduction; learnability e.g. intuitive layout, familiar patterns; efficiency e.g. reduced steps, task speed.

*Design heuristics and guidelines:*

Evaluation and rules: heuristic evaluation e.g. usability principles, design rules; usability rules e.g. error tolerance, control visibility

Structural guidance: interface guidelines e.g. platform standards, layout rules; navigation rules e.g. clear pathways, logical grouping

Design assets: design patterns e.g. navigation patterns, interaction patterns; design systems e.g. component libraries, style guides; responsive design e.g. flexible layouts, adaptive content.

*UI/UX methodologies:*

Design processes: user-centred design life cycle e.g. research stages, evaluation stages; iterative design e.g. repeated testing, refinement cycles

Development practices: agile integration e.g. sprint design, collaboration workflow; requirement analysis e.g. functional analysis, user analysis

Design artefacts and review: wireframing e.g. low fidelity layouts, screen structure; prototyping e.g. interactive mock-ups, clickable models; solution validation e.g. requirement checking, user approval.

LO2 **Discuss the principles of accessible interface structure and layout theory**

*Accessible interface structure:*

Navigation structure: navigation models e.g. linear navigation, hierarchical navigation; menu structure e.g. drop-down menus, side menus; user pathways e.g. task paths, navigation routes

Content organisation: information architecture e.g. content mapping, site structure; content grouping e.g. logical grouping, visual grouping; page hierarchy e.g. primary pages, secondary pages

Task flow: logical flow e.g. step sequences, user journeys; task structure e.g. task steps, action order.

*Layout and visual hierarchy:*

Layout foundations: alignment e.g. edge alignment, centre alignment; spacing e.g. padding, margins; proximity e.g. related elements, grouped content

Visual emphasis: contrast e.g. colour contrast, size contrast; visual weighting e.g. focal points, emphasis placement

Responsive layout: grid systems e.g. column grids, modular grids; responsive structure e.g. fluid layouts, adaptive layouts

Typography structure: typography hierarchy e.g. heading levels, text emphasis.

*Accessibility principles and standards:*

Design principles: inclusive design e.g. diverse users, equal access; accessibility by design e.g. early consideration, embedded accessibility

Standards and guidance: Web Content Accessibility Guidelines (WCAG) e.g. success criteria, conformance levels; Perceivable, Operable, Understandable and Robust (POUR) principles e.g. perceivable, operable

Accessible content qualities: perceivable content e.g. text alternatives, visual clarity; operable controls e.g. keyboard control, focus order; understandable content e.g. clear language, predictable behaviour

Quality assurance: accessibility testing e.g. manual testing, automated testing; accessibility validation e.g. standards checking, issue verification.

*Usability evaluation and metrics:*

Evaluation methods: usability testing e.g. task testing, user observation; accessibility evaluation e.g. audit testing, guideline review

Performance measures: effectiveness measures e.g. task success, goal achievement; efficiency measures e.g. time on task, step count

Outcome metrics: task success e.g. completion rate, accuracy; error rates e.g. user errors, system errors

Assessment frameworks: UX metrics e.g. usability scores, experience metrics; usability benchmarks e.g. baseline measures, comparison data.

LO3 **Implement functional and accessible UI/UX solutions for an identified scenario**

*UI/UX solution design:*

Design scope: scenario design e.g. usage scenario, problem scenario; solution scope e.g. feature boundaries, system limits; design objectives e.g. usability goals, accessibility goals

User definition: user requirements e.g. functional needs, accessibility needs; functional requirements e.g. system behaviour, interface behaviour

Context analysis: user context e.g. device usage, environment use; task definition e.g. user actions, workflow steps

Quality control: success criteria e.g. task completion, usability targets; design documentation e.g. design notes, requirement records.

*UI programming and implementation:*

Interface construction: interface components e.g. buttons, input fields; responsive components e.g. flexible layouts, adaptive elements

Layout and style: layout frameworks e.g. grid frameworks, layout libraries; styling systems e.g. style sheets, theme settings

Interaction handling: interaction behaviours e.g. click actions, hover actions; event handling e.g. user events, system events

System control: state management e.g. interface states, data states; accessibility attributes e.g. ARIA labels, role attributes.

*Accessibility implementation:*

Structural accessibility: semantic structure e.g. semantic elements, document structure; text alternatives e.g. alt-text, descriptive labels

Interaction access: keyboard navigation e.g. tab order, shortcut keys; focus management e.g. focus indicators, focus order

Standards support: assistive compatibility e.g. screen readers, assistive tools; colour contrast e.g. contrast ratios, readable colours

Verification: accessibility validation e.g. guideline checking, issue verification.

*Component integration and standards:*

Component usage: design system usage e.g. shared components, standard elements; component reuse e.g. reusable modules, shared widgets

Platform compliance: platform standards e.g. operating system rules, browser standards; cross-platform considerations e.g. device differences, screen sizes

Technical control: version control e.g. source tracking, change history; dependency management e.g. library control, version tracking.

LO4 **Refine the UI/UX solutions for an identified scenario using iterative methods**

*Usability testing methods:*

Testing types: formative testing e.g. early feedback, design checks; summative testing e.g. final evaluation, outcome measurement

Evaluation approaches: heuristic evaluation e.g. usability principles, design rules; inspection methods e.g. interface review, expert walkthrough

Test execution: user testing sessions e.g. moderated testing, unmoderated testing; scenario testing e.g. task scenarios, usage scenarios.

*User feedback and evaluation:*

Feedback sources: user feedback collection e.g. feedback forms, interviews; observation findings e.g. behaviour notes, usage patterns

Feedback types: qualitative feedback e.g. opinions, observations; quantitative feedback e.g. scores, counts

Issue identification: usability issues e.g. navigation problems, control confusion; accessibility issues e.g. keyboard barriers, contrast problems.

*Analysis and optimisation:*

Analytical focus: usability analysis e.g. task analysis, issue analysis; accessibility analysis e.g. compliance review, barrier analysis

Decision support: performance insights e.g. efficiency trends, error patterns; evidence evaluation e.g. test evidence, feedback evidence

Improvement planning: issue prioritisation e.g. severity ranking, impact ranking; improvement recommendations e.g. design changes, interaction changes.

*Iterative refinement:*

Design adjustments: interface adjustments e.g. layout changes, control changes; interaction improvements e.g. behaviour changes, response changes

Implementation cycle: usability improvements e.g. efficiency improvements, clarity improvements; design iteration e.g. repeated revisions, incremental changes

Validation activities: solution validation e.g. requirement checking, user confirmation; regression checking e.g. repeat testing, issue rechecking.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the core UI/UX design principles and methodologies</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate the effectiveness of UI/UX design principles and methodologies in supporting the development of usable and accessible interface structures for a range of user contexts.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the core UI/UX design foundations and principles.<br><br>P2 Review, using a selection of real-world examples, the concepts of UI/UX design heuristics and methodologies.</td><td>M1 Analyse, using a selection of real-world examples, the application of UI/UX design principles and methodologies within user interface design contexts.</td></tr>
<tr><td colspan="2">LO2 Discuss the principles of accessible interface structure and layout theory</td></tr>
<tr><td>P3 Discuss, using a selection of real-world examples, the principles of accessible interface structure and layout theory.<br><br>P4 Explore, using a selection of real-world examples, the concepts of accessibility standards and usability evaluation.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of usable and accessible interface structure and layouts for a range of user contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Implement functional and accessible UI/UX solutions for an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of the UI/UX solutions in how effectively they meet the functional and accessibility requirements of the identified scenario. |
| **P5** Design functional and accessible UI/UX solutions for an identified scenario.<br>**P6** Implement functional and accessible UI/UX solutions for an identified scenario. | **M3** Assess, using stakeholder feedback, the functionality and accessibility of implemented UI/UX solutions in meeting the requirements of an identified scenario. | |
| **LO4** Refine the UI/UX solutions for an identified scenario using iterative methods | | |
| **P7** Implement usability testing methods on the UI/UX solutions for an identified scenario to identify strengths and areas for improvement.<br>**P8** Review the UI/UX solutions developed for an identified scenario with a range of identified stakeholders to identify strengths and areas for improvement.<br>**P9** Refine, using the results of usability testing and stakeholder feedback, the UI/UX solutions for an identified scenario using an iterative methodology. | **M4** Evaluate the impact of usability testing and user feedback on the refinement and improvement of UI/UX solutions. | |

## Recommended Resources

### Textbooks

Lazar, J., Goldstein, D.F. and Taylor, A. (2024) *Ensuring Digital Accessibility through Process and Policy*. 3rd edn. Burlington, MA: Morgan Kaufmann.

Marcus, A., Rosenzweig, E., and Soares, M. M. (Eds.). (2024). *Design, User Experience, and Usability* (Lecture Notes in Computer Science, Vol. 14714). Springer.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N. and Diakopoulos, N. (2021) *Designing the User Interface: Strategies for Effective Human–Computer Interaction*. 6th edn. Pearson.

Stephanidis, C., & Salvendy, G. (Eds.). (2024). *Designing for Usability, Inclusion and Sustainability in Human–Computer Interaction*. CRC Press.

Unger, R. and Chandler, C. (2024) *A Project Guide to UX Design: For User Experience Designers in the Field or in the Making*. 3rd edn. Berkeley, CA: New Riders.

Yablonski, J. (2024) *Laws of UX: Using Psychology to Design Better Products & Services.* 2nd edn. O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.nngroup.com/articles/ten-usability-heuristics/ | Nielsen Norman Group<br>10 Usability Heuristics<br>(Article) |
| interaction-design.org/literature/topics/user-centered-design | Interaction Design Foundation<br>User Centred Design<br>(General reference) |
| developer.mozilla.org/en-US/docs/Learn/Accessibility | MDN Web Docs<br>Accessibility learning area<br>(Tutorial) |
| www.w3.org/WAI/fundamentals/accessibility-intro/ | W3C Web Accessibility Initiative<br>Web accessibility introduction<br>(General reference) |
| www.smashingmagazine.com/2021/03/complete-guide-accessible-front-end-components/ | Smashing Magazine<br>Accessible front end Components<br>(Tutorial) |

| | |
|---|---|
| www.nngroup.com/articles/usability-testing-101/ | Nielsen Norman Group<br>Usability Testing 101<br>(Article) |
| digital.gov/event/2018/06/14/usability-testing-with-steve-krug/ | Digital.gov<br>Usability testing methods<br>(Webinar) |
| www.uxdesigninstitute.com/blog/prototyping-guide/ | UX Design Institute<br>A complete guide to prototyping<br>(Article) |
| uxplaybook.org/articles/understanding-interaction-design-in-ux | UXPlaybook<br>Understanding Interaction Design in UX: Crafting Seamless User Journeys<br>(Article) |

**Journals and articles**

Ara, J., Sik-Lányi, C., and Kelemen, Á. (2023) 'Accessibility engineering in the web evaluation process: a systematic literature review', *Universal Access in the Information Society*. Available at: doi.org/10.1007/s10209-023-00967-2.

Martins, B., and Duarte, C. (2023) 'A large-scale web accessibility analysis considering technology adoption', *Universal Access in the Information Society*. Available at: doi.org/10.1007/s10209-023-01010-0.

Ramadhanti, N. T., Budiyanto, C. W., and Yuana, R. A. (2023) 'The use of heuristic evaluation on UI/UX design', *AIP Conference Proceedings*, 2540(1), 080008. Available at: doi.org/10.1063/5.0105701.

Weichbroth, P. (2024) 'Usability Testing of Mobile Applications: A Methodological Framework', *Applied Sciences*, 14(5), 1792. Available at: doi.org/10.3390/app14051792.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science*

*Unit 405: Core Software Design and Development*

*Unit 406: Core Software Testing*

*Unit 408: Applied Software Development*

*Unit 429: Responsive Website Development*

*Unit 433: Game Development.*

 © Pearson Education Limited 2026

# Unit 428: Maths for Computing

**Unit code:** Y/652/3280

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Mathematics underpins all areas of computing, from software development and data analysis to cybersecurity, networking and computer graphics. Professionals working in roles such as Software Developer, Data Analyst, Network Engineer and Systems Technician are required to apply mathematical reasoning to solve problems, interpret data and support effective system design. As digital systems increase in complexity, the ability to use mathematics accurately and critically becomes essential within the computing sector.

The purpose of this unit is to develop students' understanding of core mathematical principles and their application within computing contexts. The unit aims to equip students with the ability to apply mathematical methods, analyse their effectiveness and evaluate their suitability when solving computing-related problems. This supports both academic progression and the development of skills required in technical computing roles.

The unit covers a structured range of mathematical topics aligned to computing needs. Students will apply arithmetic and algebraic methods to solve problems and explore logic. They will also use Boolean algebra to support computing processes, as well as functions, graphs and numerical methods to interpret data. The unit also introduces geometry, vectors and matrices, highlighting their importance in areas such as computer graphics, simulations and systems modelling. These topics are designed to develop both mathematical understanding and the ability to apply concepts in practical computing scenarios.

On successful completion of this unit, students will be able to apply mathematical methods accurately, analyse different approaches and evaluate their effectiveness in a range of computing contexts. They will develop transferable skills in problem solving, logical reasoning, data interpretation and mathematical communication. These skills support progression to higher-level study and to roles within the computing sector, such as Software Developer, Data Analyst, Network Engineer or Cybersecurity Specialist, where confident and critical use of mathematics is essential.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Apply a range of fundamental arithmetic and algebraic methods to computing problems

LO2 Explore the principles of logic and Boolean algebra to support computing processes

LO3 Implement functions, graphs and numerical methods to interpret computing-related data

LO4 Review the principles of geometry, vectors and matrices in computing contexts.

## Essential Content

LO1 **Apply a range of fundamental arithmetic and algebraic methods to computing problems**

*Arithmetic fundamentals:*

integers, real numbers, fractions, percentages, ratios, indices, standard form, order of operations, numerical accuracy, rounding, precision, significant figures, calculation efficiency.

*Algebraic expressions and equations:*

Variables, constants, algebraic expressions, linear equations, simultaneous equations, inequalities, rearrangement of formulas, substitution, solution accuracy.

*Algebra in computing contexts:*

Algorithmic calculations, parameter values, configuration settings, resource estimation, performance metrics, suitability of methods e.g. memory allocation, execution time calculations, comparison of alternative methods, selection of appropriate approaches for different computing problems.

*Evaluation of methods and results:*

Checking accuracy and reasonableness of outputs, considering efficiency of calculations, impact of precision and rounding on results, identifying limitations of methods and suitability for different contexts.

LO2 **Explore the principles of logic and Boolean algebra to support computing processes**

*Principles of logic:*

Propositions, logical statements, logical reasoning, truth values, implication, equivalence, logical consistency.

*Boolean algebra concepts:*

Boolean variables, Boolean expressions, logical operators AND, OR, NOT, operator precedence, simplification principles.

*Truth tables and logical representation:*

Truth tables, logical evaluation, condition testing, decision outcomes, correctness of results.

*Logic in computing processes:*

Control logic, conditional processing, decision structures, system states, efficiency of logic design e.g. access control conditions, program branching, comparison of logical structures, simplification of Boolean expressions.

*Evaluation of logic:*

Assessing correctness and reliability of logical outcomes, identifying inefficient or redundant logic, considering clarity, maintainability and efficiency of logical designs.

LO3 **Implement functions, graphs and numerical methods to interpret computing-related data**

*Mathematical functions:*

Linear functions, quadratic functions, polynomial functions, exponential functions, function notation, input–output relationships, functional behaviour.

*Graphs and graphical representation:*

Cartesian co-ordinates, axes, scales, plotting points, line graphs, curves, trends, intersections, clarity of representation.

*Numerical methods and modelling:*

Approximation techniques, estimation, iterative methods, basic interpolation, numerical accuracy, computational efficiency.

*Data interpretation in computing contexts:*

Trend identification, pattern recognition, model representation, suitability of methods e.g. performance data trends, system output visualisation, comparison of graphical, numerical and functional approaches.

*Evaluation of methods:*

Assessing accuracy and usefulness of models and outputs, selecting appropriate methods for different computing scenarios, identifying strengths and limitations of approaches, considering efficiency and suitability for decision-making.

LO4 **Review the principles of geometry, vectors and matrices in computing contexts**

*Geometric principles:*

Two-dimensional shapes, three-dimensional shapes, angles, transformations, co-ordinate geometry, spatial representation.

*Vectors:*

Vector quantities, magnitude, direction, vector representation, vector addition, scalar multiplication, directional accuracy.

*Matrices:*

Matrix notation, matrix operations, matrix multiplication, transformations, systems representation, structural relationships.

*Applications in computing contexts:*

Graphics positioning, transformations, simulations, data structures, effectiveness of mathematical representation e.g. screen coordinates, object movement, comparison of different mathematical representations.

*Selection and evaluation of methods:*

Choosing appropriate geometric, vector or matrix methods for specific computing applications, assessing effectiveness, accuracy and efficiency, identifying limitations and suitability of different approaches.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Apply a range of fundamental arithmetic and algebraic methods to computing problems | | |
| **P1** Apply arithmetic calculations relevant to computing scenarios.<br>**P2** Solve algebraic expressions and equations in computing contexts. | **M1** Analyse the effectiveness of arithmetic and algebraic methods when applied to computing problems. | **D1** Evaluate the suitability of arithmetic and algebraic methods for different computing problem contexts. |
| **LO2** Explore the principles of logic and Boolean algebra to support computing processes | | |
| **P3** Explore logical expressions using Boolean operators.<br>**P4** Produce truth tables to represent logical outcomes in computing processes. | **M2** Examine how logic and Boolean algebra support decision-making in computing processes. | **D2** Evaluate the role of logic and Boolean algebra in supporting efficient and reliable computing processes. |
| **LO3** Implement functions, graphs and numerical methods to interpret computing-related data | | |
| **P5** Implement numerical methods to interpret computing-related data.<br>**P6** Construct functions and graphs to represent computing-related data. | **M3** Interpret the use of functions, graphs and numerical methods when analysing computing-related data. | **D3** Evaluate the effectiveness of functions, graphs and numerical methods for interpreting computing-related data. |
| **LO4** Review the principles of geometry, vectors and matrices in computing contexts | | |
| **P7** Review the use of geometric principles in computing contexts.<br>**P8** Apply vectors and matrices in computing contexts. | **M4** Analyse how geometry, vectors and matrices are used to solve computing problems. | **D4** Evaluate the selection of geometric, vector and matrix methods for specific computing applications. |

## Recommended Resources

### Textbooks

Goodaire, E. and Parmenter, M. (2023) *Discrete Mathematics with Graph Theory*. 4th edn. London: Pearson Education.

Lay, D.C., Lay, S.R. and McDonald, J.J. (2024) *Linear Algebra and Its Applications*. 6th edn. Harlow: Pearson.

Rosen, K.H. (2023) *Discrete Mathematics and Its Applications*. 9th edn. New York: McGraw-Hill.

Stewart, J. and Day, T. (2024) *Calculus: Concepts and Contexts*. 5th edn. Boston: Cengage Learning.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| ajmc.aut.ac.ir/ | AUT Journal of Mathematics and Computing<br>Peer-reviewed articles on applied mathematics and computing<br>(Research) |
| aace.org/pubs/jcmst/ | Journal of Computers in Mathematics and Science Teaching<br>Use of computing in mathematics education<br>(Research) |
| jocg.org/ | Journal of Computational Geometry<br>geometry and spatial computation research<br>(Research) |
| link.springer.com/journal/12190 | Journal of Applied Mathematics and Computing<br>Numerical methods and computational modelling<br>(Research) |

| | |
|---|---|
| www.mathsciteacher.com/ | Journal of Mathematics and Science Teacher<br>Mathematics education and computational thinking<br>(Research) |
| serdica-comp.math.bas.bg/ | Serdica Journal of Computing<br>Computing theory and applications<br>(Research) |
| www.computer.org/csdl/journal/tg | IEEE Transactions on Visualization and Computer Graphics<br>Graphical data, matrices and geometry in computing<br>(Research) |
| www.journal-cand.com/ | Computational Algorithms and Numerical Dimensions<br>Numerical methods and algorithms in computing<br>(Research) |

### Journals and articles

Al-Wardi, Z. (2023) 'Optimized Boolean expression embedding in quantum and reversible logic circuits', *AIP Conference Proceedings*, 2591(1), 020035. Available at: doi.org/10.1063/5.0119324.

Navarro, E.R. and de Sousa, M.C. (2023) 'The concept of computational thinking in mathematics education', *Journal of Mathematics and Science Teacher*, 3(2), em046. Available at: doi.org/10.29333/mathsciteacher/13630.

Puspitasari, N. and Mayrohmah, S.H. (2024) 'The role of Boolean algebra in computer science and logic circuit design', *Jurnal Informa*, 9(2). Available at: doi.org/10.46808/informa.v9i2.258.

Zhang, Y. *et al*. (2024) 'Numerical methods for computing the discrete and continuous Laplace transforms', *IMA Journal of Applied Mathematics*, 89(5), pp. 891–950. Available at: doi.org/10.1093/imamat/hxae033.

### Links

This unit links to the following related units:

*Unit 405: Core Software Design and Development*

*Unit 411: Core Data Analytics*

*Unit 413: Core Networking*

*Unit 420: Core AI*

*Unit 425: AI Programming.*

# Unit 429: Responsive Website Development

**Unit code:** A/652/3281

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Websites play a central role in how organisations communicate, deliver services and engage with users across a wide range of sectors. With access increasingly taking place on mobile phones, tablets and desktop devices, modern websites are expected to adapt smoothly to different screen sizes and user contexts. This unit explores the principles and technologies used to design and develop responsive websites that function effectively across multiple devices. Responsive Web Design (RWD) sits within the wider field of web and digital technologies and supports students developing practical skills for contemporary web development roles.

The aim of this unit is to provide students with a clear understanding of how responsive websites are designed and developed using core web technologies. Students will examine how structure, presentation and behaviour work together to support effective, accessible and adaptable web solutions. The unit focuses on the expectations placed on entry-level practitioners, introducing approaches that reflect current industry practice and employer requirements.

The unit covers essential web technologies including HTML, CSS and JavaScript, alongside the principles that underpin responsive web design. Students will explore how layouts adapt through flexible sizing, breakpoints and conditional styling, and how responsive behaviour supports usability across devices. The unit also develops practical skills through the design and development of a responsive website for an identified scenario, supported by testing, performance checks and accessibility considerations.

On successful completion of this unit, students will be able to design, develop and evaluate responsive websites that meet defined user and organisational requirements. They will gain transferable skills in problem solving, technical implementation and digital design, supporting progression into roles such as Junior Web Developer, Front-End Developer, Web Content Developer, Digital Support Technician, or UI Developer, or further study in web and software development.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Describe the essential web technologies and how they underpin website design and development

LO2 Explain the principles of responsive web design (RWD)

LO3 Explore the mechanisms that allow for responsive web design

LO4 Develop a responsive and accessible website design for an identified scenario.

## Essential Content

LO1 **Describe the essential web technologies and how they underpin website design and development**

*Core web technologies:*

HTML structure: semantic elements e.g. header, main, footer; document hierarchy e.g. headings, subheadings, content levels; page layout foundations e.g. sections, containers, page regions; content structure e.g. paragraphs, lists, media placement

Document standards and validation: web standards e.g. W3C standards, HTML specifications; markup validation e.g. validator tools, error reports; accessibility semantics e.g. landmarks, structural roles

Information organisation: content hierarchy e.g. prioritisation, structure, ordering; navigation structure e.g. menus, navigation bars, breadcrumbs; internal linking e.g. hyperlinks, navigation paths.

*Styling and presentation technologies:*

CSS syntax and rules: selectors e.g. class selectors, ID selectors, type selectors; cascading rules e.g. precedence, override order, specificity; inheritance e.g. font inheritance, colour inheritance, text inheritance; stylesheet structure e.g. external stylesheets, rule grouping, file organisation

Box model and layout: margin e.g. element spacing, layout gaps, separation; padding e.g. internal spacing, content buffers, whitespace; display properties e.g. block, inline-block, flex

Visual presentation: typography e.g. font families, font sizes, line height; colour schemes e.g. palettes, themes, brand colours; contrast e.g. contrast ratios, readability, visibility; visual hierarchy e.g. emphasis, weighting, prioritisation.

*Client-side scripting technologies:*

JavaScript fundamentals: variables e.g. values, state; data types e.g. strings, numbers; control structures e.g. conditions, loops; functions e.g. reusable routines, handlers

Event handling and interactivity: event listeners e.g. click events, submit events; user interaction e.g. buttons, forms; input validation e.g. required fields, format checks; interface behaviour e.g. dynamic responses

Dynamic content behaviour: Document Object Model (DOM) manipulation e.g. content updates, element changes; element visibility e.g. show and hide states; class manipulation e.g. class toggling, state classes.

*Document Object Model (DOM):*

DOM structure and relationships: parent nodes e.g. containers, wrappers; child nodes e.g. nested elements; element hierarchy e.g. DOM tree, nesting

Element selection and manipulation: element selection methods e.g. selectors, DOM queries; attribute manipulation e.g. src attributes, href values

Interaction between HTML, CSS, and JavaScript: structure–presentation–behaviour co-ordination e.g. semantic markup, CSS styling, script control.

LO2 **Explain the principles of responsive web design (RWD)**

*Responsive design principles:*

Responsive principles: device independence e.g. multi-device access, screen flexibility; adaptive layouts e.g. layout adjustment, content reflow; content flexibility e.g. scalable elements, fluid content; design consistency e.g. consistent layout behaviour, uniform user experience; proportion-based design e.g. relative sizing, proportional layout; resolution independence e.g. variable screen sizes, diverse display resolutions

User experience consistency: consistent behaviour e.g. predictable navigation, stable interaction; usability continuity e.g. readable content, accessible controls; interaction coherence e.g. consistent menus, consistent feedback; presentation continuity e.g. visual consistency, layout familiarity; functional continuity e.g. feature availability, interaction reliability across devices.

*Layout adaptability concepts:*

Fluid layouts: fluid grids e.g. proportional widths, relative columns; flexible containers e.g. expandable containers, resizable regions; flexible units e.g. percentages, viewport-based units; scalable content e.g. responsive images, adaptable media

Breakpoints and viewport behaviour: breakpoints e.g. layout change points, screen thresholds; viewport behaviour e.g. screen width, device orientation; resolution variation e.g. screen density, display size; device variation e.g. mobile screens, tablet screens.

*Progressive enhancement approach:*

Progressive enhancement: core content priority e.g. essential content, baseline functionality; layered features e.g. styling layers, interaction layers; functionality resilience e.g. basic usability, graceful degradation; content availability e.g. access without scripts, reduced environments

Accessibility-first design: accessibility principles e.g. perceivable content, operable controls; usability considerations e.g. readable layouts, clear navigation; inclusive access e.g. diverse user needs, varied input methods; cross-device compatibility e.g. input support, platform support.

LO3 **Explore the mechanisms that allow for responsive web design**

*Conditional styling mechanisms:*

Conditional styling: CSS conditions e.g. conditional rules, responsive conditions; media query logic e.g. media conditions, rule switching; display conditions e.g. screen size conditions, device conditions; orientation conditions e.g. portrait mode, landscape mode

Viewport and resolution awareness: viewport width e.g. screen width, device width; resolution characteristics e.g. screen density, pixel ratios; display variation e.g. small screens, large screens; device context e.g. phones, tablets, desktops.

*Flexible layout mechanisms:*

Flexible containers: flexible boxes e.g. adaptable containers, layout rows; flexible regions e.g. expandable regions, resizable sections; container behaviour e.g. content expansion, content contraction; layout flow e.g. horizontal flow, vertical flow

Proportional sizing mechanisms: relative units e.g. percentage units, viewport units; scalable dimensions e.g. flexible width, flexible height; size relationships e.g. proportional spacing, relative alignment; content scaling e.g. scalable blocks, adaptable sections.

*Responsive behaviour control:*

Responsive media: fluid images e.g. scalable images, responsive graphics; media containment e.g. content containment, aspect handling; media resizing e.g. automatic resizing, proportion preservation

Text and readability mechanisms: scalable text e.g. relative font sizing, adaptable text; line length control e.g. readable line width, text flow; spacing control e.g. adjustable spacing, proportional margins.

*Interaction and behaviour mechanisms:*

Interface adaptation: layout changes e.g. repositioned elements, reordered sections; navigation adjustment e.g. condensed menus, simplified navigation; interaction adaptation e.g. touch interaction, pointer interaction

Feature availability: conditional features e.g. device-dependent features, capability awareness; behaviour variation e.g. interaction differences, input variation; environment awareness e.g. device context, user context.

LO4 **Develop a responsive and accessible website design for an identified scenario**

*Website design considerations:*

User requirements and scenario analysis: user requirements e.g. user needs, user goals, intended audience; scenario constraints e.g. website purpose, usage context; functional expectations e.g. core features, content requirements

Visual design and layout planning: visual hierarchy e.g. content emphasis, priority ordering; layout structure e.g. page regions, content areas, component placement; design consistency e.g. visual alignment, layout coherence, style consistency

Content and navigation design: content organisation e.g. content structure, information grouping, hierarchy clarity; navigation design e.g. menu structure, navigation clarity, user pathways; readability considerations e.g. content spacing, text clarity, scanning support.

*Website development implementation:*

Responsive layout implementation: layout realisation e.g. implemented page layouts, responsive page structures; content flow handling e.g. stacked content, reordered content; layout consistency e.g. repeated layout patterns, shared structures

Responsive content handling: content prioritisation e.g. key content visibility, secondary content placement; media placement e.g. responsive images, adaptable media; text layout handling e.g. readable line length, adaptable text layout.

*Testing and performance validation:*

Cross-device testing: device coverage e.g. mobile devices, tablets, desktops; display testing e.g. different screen sizes, orientations; usability testing e.g. navigation clarity, interaction behaviour

Device implementation checks: device implementation checks e.g. mobile layout implementation, desktop layout implementation; consistency checks e.g. layout consistency, content consistency across devices

Performance considerations: load performance e.g. page loading time, asset size; rendering behaviour e.g. layout stability, visual consistency

Optimisation and refinement: performance optimisation e.g. compressed images, minified CSS, minified JavaScript; loading and caching strategies e.g. lazy loading, deferred scripts, browser caching; responsive efficiency e.g. simplified layouts, reduced layout shifts, efficient rendering; code optimisation e.g. modular code, reusable components, unused code removal; accessibility optimisation e.g. improved contrast, semantic structure, keyboard navigation; usability optimisation e.g. intuitive navigation, consistent layouts, clearer content; test-driven refinement e.g. performance metrics, usability findings, iterative improvements.

*Accessibility and usability considerations:*

Accessibility standards application: Web Content Accessibility Guidelines (WCAG) e.g. WCAG 2.1 principles, conformance levels A and AA; Perceivable, Operable, Understandable and Robust (POUR) principles e.g. perceivable content, operable controls, understandable interfaces

Accessible responsive implementation: responsive accessibility e.g. accessible layouts across devices, adaptable content presentation; input accessibility e.g. keyboard access, touch input; visual accessibility e.g. contrast ratios, readable text

Usability considerations: usability principles e.g. clarity, consistency; navigation usability e.g. clear menus.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Describe the essential web technologies and how they underpin website design and development</td><td rowspan="3">D1 Evaluate, using a selection of real-world examples, the effectiveness of essential web technologies in modern website design and development.</td></tr>
<tr><td>P1 Describe, using a selection of real-world examples, how core web technologies and the Document Object Model underpin modern website design and development.<br><br>P2 Explore, using a selection of real-world examples, how styling and scripting technologies contribute to the design and development of modern websites.</td><td>M1 Analyse, using a selection of real-world examples, how essential web technologies interact to support modern website design and development.</td></tr>
<tr><td colspan="2">LO2 Explain the principles of responsive web design (RWD)</td></tr>
<tr><td>P3 Explain, using a selection of real-world examples, the core principles of responsive web development.<br><br>P4 Explore, using a selection of real-world examples, how a progressive first approach contributes to the design and development of responsive websites.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of applying the principles of responsive web design within modern web development contexts.</td><td rowspan="4">LO2 and LO3<br><br>D2 Evaluate the effectiveness of RWD principles and mechanisms in supporting flexible and accessible web design.</td></tr>
<tr><td colspan="2">LO3 Explore the mechanisms that allow for responsive web design</td></tr>
<tr><td>P5 Explore, using a selection of real-world examples, the RWD mechanisms that allow for responsive layouts.<br><br>P6 Explore, using a selection of real-world examples, the RWD mechanisms that allow for responsive behaviours.</td><td>M3 Assess, using a selection of real-world examples, the benefits and limitations of applying the RWD mechanisms that allow responsive layouts and behaviours within modern web development contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Develop an accessible responsive website design for an identified scenario | | |
| **P7** Design a responsive website that meets the requirements of an identified scenario.<br><br>**P8** Develop a responsive website that meets the requirements of an identified scenario.<br><br>**P9** Implement a test plan to see how well the responsive website meets the requirements of an identified scenario. | **M4** Optimise, using the results of the test plan, the responsive website so that it comprehensively meets the requirements of an identified scenario. | **D3** Justify the decisions made in the design and development of the responsive website in how effectively it meets user needs and accessibility expectations for the identified scenario. |

## Recommended Resources

### Textbooks

Frain, B. (2023) *Responsive Web Design with HTML5 and CSS3*. 3rd edn. Birmingham: Packt Publishing.

Keith, J. (2024) *HTML5 for Web Designers*. 3rd edn. New York: A Book Apart.

Marcotte, E. (2024) *Responsive Design: Patterns and Principles*. New York: A Book Apart.

Peterson, C. (2021) *Learning Responsive Web Design*. Sebastopol, CA: O'Reilly Media.

Robbins, J.N. (2024) *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript and Web Graphics*. 6th edn. Sebastopol, CA: O'Reilly Media.

Ruvalcaba, Z., Boehm, A. and Delamater, M. (2024) *Murach's HTML and CSS*. 6th edn. Fresno: Murach Books.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML | MDN Web Docs<br>Introduction to structuring content with HTML<br>(Tutorial) |
| web.dev/learn/html/semantic-html/ | Google web.dev<br>Semantic HTML<br>(Tutorial) |
| developer.mozilla.org/en-US/docs/Web/CSS/Guides/Box_model | MDN Web Docs<br>CSS Box Model<br>(General reference) |
| css-tricks.com/a-complete-guide-to-css-media-queries/ | CSS Tricks<br>Complete Guide to CSS Media Queries<br>(Tutorial) |
| developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model | MDN Web Docs<br>Document Object Model (DOM)<br>(General reference) |
| www.javascripttutorial.net/javascript-dom/ | JavaScript Tutorial<br>JavaScript DOM Guide<br>(Tutorial) |

| | |
|---|---|
| web.dev/learn/design/ | web.dev<br>Learn Responsive Design<br>(Tutorial) |
| developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Responsive_Design | MDN Web Docs<br>Responsive Web design<br>(Reference) |
| www.w3.org/WAI/standards-guidelines/wcag/ | W3C Web Accessibility Initiative<br>WCAG 2 Accessibility Guidelines Overview<br>(General reference) |
| www.smashingmagazine.com/2016/02/efficient-responsive-design-process/ | Smashing Magazine<br>Efficient Responsive Design Process<br>(Article) |

**Journals and articles**

Bogdanova, G. *et al*. (2024) 'Design and Development Model of a Web Accessibility Ecosystem', *Information*, 15(10), 613. Available at: doi.org/10.3390/info15100613.

Li, W. *et al*. (2022) 'Design Factors to Improve the Consistency and Sustainable User Experience of Responsive Interface Design', *Sustainability,* 14(15), 9131. Available at: doi.org/10.3390/su14159131.

Parlakkiliç, A. (2022) 'Evaluating the effects of responsive design on the usability of academic websites in the pandemic', *Education and Information Technologies*, 27, pp. 1307–1322. Available at: doi.org/10.1007/s10639-021-10650-9.

Shanthi, A.S., Kokila, G. and Rafeeque, K.M. (2022) 'The Representation and Performance of Responsive Web Page Based on HTML5 and CSS3', *International Journal for Research in Applied Science and Engineering Technology*, 10(11). Available at: doi.org/10.22214/IJRASET.2022.47698.

**Links**

This unit links to the following related units:

*Unit 405: Core Software Design and Development*

*Unit 408: Applied Software Development*

*Unit 427: UI/UX Programming and Development*

*Unit 433: Games Development.*

# Unit 430: Emerging Technologies

**Unit code:** D/652/3282

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Emerging technologies are transforming how organisations operate, compete and deliver value in increasingly complex and digitally driven environments. Developments in areas such as artificial intelligence, data analytics, automation, cloud computing and digital platforms are reshaping business models, workforce practices and customer experiences. This unit is relevant to roles such as Digital Co-ordinators, Business Analysts and Project Support Professionals working in technology-enabled environments.

The purpose of this unit is to develop students' understanding of emerging technologies and how they are applied within organisational contexts. It focuses on building knowledge of key technological concepts and principles, and developing awareness of how these technologies influence organisational performance, operations and decision-making.

The unit covers core concepts of emerging technologies, foundational principles, and the potential organisational impacts of technology adoption. Students will explore how technologies are selected and implemented to support organisational processes, and examine the ethical, social and legal considerations associated with their use. These topics are designed to support the Learning Outcomes and to develop both theoretical understanding and applied knowledge.

On completion of this unit, students will be able to define key technological concepts, explain how technologies function, assess their impact on organisations and discuss associated ethical and legal considerations. These skills support progression to more specialist units in areas such as digital transformation, data analytics, cybersecurity, artificial intelligence and project management. The unit also enhances students' employability by developing critical thinking, analytical reasoning and an awareness of professional responsibility in the use of emerging technologies.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Define the core concepts of emerging technologies

LO2 Explain the foundational principles of a range of emerging technologies

LO3 Assess the potential impacts and opportunities for an organisation of an identified emerging technology

LO4 Discuss the ethical, social, and legal considerations for an organisation of an identified emerging technology.

## Essential Content

### LO1 **Define the core concepts of emerging technologies**

*Types of emerging technologies:*

For example, artificial intelligence (AI), including machine learning (ML) and generative AI; big data analytics and data visualisation tools; cloud computing services; Internet of Things (IoT); blockchain technologies; automation and robotics; digital platforms and enterprise systems.

*Core features and capabilities:*

Key technological capabilities including data processing and storage, real-time connectivity, automation of tasks, scalability, interoperability and integration with existing organisational systems

Consideration of how these features enable efficiency, flexibility and innovation in organisational contexts.

*Organisational applications of emerging technologies:*

Application of technologies across business functions, including use of AI in customer service, e.g. chatbots, analytics in decision-making, automation in operations and logistics and cloud-based tools for collaboration and remote working.

*Technology context and drivers:*

Drivers of technological adoption including digital transformation strategies, global competition, organisational need for efficiency, innovation and responsiveness

The role of technology in enabling new business models, including platform-based and data-driven approaches.

### LO2 **Explain the foundational principles of a range of emerging technologies**

*Foundational principles of emerging technologies:*

Underlying principles of emerging technologies, including how systems process data, automation logic, connectivity frameworks and integration across digital environments

Impact on business processes including increased efficiency, automation of routine tasks, improved accuracy, and real-time monitoring and decision-making.

Influence on productivity and operational performance.

*Strategic impact on organisations:*

Role of emerging technologies in shaping competitive advantage, enabling innovation, supporting organisational agility and facilitating new business models such as digital platforms and service-based offerings.

*Workforce implications:*

Impact on employment roles, including job redesign, automation of tasks and demand for new digital skills

Consideration of human–technology collaboration, remote and hybrid working environments, and the need for continuous skills development.

*Risks, governance and challenges:*

Challenges associated with technology adoption including cybersecurity risks, data protection and privacy issues, ethical considerations such as algorithmic bias, and organisational challenges relating to cost, implementation and change management.

*Data-driven organisational performance:*

Use of data analytics, dashboards and performance monitoring tools to measure organisational outcomes and inform decision-making.

LO3 **Assess the potential impacts and opportunities for an organisation of an identified emerging technology**

*Selection and application of digital tools in practice:*

Identifying and selecting technologies based on organisational objectives, operational needs and performance challenges

Consideration of suitability, cost, scalability and alignment with organisational strategy

Assessment of technology effectiveness using key performance indicators (KPIs), Return on Investment (ROI), operational outcomes and organisational performance measures

Use of emerging technologies such as analytics platforms, collaboration systems, automation tools and AI applications to support business functions, improve processes and enhance decision-making.

*Integration, implementation and problem-solving:*

Embedding technologies within organisational systems and workflows, including consideration of change management, user adoption, training and alignment with organisational culture and strategy

Application of emerging technologies to address organisational challenges, improve efficiency, enhance customer experience and support innovation and continuous improvement.

*Development of digital skills:*

Building digital literacy, data interpretation skills and adaptability required to operate effectively in technology-enabled environments.

LO4 **Discuss the ethical, social, and legal considerations for an organisation of an identified emerging technology**

*Ethical considerations:*

Data privacy, transparency, bias in AI, responsible use of technology and organisational accountability.

*Social implications:*

Impact on employment, digital inclusion, accessibility, user behaviour and societal change.

*Legal and regulatory frameworks:*

Data protection regulations, compliance requirements, intellectual property and organisational responsibilities, practice currency.

*Impact on organisations and stakeholders:*

How ethical, social and legal considerations influence decision-making, adoption and use of emerging technologies

Consideration of the effects of technology on employees, customers and organisational culture, including user experience, acceptance and behavioural change.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Define the core concepts of emerging technologies</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate how emerging technologies support organisational performance, innovation and operational effectiveness.</td></tr>
<tr><td>P1 Define different types of emerging technologies used in organisational contexts.<br><br>P2 Identify key features and functions of emerging technologies.</td><td>M1 Analyse how emerging technologies support organisational processes.</td></tr>
<tr><td colspan="2">LO2 Explain the foundational principles of a range of emerging technologies</td></tr>
<tr><td>P3 Explain foundational principles of selected emerging technologies.<br><br>P4 Describe how emerging technologies influence organisational operations.</td><td>M2 Analyse the relationship between emerging technologies and organisational outcomes.</td></tr>
<tr><td colspan="2">LO3 Assess the potential impacts and opportunities for an organisation of an identified emerging technology</td></tr>
<tr><td>P5 Assess the potential impacts of an emerging technology in an organisation.<br><br>P6 Describe opportunities created by the use of emerging technologies.</td><td>M3 Analyse the suitability of an emerging technology that supports organisational needs and challenges.</td><td>D2 Evaluate the effectiveness of an emerging technology in supporting organisational performance.</td></tr>
<tr><td colspan="2">LO4 Discuss the ethical, social, and legal considerations for an organisation of an identified emerging technology</td><td rowspan="2">D3 Critically evaluate the impact of ethical, social and legal considerations on organisational adoption of emerging technologies.</td></tr>
<tr><td>P7 Discuss ethical, social and legal considerations of an emerging technology.<br><br>P8 Review organisational responses to ethical, social and legal considerations when implementing emerging technologies.</td><td>M4 Evaluate how ethical, social and legal factors influence organisational decision-making.<br><br>M5 Analyse the implications of ethical, social and legal considerations for organisations.</td></tr>
</table>

## Recommended Resources

### Textbooks

Auer, M.E. and May, D. (eds.) (2024) *Yearbook of Emerging Technologies in Learning*. Cham: Springer.

Fenton, A. *et al*. (2025) *Strategic digital transformation*. London: Routledge.

Khullar, V. *et al*. (2026) *Artificial Intelligence and Society 5.0: Issues, Opportunities, and Challenges*. London: Routledge.

Rogers, D.L. (2023) *The Digital Transformation Roadmap: Rebuild Your Organization for Continuous Change*. 2nd edn. New York: Columbia Business School Publishing.

Singh, P.D. and Angurala, M. (eds.) (2024) *Integration of Cloud Computing and IoT: Trends, Case Studies and Applications*. London: CRC Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.blockchain-council.org | Blockchain Council<br>Resources on blockchain<br>(General reference) |
| ico.org.uk | Information Commissioner's Office (UK)<br>Information from the UK's information regulator<br>(General reference) |
| www.ibm.com/topics/artificial-intelligence | IBM<br>What is artificial intelligence (AI)?<br>(General reference) |
| www.ibm.com/topics/internet-of-things | IBM<br>What is the Internet of Things (IoT)?<br>(General reference) |
| www.mckinsey.com/capabilities/mckinsey-digital/our-insights | McKinsey Digital<br>Tech and AI Insights<br>(Article) |

| | |
|---|---|
| setr.stanford.edu | Stanford Emerging Technology Review<br>Reporting on Key Technology Areas and their Policy Implications<br>(Report) |
| www.unesco.org/en/artificial-intelligence | UNESCO<br>Artificial Intelligence and emerging technologies<br>(Report) |
| www3.weforum.org | World Economic Forum<br>Emerging technologies reports<br>(Report) |
| www.futureinsights.com | Emerging technology trends<br>(General reference) |
| oecd.ai/en/ | OECD.AI Policy Observatory<br>Policies, data and analysis for trustworthy artificial intelligence<br>(General reference) |
| www.gartner.com/en | Gartner<br>Gartner Insights<br>(Reports) |

## Journals and articles

Chaudhuri, R.R. *et al*. (2024) 'Assessing the influence of emerging technologies on organizational data-driven culture and innovation capabilities', *Technological Forecasting and Social Change*, 198, 123165. Available at: doi.org/10.1016/j.techfore.2023.123165.

Kashefi, P., Kashefi, Y. and Mirsaraei, A.H.G. (2024) 'Shaping the future of AI: balancing innovation and ethics in global regulation', *Uniform Law Review*, 29(3), pp. 524–548. Available at: doi.org/10.1093/ulr/unae040.

Kühn, U. and Williams, H. (2025) 'Governing the impact of emerging technologies: Actors, technologies, and regulation', *Journal of Strategic Studies*, 48(3), pp. 559–570. Available at: doi.org/10.1080/01402390.2025.2472213.

Singh, J. *et al*. (2025) 'A Systematic Review of Blockchain, AI, and Cloud Integration for Secure Digital Ecosystems', *International Journal of Networked and Distributed Computing*, 13, Article 28. Available at: doi.org/10.1007/s44227-025-00072-1.

## Links

This unit links to the following related units:

*Unit 403: Professional Practice*

*Unit 409: Core Cybersecurity*

*Unit 411: Core Data Analytics*

*Unit 420: Core AI*

*Unit 421: AI Automation and Digital Productivity*

*Unit 422: Ethics and AI*

*Unit 426: Digital Project Management.*

# Unit 431: Operating Systems

**Unit code:** F/652/3283

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Modern computing systems rely on operating systems to manage hardware resources, support applications and provide secure, stable environments for users. From personal computers and mobile devices to enterprise servers and cloud platforms, operating systems play a central role in how digital technologies function in practice. Understanding how operating systems operate beneath the user interface is essential for anyone pursuing a career in computing, IT support or systems administration.

The aim of this unit is to develop students' understanding of how operating systems manage processes, memory, storage and security within real computing environments. Students will explore how operating systems co-ordinate multiple tasks, allocate and protect system resources, and enforce controls that support performance, reliability and security. The unit also introduces the practical application of operating system knowledge, reflecting professional expectations in the workplace.

The unit covers a range of core topics that underpin effective operating system design and use. Key areas include process and memory management techniques, such as scheduling, concurrency and virtual memory, which enable systems to run efficiently. Students will also examine how file systems and storage structures are organised and maintained, how operating systems implement security through access control and monitoring, and how scripting can be used to automate tasks and support system administration.

On successful completion of this unit, students will be able to explain how operating systems manage key system resources and apply this knowledge in practical and professional contexts. Students will have developed technical understanding alongside transferable skills such as problem-solving, analytical thinking and technical communication. These skills are directly relevant to roles including IT Support Technician, Junior Systems Administrator, Infrastructure Technician, DevOps Assistant and Junior Cybersecurity Analyst, and support progression to further study or employment in computing and IT-related fields.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss how process and memory management is carried out by an identified operating system

LO2 Examine how file systems and storage structures are managed by an identified operating system

LO3 Explore how security is managed by an identified operating system

LO4 Deliver a presentation on developing OS scripts for an identified OS scenario to an identified audience that shows professionalism.

## Essential Content

LO1 **Discuss how process and memory management is carried out by an identified operating system**

*Process management:*

Process life cycle and execution: process states e.g. new, ready, running, waiting, terminated; process creation e.g. fork operations, process initialisation, process spawning; process termination e.g. normal termination, error termination, forced termination

Scheduling and execution control: scheduling concepts e.g. CPU scheduling, time-sharing, priority scheduling; scheduling approaches e.g. pre-emptive scheduling, non-pre-emptive scheduling, round-robin scheduling; multitasking models e.g. cooperative multitasking, pre-emptive multitasking, time-sliced multitasking

Concurrency and co-ordination: concurrency e.g. parallel execution, overlapping execution, thread concurrency; context switching e.g. state saving, state restoration, execution switching; process synchronisation e.g. mutual exclusion, locking mechanisms, co-ordination primitives

Inter-process communication principles e.g. message passing, shared memory, pipes; contention issues e.g. competing resource access, scheduling conflicts, lock contention; deadlock conditions e.g. circular wait, resource hold, dependency chains; deadlock handling approaches e.g. prevention, avoidance, detection.

*Memory management:*

Memory allocation and organisation: memory allocation e.g. fixed partitioning, dynamic partitioning, contiguous allocation; address spaces e.g. logical addresses, physical addresses, virtual addresses

Virtual memory mechanisms: virtual memory e.g. abstraction of physical memory, demand paging, memory illusion; paging concepts e.g. page frames, page tables, page faults; page replacement concepts e.g. first-in-first-out (FIFO) replacement, least recently used, clock algorithm

Memory protection and control: memory protection mechanisms e.g. access control, isolation enforcement, protection rings; fragmentation types e.g. internal fragmentation, external fragmentation, memory wastage; swapping operations e.g. disk swapping, process suspension, memory eviction

Performance and resource trade-offs: resource trade-offs e.g. memory usage versus speed, protection versus efficiency, responsiveness versus stability.

LO2 **Examine how file systems and storage structures are managed by an identified operating system**

*File systems management:*

File organisation and structure: file organisation methods e.g. contiguous allocation, linked allocation, indexed allocation; directory structures e.g. single-level directories, hierarchical directories, tree structures; access methods e.g. sequential access, direct access, indexed access

File metadata and access control: file metadata e.g. file attributes, timestamps, ownership information; permissions models e.g. read permissions, write permissions, execute permissions; path and namespace structures e.g. absolute paths, relative paths, namespace hierarchy

File system mounting and namespace management: file system mounting e.g. mount points, mount tables, automatic mounting; namespace management approaches e.g. unified namespaces, virtual file systems, path resolution

Consistency and recovery: file consistency e.g. file integrity, metadata consistency, atomic operations; recovery methods e.g. crash recovery, consistency checking, repair utilities; performance considerations e.g. access latency, throughput, access patterns.

*Storage structures:*

Data structures and allocation: inode structures e.g. inode tables, metadata pointers, file descriptors; block allocation methods e.g. fixed block allocation, variable block allocation, extent-based allocation; allocation trade-offs e.g. storage efficiency, access speed, fragmentation risk

Journaling and reliability: journaling mechanisms e.g. write-ahead logging, transactional updates, log-based recovery; journaling modes e.g. write-back mode, ordered mode, data journaling mode; reliability considerations e.g. crash tolerance, fault recovery, data durability

Caching and buffering mechanisms: caching e.g. buffer cache, page cache, write caching; buffering strategies e.g. read buffering, write buffering, delayed writes; cache coherence e.g. cache synchronisation, stale data prevention, flush operation

Performance and consistency: fragmentation e.g. internal fragmentation, external fragmentation, block wastage; performance impacts e.g. access latency, throughput variation, recovery overhead; crash consistency concepts e.g. atomic updates, recovery guarantees, consistency states.

LO3 **Explore how security is managed by an identified operating system**

*Access control and authentication management:*

Authentication and authorisation: authentication mechanisms e.g. passwords, biometric authentication, token-based authentication; authentication factors e.g. knowledge factors, possession factors, inherence factors; authorisation structures e.g. permission assignment, access rights, role allocation

User privileges and access enforcement: user and group permissions e.g. file permissions, directory permissions, execution permissions; privilege levels e.g. user mode, administrator mode, kernel mode; least privilege e.g. minimal access rights, task-specific permissions, role restriction; access control lists (ACLs) e.g. permission entries, user mappings, resource bindings

Privilege management: privilege levels e.g. user mode, administrator mode, kernel mode; least privilege e.g. minimal access rights, task-specific permissions, role restriction; privilege escalation e.g. vertical escalation, horizontal escalation, permission misuse.

*Secure operating system design:*

Secure design and isolation principles: secure design principles e.g. defence in depth, secure defaults, fail-safe design; isolation principles e.g. process isolation, memory isolation, user separation; sandboxing concepts e.g. restricted execution environments, permission confinement, application containment

System hardening and configuration management: attack surface concepts e.g. exposed services, open ports, enabled features; configuration control e.g. secure configurations, service disabling, feature minimisation; policy enforcement concepts e.g. security policies, rule enforcement, configuration controls

Kernel-level security mechanisms: kernel security mechanisms e.g. secure boot, kernel hardening, memory protection enforcement; mandatory access control systems e.g. SELinux, AppArmor, kernel security modules; privilege boundary enforcement e.g. system calls, controlled resource access, kernel mediation.

*Security monitoring and integrity management:*

Auditing and logging management: auditing e.g. security auditing, event tracking, accountability; logging e.g. system logs, security logs, audit trails; log management e.g. log rotation, log retention, log review

Integrity protection: integrity principles e.g. file integrity, system integrity, trust verification; integrity monitoring e.g. checksum validation, hash comparison, integrity checks; tamper detection e.g. unauthorised changes, integrity violations, alert generation

Usability and security balance: usability and security trade-offs e.g. ease of use versus protection, accessibility versus restriction, productivity versus control.

LO4 **Deliver a presentation on developing OS scripts for an identified OS scenario to an identified audience that shows professionalism**

*Operating system scripting concepts:*

Scripting purpose and automation: scripting purposes e.g. task automation, system administration, routine tasks; automation concepts e.g. repeatable processes, scheduled execution, efficiency improvement; command structures e.g. command syntax, arguments, flags

Execution environments and variables: script execution environments e.g. shell environments, command interpreters, runtime contexts; environment variable concepts e.g. system variables, user variables, configuration values

File, process and permission handling: file and directory operation concepts e.g. file creation, modification, deletion; process control concepts e.g. process execution, process termination, job control; permissions concepts in scripting e.g. script permissions, execution rights, access enforcement

Input, output and error handling: input and output redirection e.g. standard input, standard output, standard error; pipeline use e.g. command chaining, stream processing, data flow; error handling methods e.g. exit codes, error messages, exception handling; logging output e.g. log files, status reporting, diagnostic output

Optimisation and refinement: script efficiency e.g. streamlined logic, fewer steps; resource efficiency e.g. lower CPU usage, reduced memory usage; code quality e.g. clean structure, reusable functions; usability improvement e.g. clear outputs, readable logs; feedback use e.g. audience feedback, iterative updates.

*Audience-appropriate communication:*

Audience focus: audience identification e.g. target audience, stakeholders; result adaptation e.g. tailored outputs, adjusted detail; terminology selection e.g. appropriate language, technical terms

Presentation standards: visual formats e.g. charts, tables, figures; professional standards e.g. reporting conventions, technical standards; technical communication e.g. structured explanation, formal presentation.

*Professional practice:*

Audience awareness and communication: audience identification e.g. learner profiles, prior knowledge levels, learning needs; audience adaptation e.g. content adjustment, pace adjustment, language choice; inclusive communication and accessibility concepts e.g. clear language, accessible formats, inclusive delivery

Presentation management: delivery style e.g. demonstrative delivery, guided instruction, interactive explanation; professional conduct e.g. punctuality, appropriate language, professional behaviour; time management e.g. presentation timing, pacing; presentation structure and pacing e.g. clear introduction, logical sequencing, structured conclusion

Feedback and reflective practice: Q&A handling concepts e.g. question clarification, response accuracy, audience engagement; feedback capture concepts e.g. verbal feedback, written feedback, evaluation forms; reflective awareness e.g. self-evaluation, improvements, skills development.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Discuss how process and memory management is carried out by an identified operating system | | **LO1, LO2 and LO3**<br>**D1** Evaluate, using a selection of real-world examples, the effectiveness of an identified operating system in managing processes, memory, files and security for a range of organisational contexts. |
| **P1** Discuss, using a selection of real-world examples, how process management is carried out by an identified operating system.<br>**P2** Review, using a selection of real-world examples, how memory management is carried out by an identified operating system. | **M1** Analyse, using a selection of real-world examples, how process management techniques support efficient execution and resource allocation within an identified operating system. | |
| **LO2** Examine how file systems and storage structures are managed by an identified operating system | | |
| **P3** Examine, using a selection of real-world examples, how file systems are managed by an identified operating system.<br>**P4** Explain, using a selection of real-world examples, how storage structures are managed by an identified operating system. | **M2** Analyse, using a selection of real-world examples, how file system and storage structures support efficient data storage and retrieval within an identified operating system. | |
| **LO3** Explore how security is managed by an identified operating system | | |
| **P5** Explore, using a selection of real-world examples, how secure OS system design helps manage and reduce security risks.<br>**P6** Examine, using a selection of real-world examples, how authentication and monitoring operations are managed by an identified operating system. | **M3** Assess, using a selection of real-world examples, the benefits and limitations of how an identified operating system manages system security for a range of organisational contexts. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Deliver a presentation on developing OS scripts for an identified OS scenario to an identified audience that shows professionalism | | |
| **P7** Deliver a presentation on developing OS scripts for an identified OS scenario to an identified audience that shows professionalism.<br><br>**P8** Review the feedback from the identified audience on the scripts that have been developed to identify strengths and areas for improvement. | **M4** Optimise, using the results of the audience feedback, the developed OS scripts so that they fully meet the requirements of the OS scenario. | **D2** Evaluate the effectiveness of the optimised OS scripts in how comprehensively they meet the requirements of the OS scenario. |

## Recommended Resources

### Textbooks

Arpaci-Dusseau, R.H. and Arpaci-Dusseau, A.C. (2023) *Operating Systems: Three Easy Pieces*. 1.10 edn. Madison, WI: Arpaci-Dusseau Books.

Blum, R. and Bresnahan, C. (2021) *Linux Command Line and Shell Scripting Bible*. 4th edn. Hoboken, NJ: Wiley.

Stallings, W. and Brown, L. (2022) *Computer Security: Principles and Practice*. 5th edn. Harlow: Pearson.

Stallings, W. (2023) *Operating Systems: Internals and Design Principles*. 10th edn. Harlow: Pearson Global Editions.

Tanenbaum, A.S. and Bos, H. (2023) *Modern Operating Systems*. 5th edn. Harlow: Pearson Global Editions.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/operating-systems/states-of-a-process-in-operating-systems/ | GeeksforGeeks<br>States of Process in Operating Systems<br>(General reference) |
| www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm | TutorialsPoint<br>Operating System Scheduling Algorithms<br>(General reference) |
| codelucky.com/virtual-memory-paging-segmentation/ | CodeLucky<br>Virtual memory, paging and segmentation<br>(Article) |
| www.geeksforgeeks.org/operating-systems/file-systems-in-operating-system/ | GeeksforGeeks<br>File Systems in Operating Systems<br>(General reference) |
| phoenixnap.com/glossary/filesystem | phoenixNAP<br>What Is a File System?<br>(Article) |

| | |
|---|---|
| www.guru99.com/file-systems-operating-system.html | Guru99<br>File Systems in Operating Systems<br>(Article) |
| www.geeksforgeeks.org/operating-systems/operating-system-security/ | GeeksforGeeks<br>Operating System Security<br>(General reference) |
| www.numberanalytics.com/blog/access-control-essentials-operating-systems | Number Analytics<br>Access Control Essentials for Operating Systems<br>(Article) |
| www.geeksforgeeks.org/operating-systems/shell-in-operating-system/ | GeeksforGeeks<br>Shell in Operating Systems<br>(General reference) |
| intellipaat.com/blog/what-is-shell-scripting/ | Intellipaat<br>What Is Shell Scripting?<br>(Article) |

**Journals and articles**

Akhtar, Z.B. (2024) 'Operating Systems (OS): An Insight Investigative Research Analysis and Future Directions', *Journal of Technology and Informatics*, 6(1). Available at: doi.org/10.37802/joti.v6i1.637.

Bhurtel, M. and Rawat, D.B. (2023) 'Unveiling the landscape of operating system vulnerabilities', *Future Internet*, 15(7), 248. Available at: doi.org/10.3390/fi15070248.

Ghosh, A. (2025) 'Operating system security: Protection policies and mechanisms', *JETNR*, 3(1). Available at: rjpn.org/jetnr/papers/JETNR2501035.pdf.

Pathak, P.C., Nadeem, M. and Ansar, S.A. (2024) 'Security assessment of operating systems using decision-making algorithms', *International Journal of Information Technology*, 17, pp. 3609–3618. Available at: doi.org/10.1007/s41870-023-01706-9.

**Links**

This unit links to the following related units:

*Unit 404: Digital Systems Analysis*

*Unit 409: Core Cybersecurity*

*Unit 432: System Architecture*

*Unit 435: Client-Server Computer Systems.*

# Unit 432: System Architecture

**Unit code:** H/652/3284

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Modern computing relies on well-designed system architecture to deliver performance, reliability and scalability across a wide range of digital environments. From personal computers and enterprise servers to cloud platforms and embedded devices, architectural decisions determine how hardware components work together to process data efficiently. As organisations continue to expand their use of digital systems, there is increasing demand for computing professionals who understand how system structure affects capability, limitations and behaviour at a fundamental level.

The aim of this unit is to develop learners' understanding of the principles, models and design considerations that underpin modern computer systems architecture. Learners will explore how processors, memory, storage and interconnect technologies operate together within computing systems, and examine how architectural decisions influence system performance and scalability. The unit also introduces the practical application of architectural concepts through the design of computer systems that address identified operational and performance requirements.

The unit covers the core principles that underpin computer system architecture, including processor operation, instruction execution, memory organisation and system interconnects. Students explore common architectural models and examine how different components interact within a complete system. The unit also introduces key performance concepts such as throughput, latency and bottlenecks, before bringing these ideas together through the design of a computer system for a given scenario, where performance and scalability requirements must be considered.

On successful completion of this unit, students will be able to explain architectural principles, analyse how system design affects performance, and design computing systems that meet defined requirements. These skills support progression into roles such as Systems Technician, Junior Infrastructure Engineer, IT Support Engineer, Junior Systems Architect and Technical Operations Analyst, as well as further study in areas including operating systems, networking and cloud infrastructure.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the fundamental principles of computer system architecture

LO2 Describe core architectural models and system components

LO3 Discuss how system architecture impacts performance

LO4 Design a computer system for performance and scalability based on an identified scenario.

## Essential Content

LO1 **Explain the fundamental principles of computer system architecture**

*Processor fundamentals and instruction execution:*

Processor components: processor components e.g. CPU, control unit, arithmetic logic unit (ALU); register e.g. general-purpose, special-purpose; clocking e.g. clock cycles, clock speed

Instruction architecture: instruction set architecture e.g. complex instruction set computer (CISC), reduced instruction set computer (RISC); instruction format e.g. opcode, operands; operand representation e.g. registers, memory references

Execution flow: instruction cycle stages e.g. fetch, decode, execute; micro-operation e.g. data movement, arithmetic operations; execution sequencing e.g. sequential execution, branching

Control mechanisms: interrupt e.g. timer interrupt, I/O interrupt; exception e.g. divide-by-zero, invalid instruction; service routine e.g. interrupt handling, exception handling; priority handling e.g. interrupt levels, priority masks; addressing e.g. immediate, direct, indirect.

*Memory organisation and memory models:*

Memory access: hierarchy e.g. registers, cache, main memory; cache e.g. L1 cache, L2 cache; secondary storage e.g. hard disc drive (HDD), solid-state drive (SSD); locality principles e.g. temporal locality, spatial locality

Addressing models: addressing e.g. logical addressing, physical addressing, virtual addressing; memory e.g. pagination, abstraction; paging e.g. pages, frames; address translation e.g. page tables, translation lookaside buffer (TLBs)

Memory protection: protection e.g. access control, isolation; consistency e.g. shared memory consistency, cache coherence; model e.g. shared memory models, distributed shared memory

Mapping storage: memory-mapped I/O e.g. RAM-mapped devices, ROM-mapped firmware; storage characteristics e.g. volatile, non-volatile.

*System interconnects, buses and input/output:*

Bus control: data bus e.g. data transfer lines, width; address bus e.g. memory addressing, device addressing; control bus e.g. read signals, write signals; arbitration e.g. bus prioritisation, access scheduling; interconnect standards e.g. PCIe, USB

Controllers: interconnect e.g. system buses, interconnect fabrics; controller e.g. I/O controllers, disk controllers; device controller e.g. storage controllers, network controllers

I/O subsystems: subsystem e.g. device management, driver interaction; transfer e.g. direct memory access, programmed I/O; buffering e.g. input buffers, output buffers; interrupt-driven I/O e.g. interrupt signalling, interrupt requests; polling e.g. status checking, busy waiting.

LO2 **Describe core architectural models and system components**

*Architectural models and stored-program concepts:*

Architectural models: architectural models e.g. von Neumann, Harvard; stored-program concepts e.g. instructions in memory, data in memory

Pathway organisation: organisation models e.g. shared pathways, separate pathways; handling e.g. instruction flow, data flow

Bus organisation and trade-offs: organisation structures e.g. single-bus, multi-bus; co-ordination e.g. contention management, access sequencing; trade-offs e.g. simplicity, performance.

*Core system components and subsystems:*

Processor subsystems: subsystem components e.g. CPU cores, execution units, control logic; co-ordination processes e.g. instruction processing, task co-ordination

Memory and storage subsystem: memory subsystem e.g. main memory, cache memory, registers; storage subsystem e.g. HDD, SSD, Non-Volatile Memory Express (NVMe); access methods e.g. block access, sequential access

Input/output and network subsystem: I/O subsystem e.g. peripheral management, device interfacing; network interface e.g. network interface controller (NIC), network adapters; data exchange processes e.g. packet transmission, frame transfer

Subsystem interaction and component roles: subsystem boundary e.g. functional separation, interface boundaries; dependencies e.g. data dependencies, control dependencies; component roles e.g. graphics processing unit (GPU) processing, NIC communication, storage control.

*Controllers, interfaces and co-ordination mechanisms:*

Control and signalling mechanisms: control signals e.g. command signals, status signals, control flags; co-ordination mechanisms e.g. synchronisation, timing co-ordination

Interfaces and protocols: interface standards e.g. Serial Advanced Technology Attachment (SATA), NVMe, PCIe; protocol rules e.g. transfer rules, timing constraints, handshaking; compatibility e.g. interface limits, version support

Addressing and mapping: device addressing e.g. identifiers, addresses, device IDs; mapping methods e.g. resource mapping, device binding, address translation

Co-ordination and transfer concepts: shared resource co-ordination e.g. arbitration, scheduling, prioritisation; data transfer mode e.g. programmed I/O, Direct Memory Access (DMA), interrupt-driven transfer.

LO3 **Discuss how system architecture impacts performance**

*Performance measures and observable characteristics:*

Performance: performance measures e.g. throughput, latency, jitter; timing e.g. response time, turnaround time, deadline compliance; capacity e.g. utilisation, bandwidth, transaction rate

Workload and demand characteristics: workload e.g. CPU-bound, I/O-bound, memory-bound; demand patterns e.g. peak load, sustained load, burst load; workload variability e.g. predictable workloads, fluctuating workloads

Measurement and baseline: measurement e.g. benchmarking, sampling, profiling; baseline e.g. reference performance levels, baseline metrics; comparison e.g. before-and-after measurement, trend comparison

Performance trade-off: trade-off e.g. speed versus cost, capacity versus efficiency, performance versus reliability, responsiveness versus resource usage.

*Bottlenecks, contention and limiting factors:*

Resource bottlenecks and contention: bottleneck e.g. constrained resources, saturation points; contention e.g. shared CPU access, shared memory access, disk I/O contention; queuing e.g. wait queues, service delays, queue length

Architecture-specific limits and overheads: limiting factors e.g. cache misses, memory latency, instruction throughput; interconnect limits e.g. bus contention, memory channel limits; overheads e.g. interrupt handling, context switching, system call overhead.

*Optimisation mechanisms and architectural improvement options:*

Concurrency and parallelism optimisation: concurrency e.g. task concurrency, thread concurrency, asynchronous execution; parallelism e.g. multicore execution, simultaneous processing, single instruction, multiple data (SIMD) execution; co-ordination e.g. synchronisation, task co-ordination, lock management

Execution and memory optimisation: cache hierarchy e.g. multi-level caches, shared caches, cache coherence; locality optimisation e.g. data locality, instruction locality, temporal locality; execution e.g. pipelining, branch prediction, out-of-order execution

Scaling and constraint management: scaling e.g. core count scaling, parallel workload scaling, load distribution; constraint e.g. memory bandwidth limits, interconnect capacity, thermal limits; fallback strategies e.g. graceful degradation, workload throttling.

LO4 **Design a computer system for performance and scalability based on an identified scenario**

*Requirements, constraints and scenario parameters:*

Workload and usage requirements: workload profiles e.g. peak load, sustained load, burst load; usage patterns e.g. periodic demand, continuous demand, irregular demand

Performance and scalability requirements: performance requirements e.g. response time targets, throughput targets, latency thresholds; scalability requirements e.g. growth capacity, concurrent users, transaction volume

Operational constraints and assumptions: environmental constraints e.g. physical space, power limits, cooling capacity; operational constraints e.g. availability expectations, compatibility requirements, maintenance windows; assumption concepts e.g. usage assumptions, growth assumptions, technology assumptions.

*System configuration, modularity and component selection:*

Configuration and modularity: modularity e.g. component separation, interchangeable modules, independent upgrades; integration boundary e.g. subsystem interfaces, data exchange points, control boundaries

Component and subsystem selection: processor selection e.g. core count, clock speed, instruction set support; memory selection e.g. capacity, speed, access latency; storage selection concepts e.g. storage type, throughput, access time

Balancing, redundancy and compatibility: balancing e.g. CPU-memory balance, I/O balance, compute–storage balance; redundancy e.g. fault tolerance, component duplication, failover support; compatibility e.g. interface compatibility, driver support, operating system support.

*Scalability and performance design considerations:*

Scaling approaches and concurrency design: vertical scaling e.g. resource expansion, hardware upgrades, capacity enhancement; horizontal scaling e.g. node addition, workload distribution, service replication; concurrency design e.g. concurrent task handling, thread allocation, process synchronisation

Capacity planning, risk and future-proofing: capacity planning e.g. forecast growth, resource headroom, utilisation targets; performance risk e.g. bottleneck risk, saturation risk, contention risk; future-proofing e.g. upgrade paths, scalability margins, technology life cycle support.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the fundamental principles of computer system architecture</td><td rowspan="6">LO1, LO2 and LO3<br><br>D1 Evaluate the effectiveness of computer system architectures, models and components in delivering a high performance computer system to an identified organisational context.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, how the processor fundamentals and system interconnects contribute towards instruction execution.<br><br>P2 Explore, using a selection of real-world examples, how memory organisation and models are implemented in identified computer architectures.</td><td>M1 Analyse, using a selection of real-world examples, how fundamental computer system architecture principles interact to support system operation and data processing.</td></tr>
<tr><td colspan="2">LO2 Describe core architectural models and system components</td></tr>
<tr><td>P3 Describe, using a selection of real-world examples, the core computer architecture models and system components.<br><br>P4 Explain, using a selection of real-world examples, the components and control mechanisms used by modern computer system architectures.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of different architectural models and components used in a range of identified computer systems.</td></tr>
<tr><td colspan="2">LO3 Discuss how system architecture impacts performance</td></tr>
<tr><td>P5 Discuss, using a selection of real-world examples, how performance measures and limiting factors can determine the overall performance of a system.<br><br>P6 Explore, using a selection of real-world examples, how optimising a system architecture can impact system performance.</td><td>M3 Assess, using a selection of real-world examples, the benefits and limitations of different optimisation strategies on a computer system architecture.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Design a computer system for performance and scalability based on an identified scenario | | |
| **P7** Design a computer system that meets identified performance and scalability requirements of an identified scenario.<br><br>**P8** Review the design of the computer system with identified stakeholders to identify strengths and areas for improvement. | **M4** Evaluate, using the stakeholder feedback, the effectiveness of the designed system in meeting performance and scalability requirements of the identified scenario. | **D2** Justify the decisions made in the design of the computer system in how comprehensively it addresses the performance and scalability requirements of the identified scenario. |

## Recommended Resources

### Textbooks

Hennessy, J.L. and Patterson, D.A. (2025) *Computer Architecture: A Quantitative Approach*. 7th edn. Cambridge, MA: Morgan Kaufmann.

Null, L. and Lobur, J. (2023) *The Essentials of Computer Organization and Architecture*. 6th edn. Burlington, MA: Jones & Bartlett Learning.

Patterson, D.A. and Hennessy, J.L. (2023) *Computer Organization and Design RISC-V Edition: The Hardware/Software Interface*. 3rd edn. Cambridge, MA: Morgan Kaufmann.

Shiva, S.G. (2025) *Computer Organization, Design, and Architecture*. Boca Raton, FL: CRC Press.

Stallings, W. (2021) *Computer Organization and Architecture: Designing for Performance*. 11th edn. Harlow: Pearson Education Limited.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.ibm.com/think/topics/central-processing-unit | IBM<br>What is a central processing unit (CPU)?<br>(General reference) |
| www.geeksforgeeks.org/computer-organization-architecture/memory-hierarchy-design-and-its-characteristics/ | GeeksforGeeks<br>Memory Hierarchy Design and its Characteristics<br>(Article) |
| www.geeksforgeeks.org/computer-organization-architecture/computer-organization-risc-and-cisc/ | GeeksforGeeks<br>Difference between RISC and CISC processors<br>(Article) |
| www.geeksforgeeks.org/virtual-memory-in-operating-system/ | GeeksforGeeks<br>Virtual Memory in Operating System<br>(Article) |
| www.cloudflare.com/learning/performance/what-is-latency/ | Cloudflare<br>What is latency?<br>(General reference) |

| | |
|---|---|
| www.learncomputerscienceonline.com/von-neumann-architecture/ | Learn Computer Science Online<br>Von Neumann Architecture<br>(General reference). |
| www.numberanalytics.com/blog/ultimate-guide-harvard-architecture-microprocessors | Number Analytics<br>Mastering Harvard Architecture<br>(General reference) |
| www.geeksforgeeks.org/computer-organization-architecture/computer-organization-performance-of-computer/ | GeeksforGeeks<br>Performance metrics in computer architecture<br>(Tutorial) |
| www.cloudflare.com/learning/performance/what-is-load-balancing/ | Cloudflare<br>What is load balancing?<br>(General reference). |

**Journals and articles**

Hennessy, J.L. and Patterson, D.A. (2019) 'A new golden age for computer architecture', *Communications of the ACM*, 62(2), pp. 48–60. Available at: doi.org/10.1145/3282307.

Jouppi, N.P. *et al.* (2017) 'In-Datacenter Performance Analysis of a Tensor Processing Unit', in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 1–12. Available at: doi.org/10.1145/3079856.3080246.

Zhao, Y. *et al*. (2024) 'A Systematic Mapping Study on Architectural Approaches to Software Performance Analysis', *arXiv*. Available at: doi.org/10.48550/arXiv.2410.17372.

**Links**

This unit links to the following related units:

*Unit 404: Digital Systems Analysis*

*Unit 409: Core Cybersecurity*

*Unit 431: Operating Systems*

*Unit 435: Client-Server Computer Systems.*

 © Pearson Education Limited 2026

# Unit 433: Game Development

**Unit code:** J/652/3285

**Unit level:** 4

**Credit value:** 15

---

## Introduction

The game development sector is a dynamic and expanding part of the digital and creative industries, supporting applications across entertainment, education, simulation and interactive media. Modern games combine creative design with technical implementation, requiring an understanding of how gameplay systems, user interaction and programming concepts work together. As the industry continues to evolve, there is increasing demand for individuals who can apply both creative and technical thinking to develop engaging digital experiences.

The aim of this unit is to provide students with a foundation in the key principles of game development. It focuses on how game design theory influences player experience, how programming concepts control interactive behaviour, and how development tools and game engines support the production process. The unit is intended to build technical awareness alongside analytical skills, enabling students to understand how design and implementation decisions are made.

This unit covers core topics essential to game development. Students will explore game design frameworks, gameplay structures and player interaction systems, as well as fundamental programming concepts such as logic, state management and timing. The unit also examines game engines and development tools, including how assets, scripts and workflows are managed. This learning is brought together through the design and development of an interactive game prototype for an identified scenario.

On successful completion of this unit, students will be able to explain key game design concepts, apply fundamental programming principles and develop a simple interactive prototype. They will also be able to review development tools and evaluate design and technical decisions. These skills support progression into roles such as Junior Game Designer, Game Programmer, Technical Designer, QA Tester and Interactive Media Developer, as well as further study in game development or computing-related disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the fundamental concepts of game design theory

LO2 Explore the fundamental programming concepts for games

LO3 Review a range of game engines and tools

LO4 Develop an interactive digital game prototype for an identified scenario.

## Essential Content

LO1 **Explain the fundamental concepts of game design theory**

*Game design frameworks:*

Game design theory concepts: game design theory foundations e.g. gameplay systems, player experience, interactive design principles; design purpose e.g. player engagement, meaningful interaction, structured gameplay experiences; system relationships e.g. interaction between rules, player actions, outcomes; player-centred design e.g. user experience, player behaviour, engagement outcomes

Game design theory frameworks: theoretical frameworks e.g. mechanics-dynamics-aesthetics, formal design models, system-based design models; framework application e.g. analysing gameplay systems, structuring player interaction, linking mechanics to player experience.

Core system components: mechanics e.g. player movement, combat systems; dynamics e.g. player interactions, physics reactions; game rules e.g. scoring systems, gameplay restrictions; game states e.g. menu states, gameplay states

Player objectives and outcomes: player objectives e.g. mission goals, objectives; win conditions e.g. level completion, boss defeat; loss conditions e.g. player failure, game over

Interaction and feedback systems: input systems e.g. controller input, keyboard input; output feedback e.g. visual effects, audio cues; aesthetics e.g. game visuals, sound design.

*Gameplay structures:*

Progression systems: progression systems e.g. experience systems, unlock systems; level progression e.g. levels, stages; progression paths e.g. skill trees, branching levels

Challenge and balance: challenge scaling e.g. enemy difficulty, level challenge; difficulty curves e.g. increasing challenge, boss difficulty; pacing models e.g. exploration, combat phases

Structure and sequencing: core loops e.g. play-reward cycles, repeat actions; mission structures e.g. main quests, side objectives.

*Player interaction and experience:*

Player control and interaction: interaction models e.g. first-person control, third-person control; control schemes e.g. controller layouts, keyboard controls; player agency e.g. player choices, decision making

Engagement and motivation: engagement types e.g. competitive play, cooperative play; immersion factors e.g. game worlds, audio design; motivation drivers e.g. rewards, achievements

Usability and behaviour: usability principles e.g. intuitive controls, clear interfaces; accessibility factors e.g. subtitles, control remapping; user behaviour e.g. exploration, combat actions.

*Design consistency and coherence:*

System coherence and logic: system coherence e.g. consistent mechanics, unified systems; internal logic e.g. rule consistency, predictable outcomes; system stability e.g. stable gameplay, reliable performance

Design alignment and balance: mechanic alignment e.g. consistent controls, balanced mechanics; experience consistency e.g. gameplay consistency, uniform feel; balance factors e.g. difficulty tuning, enemy balance

Clarity and predictability: rule clarity e.g. clear objectives, simple rules; interaction predictability e.g. consistent responses, expected behaviour; visual consistency e.g. consistent visuals, user interface (UI) layout.

*Effectiveness evaluation dimensions:*

Performance and engagement metrics: engagement indicators e.g. play time, session frequency; balance metrics e.g. success rates, difficulty outcomes; fairness measures e.g. equal challenge, balanced gameplay

Quality and usability measures: objective clarity e.g. clear goals, defined tasks; feedback quality e.g. clear UI feedback, audio signals; usability factors e.g. ease of control, navigation clarity

Classification and balance: genre conventions e.g. first-person shooter (FPS) gameplay, RPG systems; game typologies e.g. single-player games, multiplayer games; challenge-reward balance e.g. effort vs reward, risk vs reward.

LO2 **Explore the fundamental programming concepts for games**

*Game programming logic:*

Control flow and logic: control flow e.g. game loop sequences, frame update order, logic steps; conditional statements e.g. player decisions, collision checks; iteration structures e.g. enemy spawn loops, animation loops, update cycles

Event and interaction logic: event-driven models e.g. input events, collision events, trigger events; input handling e.g. player controls, keyboard input, controller input; output responses e.g. visual feedback, sound effects.

*Game state management:*

State structures: game states e.g. menu screen, gameplay mode, pause mode; state transitions e.g. level changes, menu navigation, game over events; finite state machines e.g. enemy AI states, animation states

State control: state variables e.g. score values, health values; state control e.g. state updates, state switching, state tracking; state persistence e.g. save data, player progress, checkpoints.

*Interaction and responsiveness:*

Event handling and input: event handling e.g. input events, collision events; player input processing e.g. movement input, action input, control input; system reactions e.g. character movement, UI updates, physics responses

Responsiveness factors: timing controls e.g. input timing, cooldown timers; responsiveness factors e.g. frame rate, system responsiveness; input latency e.g. controller delay, input lag, response delay.

*Timing and update model concepts:*

Update cycles: frame updates e.g. render frames, animation frames; update loops e.g. main game loop, update cycle; execution timing e.g. frame timing, update timing

Timing control: delta time e.g. frame time, time scaling; timing accuracy e.g. consistent frame rate, stable updates; synchronisation e.g. animation sync, physics sync, audio sync.

*Selection criteria for programming constructs:*

Performance and efficiency: performance factors e.g. frame rate, memory usage; efficiency considerations e.g. optimised loops, reduced processing; optimisation factors e.g. code optimisation, asset optimisation, performance tuning

Code quality: readability factors e.g. clear code structure, naming conventions; maintainability e.g. modular code, reusable code, easy updates; modularity e.g. game components, scripts, modules.

LO3 **Review a range of game engines and tools**

*Game engine architecture:*

Core engine systems: engine components e.g. scene systems, object systems; rendering systems e.g. lighting, shading, graphics pipelines; physics engines e.g. collision systems, gravity systems; input systems e.g. controller input, keyboard input

Integrated subsystems: audio systems e.g. sound effects, background audio; animation systems e.g. character animation, object animation; scripting systems e.g. behaviour scripts, event scripts.

*Asset and content pipelines:*

Asset management: asset import e.g. model import, texture import; asset organisation e.g. asset libraries, structured storage; file formats e.g. image formats, model formats

Content workflows: scene assembly e.g. level building, layout design; production workflows e.g. asset pipelines, development processes; resource handling e.g. asset storage, file management.

*Scripting and development environments:*

Development tools: Code editors e.g. script editors, text editors; debugging tools e.g. error logs, breakpoints; integrated environments e.g. development platforms, integrated development environments (IDEs)

Scripting systems: scripting languages e.g. C# scripting, Python scripting; visual scripting e.g. node systems, visual logic; script execution e.g. runtime scripts, behaviour scripts.

*Tool capabilities and limitations:*

Performance and compatibility: performance constraints e.g. frame rate limits, memory limits; platform compatibility e.g. PC platforms, console platforms; hardware requirements e.g. GPU requirements, RAM limits

Tool support: feature sets e.g. engine features, tool functions; documentation quality e.g. manuals, guides; community support e.g. forums, tutorials.

*Workflow and productivity factors:*

Development workflows: iteration cycles e.g. build-test cycles, design loops; development efficiency e.g. rapid prototyping, build iteration; deployment methods e.g. build exports, platform deployment

Collaboration tools: team workflows e.g. shared projects, coordinated tasks; collaboration tools e.g. version control, shared repositories; version tracking e.g. file versions, change tracking.

*Profiling and optimisation tooling concepts:*

Performance analysis: profiling methods e.g. performance tracking, usage analysis; performance metrics e.g. frame rates, load times; resource usage e.g. RAM usage, asset sizes

Optimisation techniques: optimisation techniques e.g. code optimisation, asset optimisation; resource management e.g. memory control, asset loading; bottleneck identification e.g. CPU limits, GPU limits.

*Real-world examples:*

A range of real-world game engines and tools e.g.

Game engines: Unity e.g. 2D games, 3D games, mobile games; Unreal Engine e.g. AAA games, real-time rendering; Godot e.g. indie games, lightweight projects

Development tools: GameMaker Studio e.g. 2D games, rapid prototyping; Construct 3 e.g. browser games, drag-and-drop development; Cocos2d-x e.g. mobile games, 2D engines

Version control and profiling tools: Perforce Helix Core e.g. asset version control, team collaboration; Unreal Insights e.g. performance profiling, frame analysis.

LO4 **Develop an interactive digital game prototype for an identified scenario**

*Game prototype design foundations:*

Core gameplay design: gameplay objectives e.g. mission goals, player objectives, tasks; core mechanics e.g. player actions, movement systems; interaction design e.g. player inputs, control systems

System behaviour design: player actions e.g. movement actions, interaction inputs; gameplay rules e.g. rule sets, behaviour rules; system behaviours e.g. object responses, game reactions.

*Scenario requirements and success criteria:*

Requirements definition: functional requirements e.g. gameplay features, system functions; non-functional requirements e.g. performance limits, usability needs; constraints e.g. time limits, scope limits

Success criteria: performance criteria e.g. frame rate targets, responsiveness levels; usability criteria e.g. ease of use, interface clarity; success metrics e.g. objective completion, player outcomes.

*Prototype construction concepts:*

Game build elements: scene composition e.g. level layout, scene setup; asset integration e.g. models, textures, audio; gameplay implementation e.g. mechanics setup, interaction logic

System configuration: logic systems e.g. scripting logic, behaviour systems; object placement e.g. environment positioning, object setup; interface elements e.g. UI components, heads-up display (HUD) elements.

*Testing and refinement concepts:*

Testing activities: functional testing e.g. feature testing, system checks; playtesting sessions e.g. user testing, gameplay testing; error identification e.g. bugs, faults

Refinement processes: issue tracking e.g. bug tracking, issue logs; iteration cycles e.g. testing cycles, improvement cycles; refinement actions e.g. updates, improvements.

*Test evidence and evaluation criteria:*

Test documentation: test evidence e.g. test logs, screenshots; defect categories e.g. minor issues, major issues; requirements traceability e.g. requirement mapping, test tracking

Evaluation records: test results e.g. pass results, fail results; usability findings e.g. user feedback, interface issues; performance observations e.g. frame rate issues, lag issues.

*Prototype evaluation:*

Evaluation criteria: requirement alignment e.g. feature matching, goal alignment; functionality assessment e.g. feature completeness, system operation; system performance e.g. performance levels, responsiveness

Improvement analysis: design strengths e.g. effective mechanics, strong features; design weaknesses e.g. bugs, usability issues; improvement areas e.g. updates, enhancements.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explain the fundamental concepts of game design theory | | **LO1, LO2 and LO3**<br>**D1** Evaluate the effectiveness of fundamental game design concepts, programming concepts and game engines and tools in supporting the development of modern computer game workflows. |
| **P1** Explain, using a selection of real-world examples, the fundamental concepts of game design theory, including frameworks, structures and player interaction.<br>**P2** Explore, using a selection of real-world examples, how design consistency and effectiveness evaluation contribute to the development of high quality computer games. | **M1** Analyse, using a selection of real-world examples, how fundamental game design concepts influence user experience. | |
| **LO2** Explore the fundamental programming concepts for games | | |
| **P3** Explore, using a selection of real-world examples, fundamental programming concepts used in game development, including logic, state management, interaction and timing.<br>**P4** Review, using a selection of real-world examples, how the selection criteria for programming constructs contribute to the development of high performance games. | **M2** Assess, using a selection of real-world examples, how fundamental programming concepts support game logic and interactive behaviour. | |
| **LO3** Review a range of game engines and tools | | |
| **P5** Review, using a selection of real-world examples, the core features of game engines used in modern computer game development.<br>**P6** Explore, using a selection of real-world examples, the core features of tools used in modern computer game development. | **M3** Assess, using a selection of real-world examples, the technical capabilities of identified game engines and tools in relation to development requirements. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Develop an interactive digital game prototype for an identified scenario | | |
| **P7** Design an interactive digital game prototype and test plan for an identified scenario.<br><br>**P8** Develop an interactive digital game prototype for an identified scenario.<br><br>**P9** Implement the test plan on the interactive digital game prototype to identify strengths and areas for improvement. | **M4** Evaluate, using the results of the testing, the effectiveness of the developed game prototype against the requirements of the identified scenario. | **D2** Justify the decisions made in the design of the interactive digital game prototype in how comprehensively it meets the requirements of the given scenario. |

## Recommended Resources

### Textbooks


Adams, E. (2022) *Fundamentals of Game Design*. 4th edn. Berkeley: New Riders (Pearson Education).

Fullerton, T. (2024) *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. 4th edn. Boca Raton: CRC Press.

Gregory, J. (2022) *Game Engine Architecture*. 3rd edn. Boca Raton: CRC Press.

Rogers, S. (2023) *Level Up! The Guide to Great Video Game Design*. 3rd edn. Chichester: Wiley.

Schell, J. (2023) *The Art of Game Design: A Book of Lenses*. 3rd edn. Boca Raton: CRC Press.

Shaker, N., Togelius, J. and Nelson, M.J. (2024) *Procedural Content Generation in Games*. 2nd edn. Cham: Springer.

Thompson, J. and Berbank-Green, B. (2023) *Game Design Theory: A New Philosophy for Understanding Games*. London: Bloomsbury.


### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.universityxp.com/blog/2025/1/14/designing-the-core-dynamics | University XP<br>Explains how game mechanics combine to form gameplay dynamics and player interactions<br>(Blog) |
| gamedev.net/tutorials/game-design/game-design-and-theory/mechanics-dynamics-aesthetics-r2983/ | GameDev.net<br>Introduces core principles of game design theory and how to structure gameplay experiences<br>(Tutorial) |
| realpython.com/pygame-a-primer/ | Real Python<br>Introduction to game programming using Python including game loops, events and input handling<br>(Tutorial) |

| | |
|---|---|
| www.pythonfaq.net/how-to-create-a-game-loop-in-pygame-for-real-time-updates-in-python/ | PythonFAQ<br>Explains how the game loop controls updates, rendering and responsiveness in games<br>(Article) |
| gamedevacademy.org/what-is-a-state-machine-complete-guide/ | GameDev Academy<br>Explains state machines and how they are used to control behaviour and logic in games<br>(Tutorial) |
| docs.unity3d.com/6000.3/Documentation/Manual/first-time-user.html | Unity Documentation<br>Official guide for beginners explaining how to create projects, navigate the Unity editor and understand core tools<br>(Tutorial) |
| hackr.io/blog/unity-vs-unreal-engine | Hackr<br>Compares major game engines and helps understand differences in tools, workflows and capabilities<br>(Blog) |
| www.geeksforgeeks.org/blogs/game-prototyping/ | GeeksforGeeks<br>Guide to game prototyping including how to test ideas and refine gameplay concepts<br>(Tutorial) |
| karnakgames.com/creating-a-game-prototype/ | Karnak Games<br>Explains how to create and develop a game prototype using core mechanics and iterative design<br>(Article) |
| moldstud.com/articles/p-playtesting-guide-for-designers-creating-engaging-player-experiences | MoldStud<br>Guide to playtesting techniques and using feedback to improve game prototypes<br>(Article) |

## Journals and articles

Chrysanthi, A., Katifori, A. and Kapralos, A. (2023) 'Point of Equilibrity: a narrative video game on decision-making during the COVID-19 pandemic crisis', *Frontiers in Computer Science*, 5. Available at: doi.org/10.3389/fcomp.2023.1178972.

Lu, A.S. and Moller, A.C. (2024) 'Elaborating the Role of Narrative and Self-Determination Theory in Video Game Design Research', *Interacting with Computers*. Available at: doi.org/10.1093/iwc/iwae058.

Pérez, C. *et al*. (2024) 'A Comparative Analysis of Energy Consumption Between The Widespread Unreal and Unity Video Game Engines', *Software Engineering*. Available at: doi.org/10.48550/arXiv.2402.06346.

Videnovik, M*. et al*. (2023) 'Game-based learning in computer science education: a scoping literature review', *International Journal of STEM Education*, 10. Available at: doi.org/10.1186/s40594-023-00447-2.

## Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science.*

# Unit 434: Green Computing

**Unit code:** **K/652/3286**

**Unit level:** **4**

**Credit value:** **15**

---

## Introduction

As organisations continue to depend on digital technologies to deliver services and support operations, the environmental impact of computing has become an increasingly important consideration. From data centres and cloud platforms to everyday devices and software, ICT systems consume energy, generate carbon emissions and contribute to electronic waste. Green computing focuses on reducing these impacts through more sustainable design, efficient resource use and responsible management of technology across its life cycle, making it a key area of practice for modern computing professionals.

The aim of this unit is to develop students' understanding of green computing and its role within an organisational context. It introduces the principles that underpin sustainable ICT, including energy efficiency, carbon awareness and resource optimisation, and explores how these can be applied in real-world scenarios. Students will gain insight into how environmental considerations are integrated into technical decision-making, and why sustainability is now a priority for organisations, regulators and wider society.

This unit explores a range of topics that support the development of practical and applied knowledge. Students will examine the environmental impact of hardware, software and ICT life cycle processes, including life cycle assessment, carbon measurement and e-waste management. The unit also reviews organisational strategies, policies and technologies that enable more sustainable computing, including infrastructure efficiency, virtualisation and cloud optimisation. Learning is applied through the development of a green computing strategy for a defined organisational scenario, allowing students to bring together technical understanding with planning and communication skills.

On successful completion of this unit, students will be able to assess the environmental impact of computing systems, apply green computing principles and present practical sustainability strategies for organisations. These skills support progression into roles such as IT Support Technician, Systems Administrator, Cloud Operations Technician, Sustainability Analyst and Data Centre Technician, or further study in related areas of computing and digital sustainability.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core principles of green computing

LO2 Explore the environmental impacts associated with hardware, software, and ICT life cycle processes

LO3 Review a range of organisational strategies and technologies that support green computing

LO4 Present a regulatory compliant green computing strategy for an identified organisational scenario to a range of key stakeholders.

## Essential Content

### LO1 **Explain the core principles of green computing**

*Green computing principles:*

Sustainability governance: sustainability e.g. long-term use, efficiency, environmental balance; environmental responsibility e.g. compliance, ethical practice, organisational accountability

Resource management: resource efficiency e.g. energy use, device utilisation, system efficiency; resource optimisation e.g. capacity planning, allocation efficiency, system tuning

Impact management: circular economy e.g. reuse, repair, recycling; waste minimisation e.g. reduction, reuse, controlled disposal

Life cycle perspective: life cycle awareness e.g. production, operation, end-of-life.

*Energy efficiency concepts:*

Power management: power consumption e.g. device energy use, system demand; power management e.g. sleep modes, shutdown processes, scheduling

Efficiency performance: performance per watt e.g. system output, how efficiently a system runs; efficiency metrics e.g. benchmarks, performance indicators, usage data

Workload optimisation: workload distribution e.g. load balancing, task allocation; demand management e.g. peak control, scheduling, usage limits

Energy control: energy optimisation e.g. efficiency improvement, energy reduction; energy regulation e.g. standards, operational limits, compliance controls.

*Carbon footprint concepts:*

Emissions concepts: greenhouse gas emissions e.g. carbon dioxide, methane; emissions measurement e.g. monitoring, data collection, tracking

Carbon accounting: carbon accounting e.g. tracking emissions, reporting frameworks; carbon intensity e.g. per system, per service, per device

Carbon sources: embodied carbon e.g. manufacturing impact, material sourcing; operational carbon e.g. energy use, cooling, system operation

Standards and scope: scope definitions e.g. Scope 1, Scope 2, Scope 3.

*ICT environmental impact factors:*

Life cycle impacts: life cycle impacts e.g. manufacture, operation, disposal; environmental footprint e.g. energy use, carbon output

Production systems: device manufacture e.g. assembly processes, fabrication methods, testing procedures; infrastructure systems e.g. data centres, network systems

Supply chain: supply chains e.g. suppliers, logistics networks, sourcing processes; resource extraction e.g. mining, raw material sourcing

End-of-life: end-of-life processes e.g. reuse, refurbishment, recycling; disposal methods e.g. landfill, incineration.

LO2 **Explore the environmental impacts associated with hardware, software, and ICT life cycle processes**

*ICT life cycle and assessment concepts:*

Life cycle assessment: life cycle assessment e.g. product stages, system evaluation, environmental review; life cycle metrics e.g. usage data, impact results, comparison results

System modelling: system boundaries e.g. system limits, what is included in the system; impact modelling e.g. scenario modelling, system analysis, expected results

Process analysis: process mapping e.g. workflow diagrams, process steps; material flows e.g. resource inputs, outputs, waste streams.

*Hardware environmental impacts:*

Material impacts: raw materials e.g. metals, plastics, minerals; rare earth elements e.g. lithium, cobalt, neodymium

Production impacts: manufacturing emissions e.g. carbon output, factory emissions; transport emissions e.g. shipping emissions, logistics emissions

Device life cycle: device lifespan e.g. usage period, durability, replacement cycle; repairability e.g. component replacement, maintenance, upgrades

Waste impacts: electronic waste e.g. discarded devices, obsolete hardware; hazardous substances e.g. toxic materials, chemical waste.

*Software and service environmental impacts:*

Efficiency factors: computational efficiency e.g. processing load, energy demand; storage efficiency e.g. data size, storage use

Data impacts: data transfer e.g. network usage, data movement; data intensity e.g. high data processing, large datasets

System operation: software complexity e.g. code size, processing demand; system utilisation e.g. resource usage, active workloads

Service demand: cloud energy demand e.g. data centre usage, server demand; processing overheads e.g. background processes, extra computation.

*End-of-life and circularity:*

Reuse systems: reuse pathways e.g. second use, redistribution; refurbishment processes e.g. repair, reconditioning

Recycling systems: recycling streams e.g. material recovery, waste sorting; remanufacturing systems e.g. rebuilding products, component reuse

Disposal controls: secure disposal e.g. safe disposal, data destruction; compliance standards e.g. legal requirements, industry standards

Regulation and impact: Waste Electrical and Electronic Equipment (WEEE) regulations e.g. e-waste rules, recycling laws; disposal impacts e.g. landfill impact, environmental damage.

LO3 **Review a range of organisational strategies and technologies that support green computing**

*Organisational governance and policy:*

Governance frameworks: sustainability strategy e.g. environmental planning, carbon reduction goals, sustainability objectives aligned to UN Sustainable Development Goals (SDGs), Science Based Targets initiative (SBTi); green IT policy e.g. energy policies, environmental guidelines, usage policies aligned to Energy Star standards, EU Code of Conduct for Data Centres, The Green Grid PUE framework

Procurement and compliance: procurement standards e.g. supplier selection, sustainability criteria, purchasing rules; compliance frameworks e.g. ISO 14001 Environmental Management Systems, ISO 50001 Energy Management Systems, ISO 14064 Greenhouse Gas (GHG) reporting, WEEE Directive, RoHS Directive, GHG Protocol, UK Environment Act

Organisational structure: organisational roles e.g. IT roles, sustainability roles, management roles; governance structures e.g. committees, reporting lines, decision processes.

*Energy and power management technologies:*

Device management: device power states e.g. sleep modes, standby modes, shutdown states; endpoint management e.g. device monitoring, remote control, update management

Energy optimisation: server consolidation e.g. reducing servers, combining workloads, system merging; power optimisation e.g. energy tuning, load reduction, efficiency improvements

Energy monitoring: energy monitoring e.g. metering tools, tracking systems, dashboards; energy controls e.g. power limits, usage policies, control systems.

*Infrastructure efficiency approaches:*

Virtualisation systems: virtualisation e.g. virtual machines, virtual servers; containerisation e.g. containers, lightweight applications

Infrastructure optimisation: hardware consolidation e.g. reducing devices, combining systems, resource sharing; storage optimisation e.g. data compression, storage management, capacity reduction

Data centre management: cooling systems e.g. air cooling, liquid cooling, airflow systems; capacity planning e.g. resource planning, scaling decisions, usage frameworks ISO/IEC 30134.

*Cloud and service optimisation:*

Cloud efficiency: cloud efficiency e.g. resource use, service optimisation, energy efficiency frameworks e.g. Amazon Web Services (AWS) Well-Architected Sustainability Pillar, Microsoft Cloud for Sustainability, Google Cloud Carbon Footprint methodology; resource right-sizing e.g. scaling resources, adjusting capacity, usage matching

Workload management: autoscaling policies e.g. automatic scaling, demand response, load adjustment; workload allocation e.g. distribution, balancing, scheduling

Service optimisation: managed services e.g. cloud services, hosted services, platform services; architecture patterns e.g. system design, how services are organised, ways systems are set up.

LO4 **Present a regulatory compliant green computing strategy for an identified organisational scenario to a range of key stakeholders**

*Organisational scenario baseline:*

Organisational context: organisational context e.g. business type, organisational structure, service environment; service portfolio e.g. services offered, digital services, operational services

Infrastructure overview: infrastructure architecture e.g. system design, network structure, platform setup; asset inventory e.g. hardware assets, software assets, system components

Performance baseline: energy profile e.g. energy usage, power demand, consumption patterns; baseline metrics e.g. energy measures, performance data, usage statistics.

*Strategy objectives and measures:*

Sustainability targets: sustainability objectives e.g. environmental goals, sustainability aims, organisational targets; carbon reduction targets e.g. emission reduction, carbon limits, footprint reduction

Performance measurement: success metrics e.g. performance indicators, measurement criteria, evaluation metrics; reporting frameworks e.g. reporting standards, audit reporting, compliance reporting

Compliance requirements: compliance requirements e.g. legal standards, regulatory requirements, policy obligations; evaluation criteria e.g. assessment criteria, performance benchmarks, review standards, current practice.

*Strategy components:*

Policy development: policy updates e.g. revised policies, updated procedures, governance changes; procurement strategy e.g. supplier selection, sustainability criteria, purchasing requirements

Operational changes: infrastructure changes e.g. system upgrades, hardware replacement, configuration changes; cloud optimisation e.g. resource allocation, workload balancing, service optimisation

Engagement activities: training programmes e.g. staff training, awareness training, skills development; awareness initiatives e.g. communication campaigns, user engagement, sustainability awareness.

*Stakeholder communication and presentation:*

Communication planning: audience requirements e.g. stakeholder needs, user requirements, communication expectations; presentation structure e.g. report format, presentation layout, content organisation

Analysis and reporting: benefit analysis e.g. cost savings, efficiency gains, environmental benefits; risk communication e.g. risk factors, mitigation strategies, impact assessment

Visual communication: visual methods e.g. charts, diagrams, dashboards; system diagrams e.g. system diagrams, network diagrams.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explain the core principles of green computing | | **LO1 and LO2** |
| **P1** Explain, using a selection of real-world examples, the core principles of green computing and energy efficiency.<br><br>**P2** Review, using a selection of real-world examples, the concepts of carbon footprints and the environmental impact of modern computing systems. | **M1** Analyse, using a selection of real-world examples, how the core principles of green computing influence sustainable ICT practices. | **D1** Evaluate, using a selection of real-world examples the effectiveness of green computing principles in mitigating the environmental impact of hardware, software, and ICT life cycle processes. |
| **LO2** Explore the environmental impacts associated with hardware, software, and ICT life cycle processes | | |
| **P3** Describe, using a selection of real-world examples, the ICT life cycle and end of life processes.<br><br>**P4** Explore, using a selection of real-world examples, the environmental impacts associated with hardware and software use in modern computing systems. | **M2** Assess, using a selection of real-world examples, the environmental impacts of hardware, software, and ICT life cycle processes in relation to sustainability. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Review a range of organisational strategies and technologies that support green computing | | **LO3 and LO4**<br>**D2** Justify the decisions made in the development of the regulatory compliant green computing strategy to allow it to comprehensively meet organisational sustainability objectives. |
| **P5** Review, using a selection of real-world examples, the standards, regulations and frameworks that affect green computing.<br>**P6** Review, using a selection of real-world examples, organisational strategies and technologies that support green computing. | **M3** Analyse, using a selection of real-world examples, the effectiveness of organisational strategies, technologies and regulatory frameworks used to support green computing. | |
| **LO4** Present a regulatory compliant green computing strategy for an identified organisational scenario to a range of key stakeholders | | |
| **P7** Develop a green computing strategy for an identified organisational scenario that complies with relevant regulations and standards.<br>**P8** Present a regulatory compliant green computing strategy for an identified organisational scenario to a range of key stakeholders to obtain feedback. | **M4** Evaluate, using the results of the stakeholder feedback, the effectiveness of the green computing strategy in meeting organisational and sustainability objectives. | |

## Recommended Resources

### Textbooks

Buyya, R., Calheiros, R.N. and Dastjerdi, A.V. (2024) *Sustainable Cloud Computing*. 2nd edn. Cambridge: Elsevier.

Murugesan, S. and Gangadharan, G.R. (2023) *Harnessing Green IT: Principles and Practices*. Hoboken: Wiley.

Rashid, M. (2024) *Green Information Technology: Sustainability in ICT Systems*. London: Routledge.

Singh, N. *et al*. (2025) *Green Computing for Sustainable Development*. 1st edn. Boca Raton: Auerbach Publications (Taylor & Francis).

Tomlinson, B. (2024) *Greening through IT: Information Technology for Environmental Sustainability*. 2nd edn. Cambridge, MA: MIT Press.

Unhelkar, B. (2022) *Green IT Strategies and Applications: Using Environmental Intelligence*. Boca Raton: CRC Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.ibm.com/think/topics/green-computing | IBM<br>What is green computing?<br>(General reference) |
| www.iea.org/topics/digitalisation | International Energy Agency<br>Tracking Digitalisation<br>(Report) |
| www.iso.org/iso-14001-environmental-management.html | International Organization for Standardization<br>Environmental management standards<br>(General reference) |
| www.itu.int/en/ITU-D/Environment | International Telecommunication Union<br>Creating a circular economy for electronics and greening digital transformation<br>(Report) |

| | |
|---|---|
| www.qaa.ac.uk/the-quality-code/education-for-sustainable-development | Quality Assurance Agency (QAA)<br>Education for Sustainable Development<br>(General reference) |
| weee-forum.org/overview/ | WEEE Forum<br>Resources in the WEEE Forum toolbox<br>(Resource Hub) |
| docs.aws.amazon.com/wellarchitected/latest/sustainability-pillar/cloud-sustainability.html | Amazon Web Services<br>Cloud Sustainability – Sustainability Pillar<br>(Tutorial) |
| standardsexplained.com/iso-50001-energy-management-systems/ | Standards Explained<br>ISO 50001 Energy Management Systems explained<br>(General reference) |
| www.datacenterknowledge.com/sustainability/what-is-data-center-pue-defining-power-usage-effectiveness | Data Center Knowledge<br>What is Data Center PUE? Power Usage Effectiveness<br>(Article) |

**Journals and articles**

Anjum, A.A.R. *et al*. (2025) 'Green IT: Exploring Sustainable Technologies for Reducing the Carbon Footprint of IT Operations', *Lecture Notes in Networks and Systems*. Available at: doi.org/10.1007/978-981-96-7505-0_38.

Hasan Amiri, S.M. *et al*. (2025) 'Green Computing: The Ultimate Carbon Destroyer for a Sustainable Future', *arXiv* (Computers and Society). Available at: doi.org/10.48550/arXiv.2508.00153.

Pazienza, A. *et al*. (2024) 'A holistic approach to environmentally sustainable computing', *Innovations in Systems and Software Engineering*, 20, pp. 347–371. Available at: doi.org/10.1007/s11334-023-00548-9.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science.*

# Unit 435: Client-Server Computer Systems

**Unit code:** L/652/3287

**Unit level:** 4

**Credit value:** 15

---

## Introduction

Client–server systems form a fundamental part of modern computing, supporting a wide range of applications including websites, enterprise systems, cloud platforms and distributed services. In many areas of the digital sector, these systems allow users to access shared resources and services efficiently through structured interactions between client devices and centralised servers. An understanding of how these systems operate is essential for students entering computing and provides a strong foundation for further technical development.

The aim of this unit is to develop students' understanding of how client–server systems are structured, how they communicate, and how they can be designed to meet specific requirements. The unit introduces both key concepts and practical approaches, enabling students to explore how theory is applied in real-world contexts. It provides a foundation for progression into related areas such as software development, networking and cloud technologies.

The unit content is organised to build knowledge progressively from core concepts through to practical implementation. Students will first examine the principles of client–server architecture, including system roles, structures and communication protocols. They will then explore how data is exchanged through mechanisms such as APIs, messaging systems and error handling processes. This knowledge is applied in the design of a basic client–server solution, before students go on to implement and test a functional application within a guided environment.

On successful completion of this unit, students will be able to explain key client–server concepts, design structured solutions, and implement and evaluate working systems. These skills support entry into a range of computing roles, including Junior Software Developer, Systems Administrator, Network Technician, IT Support Specialist, and Trainee Cloud or DevOps Engineer.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the principles of client–server architecture

LO2 Explore the different mechanisms of client–server communication

LO3 Design a basic client–server solution for an identified scenario

LO4 Implement a functional client–server application for an identified scenario.

## Essential Content

LO1 **Explain the principles of client–server architecture**

*Client–server architecture models:*

Architecture structures: centralised architectures e.g. single server systems, mainframe systems, central databases; distributed architectures e.g. multiple servers, internal LAN-based client–server system, Amazon Web Services (AWS) EC2 hosted service/
Azure web app

Tiered design: two-tier architectures e.g. client-database models, simple applications; three-tier architectures e.g. presentation layer, application layer, data layer; n-tier architectures e.g. multi-layer systems, enterprise systems, scalable architectures

Client types: thick clients e.g. desktop applications, local processing; thin clients e.g. web browsers, remote processing

Design principles: scalability e.g. system growth, user expansion; performance e.g. response time, system speed; layering e.g. system layers, separation of concerns; modularity e.g. reusable components, independent modules.

*Roles and responsibilities within client–server systems:*

System actors: clients e.g. user devices, applications; servers e.g. application servers, database servers; services e.g. web services, application services

Processing and tasks: requests e.g. user requests, service calls; data processing e.g. processing login requests, handling form submissions, validating input data, retrieving database records; task allocation e.g. workload distribution, resource assignment; execution e.g. server processing requests, handling client requests

System management: sessions e.g. user sessions, active sessions; resources e.g. CPU, memory, storage, database tables; concurrency e.g. multiple users, parallel tasks; workloads e.g. system load, user demand

Control and co-ordination: co-ordination e.g. task co-ordination, system co-ordination; process control e.g. request handling control, process management; responsibility distribution e.g. role division, which component (client or server) performs each task.

*Communication principles and protocols:*

Communication models: request–response e.g. client request, server response; statelessness e.g. independent requests, no session data; statefulness e.g. stored sessions, persistent data

Timing and behaviour: synchronisation e.g. real-time communication, blocking calls; asynchronisation e.g. background tasks, non-blocking calls

Protocols and standards: protocols e.g. Hypertext Transfer Protocol (HTTP) request methods (GET, POST), API request and response formats; HTTP e.g. web communication, browser requests; HTTPS e.g. secure web communication, encrypted traffic; Transmission Control Protocol (TCP) e.g. reliable transport, connection-based communication; Internet Protocol (IP) e.g. addressing, packet delivery

Data handling: messaging e.g. message exchange, HTTP request and API response data; encoding e.g. data formats, character encoding; transmission e.g. sending HTTP requests between browser and server, transmitting API responses over TCP/IP.

*Network communication fundamentals:*

Addressing and connection: addressing e.g. IP addresses, device identification; ports e.g. service ports, application endpoints; sockets e.g. connection endpoints, communication interfaces

Data transfer: packets e.g. data packets, transmission units; bandwidth e.g. data capacity, transfer rate; latency e.g. delay time, response delay.

LO2 **Explore the different mechanisms of client–server communication**

*Data exchange mechanisms:*

Data structures and formats: data e.g. login form data, user profile data; formats e.g. JavaScript Object Notation (JSON), eXtensible Markup Language (XML), data formats; payloads e.g. login credentials, search queries, authentication response, search results

Data handling: encoding e.g. data encoding, character encoding, JSON or XML formatting; transport e.g. HTTP requests and responses over a network, communication channels; schemas e.g. data structure, validation rules, checking JSON or XML structure

Validation and processing: integrity e.g. duplicate username checks, data integrity validation; validation e.g. email format validation, password length checks, checking JSON or XML structure; transformation e.g. data conversion, format changes, processing.

*Application programming interfaces (APIs):*

API structure: APIs e.g. web APIs, application interfaces; interfaces e.g. system interfaces, communication interfaces; endpoints e.g. access points, service endpoints

Requests and responses: requests e.g. client requests, API calls; responses e.g. server responses, returned data; headers e.g. request headers, response headers, metadata

Access control: authentication e.g. login verification, token validation; authorisation e.g. access control, permission checks; parameters e.g. query parameters, input values

API management: versioning e.g. API versions, updates; resources e.g. service resources, data endpoints; uniform resource locators (URLs) e.g. resource identifiers, web addresses.

*Messaging patterns:*

Messaging models: messaging e.g. user registration event, order placed event; synchronisation e.g. real-time messages, blocking communication; asynchronisation e.g. delayed messages, background processing

Messaging systems: publish–subscribe e.g. topic-based systems, event notifications; queues e.g. message queues, task queues; events e.g. sending confirmation email, updating database record

Components: brokers e.g. message brokers, middleware systems; producers e.g. message senders, data sources; consumers e.g. message receivers, processing systems

Channels and structure: topics e.g. message topics, channels; subscriptions e.g. subscriber lists, topic subscriptions; protocols e.g. messaging rules, communication standards.

*Error handling and reliability mechanisms:*

Error management: errors e.g. API timeout error, failed HTTP request; exceptions e.g. runtime exceptions, API validation errors; faults e.g. database connection failure, server crash

Recovery mechanisms: timeouts e.g. request delays, response limits; retries e.g. retry attempts, repeated requests; recovery e.g. system recovery, error correction

System resilience: resilience e.g. system stability, fault handling; tolerance e.g. error tolerance, failure handling; redundancy e.g. backup systems, duplicate resources

Monitoring and reporting: logs e.g. system logs, error logs; monitoring e.g. system monitoring, performance tracking; reporting e.g. error reports, system reports; availability e.g. uptime, system access.

LO3 **Design a basic client–server solution for an identified scenario**

*Scenario requirements and constraints:*

Requirements definition: requirements e.g. user login functionality, product search requirement, account creation; constraints e.g. system limits, design restrictions, resource limits

User and system context: users e.g. end users, administrators, clients; stakeholders e.g. retail manager, IT administrator, end-user customer; boundaries e.g. system scope, operational limits

Operational factors: performance e.g. response time, throughput; security e.g. data protection, access control; scalability e.g. user growth, system expansion; availability e.g. uptime, service access.

*Client–server design models:*

Architecture design: architecture e.g. web application design, database system design, client–server solution design; components e.g. system components, service components, modules; layers e.g. presentation layer, application layer, data layer

System representation: diagrams e.g. client–server diagrams, request–response diagrams, flow diagrams; structures e.g. system structure, how client, server, and database are arranged; organisation e.g. grouping front end, backend, and database components

Component interaction: clients e.g. front-end systems, user interfaces; servers e.g. application servers, database servers; services e.g. backend services, web services

Deployment design: deployment e.g. system deployment, environment setup; interactions e.g. submitting login form, server validation, database query, response message.

*Interfaces and data flow:*

Interface design: interfaces e.g. web form interface, REST API endpoint; inputs e.g. user input, request data; outputs e.g. system output, response data

Data movement: flows e.g. login request flow, order processing flow; exchange e.g. data exchange, sending requests and receiving responses; payloads e.g. request payloads, response payloads

Integration and control: integration e.g. system integration, service integration; validation e.g. data validation, input checking; formatting e.g. formatting data as JSON or XML

Transfer paths: channels e.g. communication channels, data paths; paths e.g. transmission paths, routing paths.

*Workflow structures:*

Process sequences: workflows e.g. process workflows, system workflows; sequences e.g. user authentication flow, data retrieval flow; stages e.g. process stages, execution phases

Control mechanisms: control e.g. process control, workflow control; execution e.g. task execution, process execution; dependencies e.g. task dependencies, sequence dependencies

System interactions: interactions e.g. component interactions, system communication; relationships e.g. system relationships, component links; co-ordination e.g. task co-ordination, process co-ordination.

LO4 **Implement a functional client–server application for an identified scenario**

*Client–server implementation concepts:*

System components: clients e.g. user devices, applications; servers e.g. application servers, database servers; components e.g. authentication module, API controller, database access layer, routing component

Communication and interaction: interactions e.g. client–server exchanges, service communication; requests e.g. client requests, service calls; responses e.g. server responses, returned data

System operation: sessions e.g. user sessions, active connections; connections e.g. network connections, communication links; states e.g. system state, session state

Execution processes: communication e.g. sending API requests and receiving JSON responses, publishing event to message queue (e.g. order event); integration e.g. connecting client, server, and database; execution e.g. running Node.js server, opening web browser client.

*Deployment considerations:*

Deployment environment: deployment e.g. system deployment, application release; environments e.g. development environment, production environment; platforms e.g. operating systems, cloud platforms

System configuration: configurations e.g. system settings, application configuration; variables e.g. environment variables, system parameters; models e.g. local deployment, cloud deployment

Infrastructure setup: networks e.g. network setup, communication networks; hosting e.g. server hosting, cloud hosting; infrastructure e.g. hardware resources, network infrastructure

Distribution and release: distribution e.g. application distribution, system rollout; releases e.g. software release, version updates; pipelines e.g. deployment pipelines, release processes.

*Testing and verification concepts:*

Testing processes: testing e.g. testing login functionality, API endpoint testing; verification e.g. system verification, requirement checking; validation e.g. input validation, output validation

System evaluation: data e.g. sample user accounts, test order data; performance e.g. response time under load, API response latency; systems e.g. integrated systems, application systems

Quality assurance: integration e.g. integration testing, system integration; cases e.g. test cases, test scenarios; assurance e.g. quality assurance, system assurance

Fault identification: errors e.g. system errors, runtime errors; defects e.g. software defects, bugs; reliability e.g. system reliability, consistent operation.

*Evaluation considerations:*

Requirement evaluation: requirements e.g. system requirements, user requirements; functionality e.g. system features, functional behaviour

Performance review: performance e.g. response time under load, API response latency; usability e.g. user usability, interface interaction; behaviour e.g. system behaviour, response handling

Measurement criteria: criteria e.g. evaluation criteria, performance criteria; metrics e.g. response time, error rates; outcomes e.g. test results

Improvement factors: improvements e.g. system improvements, optimisation changes; constraints e.g. system constraints, resource limits; effectiveness e.g. successful login rate, transaction success rate.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explain the principles of client-server architecture | | **LO1 and LO2**<br>**D1** Evaluate the effectiveness of client-server architectural principles and communications mechanisms in delivering scalable and reliable networked systems. |
| **P1** Explain, using a selection of real-world examples, the core models of client-server architectures.<br>**P2** Describe, using a selection of real-world examples, the system roles and network communication principles used within client–server architectures. | **M1** Analyse, using a selection of real-world examples, how the principles of client-server architectures can help deliver effective organisational networks. | |
| **LO2** Explore the different mechanisms of client-server communication | | |
| **P3** Explore, using a selection of real-world examples, the different mechanisms of client–server communication, data exchange and messaging in organisational contexts.<br>**P4** Review, using a selection of real-world examples, how error handling and reliability is maintained in client–server communications. | **M2** Assess, using a selection of real-world examples, the strengths and limitations of different client–server communication mechanisms in supporting reliable data exchange. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a basic client–server solution for an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of the client-server solution in how effectively it meets the requirements of the identified scenario. |
| **P5** Design a basic client-server solution that addresses the requirements of an identified scenario.<br>**P6** Review the design documentation of the client-server solution with a range of internal stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the stakeholder feedback, the design of the client-server solution so that it comprehensively addresses the requirements of the identified enterprise scenario. | |
| **LO4** Implement a functional client–server application for an identified scenario | | |
| **P7** Implement a functional client-server application for an identified scenario, including deployment and testing considerations.<br>**P8** Review, using the results of a test plan, the performance of the client-server application in meeting the requirements of the identified enterprise scenario. | **M4** Evaluate, using the results of the test plan, the effectiveness of the developed client-server application in how well it matches the design and meets the requirements of the identified enterprise scenario. | |

## Recommended Resources

### Textbooks

Coulouris, G., Dollimore, J., Kindberg, T. and Blair, G. (2011) *Distributed Systems: Concepts and Design*. 5th edn. Harlow: Addison-Wesley.

Easttom, C. and Roberts, R.M. (2024) *Networking Fundamentals*. 4th edn. Tinley Park, IL: Goodheart-Willcox.

Kurose, J.F. and Ross, K.W. (2023) *Computer Networking: A Top-Down Approach*. 9th edn. Harlow: Pearson.

Stallings, W. (2015) *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*. Boston, MA: Pearson.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/First_steps/Client-Server_overview | Mozilla Developer Network (MDN)<br>Client–server overview and HTTP request–response fundamentals for web systems<br>(Tutorial) |
| www.geeksforgeeks.org/system-design/client-server-model/ | GeeksforGeeks<br>Client–server model explanation including roles, request–response flow and system behaviour<br>(General reference) |
| dev.to/iarchitsharma/http-request-and-response-explained-46l2 | DEV Community<br>Detailed explanation of HTTP request and response structure within client–server communication<br>(Article) |
| www.geeksforgeeks.org/node-js/rest-api-introduction/ | GeeksforGeeks<br>Introduction to REST APIs including HTTP methods, JSON/XML data exchange and CRUD operations<br>(Tutorial) |

 © Pearson Education Limited 2026 

| | |
|---|---|
| www.freecodecamp.org/news/how-to-use-rest-api/ | freeCodeCamp<br>Beginner guide to REST APIs explaining communication between client and server using HTTP<br>(Tutorial) |
| medium.com/@aslisemagultekin/exploring-apis-and-data-formats-restful-apis-xml-and-json-fa0992470236 | Medium<br>Overview of APIs with explanation of REST principles and JSON/XML data formats<br>(Article) |
| www.geeksforgeeks.org/distributed-systems/distributed-systems-tutorial/ | GeeksforGeeks<br>Introduction to distributed systems including scalability, communication and architecture concepts<br>(Tutorial) |
| www.baeldung.com/cs/distributed-systems-guide | Baeldung<br>Fundamentals of distributed systems covering scalability, reliability and system co-ordination<br>(General reference) |
| www.designgurus.io/blog/distributed-system-design-guide-for-beginners | DesignGurus<br>Beginner guide to distributed system design including scalability, fault tolerance and architecture<br>(Article) |
| www.networkacademy.io/ccna/network-fundamentals/bandwidth-latency-throughput | NetworkAcademy.io<br>Explanation of bandwidth, latency and throughput in network communication systems<br>(Tutorial) |

### Journals and articles

Nyabuto, G. (2024) 'Client-server Architecture: a Review', *International Journal of Advanced Science and Computer Applications*, 3(2). Available at: doi.org/10.47679/ijasca.v3i1.48.

Niswar, M. *et al.* (2024) 'Performance evaluation of microservices communication with REST, GraphQL, and gRPC', *International Journal of Electronics and Telecommunications*, 70(2), pp. 429–436. Available at: doi.org/10.24425/ijet.2024.149562.

Patil, O. and Shirbhate, A. (2024) 'Design and Implementation of a Java-Based Client-Server Application', *arXiv preprint*. Available at: doi.org/10.48550/arXiv.2404.10107.

Videla, A. (2024) 'Container Morphisms for Composable Interactive Systems', *arXiv preprint*. Available at: doi.org/10.48550/arXiv.2407.16713.

### Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science.*

# Unit 501: Computing Research Project (Pearson-set)

**Unit code:** **L/652/3330**

**Unit level:** **5**

**Credit value:** **30**

---

## Introduction

This unit is a **Pearson-set unit**. The project brief will be set by the centre, based on a theme provided by Pearson (this will change annually). The theme and chosen project within the theme will enable students to explore and examine a relevant and topical aspect of Computing in the context of a professional environment.

The purpose of this unit is to enable students to develop the research knowledge, skills and professional behaviours required to investigate a computing-related problem or opportunity in a structured and ethical manner. In contemporary computing roles, professionals are expected to base decisions on evidence, evaluate alternatives critically, and justify recommendations using appropriate research methods. This unit therefore positions applied research as an essential skill for progression within the computing sector.

The aim of the unit is to provide students with the opportunity to plan, conduct and evaluate a substantial computing research project. Students will engage in the full research cycle, including defining a research focus, selecting and applying appropriate methodologies, managing data responsibly, and evaluating outcomes against defined objectives. Emphasis is placed on independent working, critical evaluation and the effective communication of findings, reflecting expectations of performance at Level 5.

On successful completion of this unit, students will be able to undertake structured research activities, analyse findings critically and present evidence-based conclusions in a professional format. These capabilities support progression into computing-related roles where investigation, analysis and reporting are required, such as Software Developer, Data Analyst, Systems Analyst, Cyber Security Analyst, and IT Consultant.

***Please refer to the accompanying Pearson-set Assignment Guide and Theme and Topic Release document on HN Global for further support and guidance on the delivery of the Pearson-set unit.**

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Design an ethical research proposal for a computing project to meet a set of defined objectives

LO2 Implement the ethical research proposal for a computing project to meet a set of defined objectives

LO3 Review the outcomes of the ethical research proposal considering the effectiveness and limitations of the chosen methodology

LO4 Communicate the research findings to an intended audience.

## Essential Content

LO1 **Design an ethical research proposal for a computing project to meet a set of defined objectives**

*Research context and focus:*

Research contexts: applied computing research e.g. system improvement studies, application evaluation; academic research contexts e.g. coursework investigations, structured study tasks; industry research contexts e.g. organisational system analysis, technology feasibility studies; context types e.g. primary, secondary

Problem context: problem domains e.g. software performance issues, data processing problems; organisational needs e.g. efficiency improvement, decision support requirements

Research framing: computing disciplines e.g. software engineering, data analytics; research justification e.g. identified needs, technical limitations

Technology scanning: horizon scanning e.g. emerging technologies, new tools; innovation awareness e.g. evolving computing practices, industry trends; opportunity identification e.g. performance improvement, capability enhancement.

*Research question formulation:*

Research definition: research aims e.g. project purpose statements, investigation intentions; research objectives e.g. measurable targets, success criteria, stated outcomes

Problem structure: problem definition e.g. system faults, workflow inefficiencies; research alignment e.g. aim-objective consistency, question relevance

Feasibility control: feasibility e.g. time availability, learner capability; research boundaries e.g. technical constraints, organisational limitations.

*Research methodologies:*

Methodology types: qualitative methodology e.g. interviews, user feedback analysis; quantitative methodology e.g. numerical measurements, performance data; mixed methods e.g. combined feedback and metrics, triangulated approaches

Research sources: primary research e.g. original data collection, system testing; secondary research e.g. journals, technical documentation

Data collection approaches: data collection methods e.g. structured questionnaires, system logs; interviews e.g. stakeholder interviews, user discussions; surveys e.g. online surveys, feedback forms.

*Research design and planning:*

Project structure: research design e.g. investigation frameworks, structured approaches; project structure e.g. phased activities, task sequencing

Method planning: methodology justification e.g. approach suitability, method selection reasons; data requirements e.g. data types, required formats

Project constraints: time constraints e.g. deadlines, milestone schedules; resource constraints e.g. hardware limits, software availability.

*Ethical considerations:*

Ethical principles: research ethics e.g. ethical standards, responsible research; professional conduct e.g. codes of practice, expected behaviours, professional responsibility

Participant protection: informed consent e.g. consent forms, clear explanations; participant rights e.g. withdrawal rights, fair treatment

Data protection: confidentiality e.g. restricted access, secure handling; data protection e.g. legal compliance, GDPR principles.

LO2 **Implement the ethical research proposal for a computing project to meet a set of defined objectives**

*Research implementation:*

Implementation control: proposal adherence e.g. following approved plans, agreed objectives; procedural compliance e.g. use of stated methods, documented procedures; implementation consistency e.g. repeatable steps, consistent application

Project delivery: research execution e.g. carrying out planned activities, applying methods; task sequencing e.g. ordered steps, logical progression; milestone management e.g. progress checkpoints, deadline tracking

Resource use: resource utilisation e.g. effective tool use, appropriate system use; implementation documentation e.g. activity records, implementation logs.

*Data collection:*

Data quality: data gathering e.g. collecting required information, following methods; data accuracy e.g. correct values, error avoidance; data reliability e.g. repeatable results, consistent outcomes

Data validity: data validity e.g. relevant data, suitable measures; source credibility e.g. trusted sources, reliable references

Data integrity: data handling e.g. careful processing, structured handling; data integrity e.g. unchanged data, protected values.

*Ethical compliance during implementation:*

Ethical behaviour: ethical conduct e.g. responsible behaviour, ethical awareness; professional behaviour e.g. appropriate actions, respectful conduct; ethical responsibility e.g. duty of care, honest practice

Participant protection: consent compliance e.g. informed agreement, signed consent; participant rights e.g. right to withdraw, fair treatment

Sensitive data handling: sensitive data e.g. personal information, protected records; secure handling e.g. controlled access, secure storage.

*Data management:*

Data organisation: data organisation e.g. structured folders, logical arrangement; data storage e.g. secure servers, encrypted storage; access control e.g. permission settings, authorised access

Traceability: version control e.g. file versions, update tracking; data traceability e.g. source tracking, process links

Sustainability considerations: energy efficiency e.g. low-power processing, optimised resource usage; environmental impact e.g. data storage footprint, compute efficiency; life cycle impact e.g. design, implementation, evaluation stages.

*Quality assurance:*

Verification checks: quality checks e.g. reviewing outputs, checking procedures; error identification e.g. spotting mistakes, inconsistencies; error correction e.g. fixing errors, revising data

Process review: process monitoring e.g. progress observation, activity tracking; reflective practice e.g. self-review, improvement reflection.

LO3 **Review the outcomes of the ethical research proposal considering the effectiveness and limitations of the chosen methodology**

*Research outcomes:*

Outcome evaluation: research findings e.g. collected results, analysed outputs; outcome summaries e.g. key points, main results; objective alignment e.g. meeting aims, addressing objectives

Result relevance: result relevance e.g. usefulness of findings, relevance to problem; expected outcomes e.g. anticipated results, planned outputs; unexpected outcomes e.g. unforeseen results, additional findings.

*Methodological effectiveness:*

Method suitability: methodology suitability e.g. appropriate methods, suitable approach; method strengths e.g. reliable techniques, effective tools; research alignment e.g. method–aim match, question coverage;

Data quality: data quality e.g. accurate data, consistent results; data credibility e.g. trustworthy sources, dependable evidence; methodological robustness e.g. sound methods, stable procedures

Procedural effectiveness: procedural effectiveness e.g. smooth execution, effective steps; research validity e.g. suitable conclusions, valid results; methodological consistency e.g. repeated processes, standard application

Technology evaluation: tool comparison e.g. existing versus emerging technologies; performance analysis e.g. improved efficiency, enhanced capability; adoption considerations e.g. suitability, feasibility.

*Methodological limitations:*

Method constraints: method constraints e.g. limited access, restricted methods; sampling limitations e.g. small samples, limited participants; data limitations e.g. missing data, incomplete records

Contextual limitations: contextual limitations e.g. workplace factors, time pressures; reliability issues e.g. inconsistent results, measurement errors; bias influence e.g. researcher bias, participant bias

Scope restrictions: scope restrictions e.g. narrow focus, excluded variables; methodological weaknesses e.g. design flaws, tool limitations; data gaps e.g. unavailable data, incomplete coverage

Sustainability evaluation: environmental impact assessment e.g. resource consumption, processing efficiency; sustainability trade-offs e.g. performance versus energy usage; optimisation strategies e.g. reduced processing overhead, efficient data handling.

*Data interpretation:*

Analysis approach: data analysis e.g. basic analysis, pattern identification, anomaly detection; result comparison e.g. comparing findings, checking differences

Evidence use: evidence evaluation e.g. judging strength, checking relevance; result synthesis e.g. combining findings, drawing conclusions.

*Validity and reliability:*

Quality judgement: research validity e.g. appropriate conclusions, sound results; research reliability e.g. repeatable outcomes, consistent method on results; outcome trustworthiness e.g. confidence in findings, dependable evidence.

LO4 **Communicate the research findings to an intended audience**

*Audience identification:*

Audience identification: academic audiences e.g. tutors, student peers; professional audiences e.g. employers, project sponsors; technical stakeholders e.g. developers, IT staff; non-technical stakeholders e.g. managers, end users

Audience needs: audience needs e.g. information clarity, required detail; audience expectations e.g. professional standards, clear conclusions; audience awareness e.g. background knowledge, contextual understanding.

*Communication formats:*

Communication formats: written reports e.g. research reports, project documentation; oral presentations e.g. slide presentations, verbal briefings, posters, presentations, workshops; visual representations e.g. charts, tables, diagrams

Digital delivery: infographics e.g. visual summaries, data highlights; digital media e.g. videos, online content; structured documents e.g. formal reports, formatted submissions

Presentation tools: presentation tools e.g. slide software, document templates.

*Academic and professional conventions:*

Academic standards: logical structure e.g. clear sections, ordered content; academic style e.g. formal tone, objective language; technical terminology e.g. correct terms, subject vocabulary

Professional quality: referencing conventions e.g. citation systems, source listing; citation accuracy e.g. correct references, consistent formatting; professional tone e.g. clear language, appropriate wording

Written accuracy: language clarity e.g. understandable writing, precise meaning; grammatical accuracy e.g. correct grammar, punctuation.

*Effectiveness of communication:*

Communication quality: content organisation e.g. structured flow, logical order; message clarity e.g. clear explanations, focused points; information coherence e.g. connected ideas, consistent narrative

Audience alignment: evidence presentation e.g. use of data, supporting examples; audience alignment e.g. matching content to audience, suitable depth.

*Reflection on communication:*

Evaluation: message effectiveness e.g. clarity of message, achievement of purpose; format suitability e.g. appropriate media, suitable structure

Development: communication improvement e.g. refinement actions, skill development; feedback consideration e.g. tutor feedback, audience comments.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Design an ethical research proposal for a computing project to meet a set of defined objectives</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Justify the decisions made in the design, implementation and outcome review of the ethical research project in comprehensively addressing a set of defined objectives.</td></tr>
<tr><td>P1 Design an ethical research proposal for a computing project to meet a set of defined objectives.<br><br>P2 Review, with a range of identified stakeholders, the design of the ethical research proposal for a computing project to identify strengths and areas for improvement.</td><td>M1 Optimise, using the results of the stakeholder feedback, the design of the ethical research proposal to thoroughly address the defined objectives.</td></tr>
<tr><td colspan="2">LO2 Implement the ethical research proposal for a computing project to meet a set of defined objectives</td></tr>
<tr><td>P3 Implement the ethical research proposal for a computing project to meet a set of defined objectives.<br><br>P4 Review, with a group of internal stakeholders, the implemented ethical research proposal in terms of ethical compliance, data management and quality assurance.</td><td>M2 Analyse, using the results of the stakeholder review, the effectiveness of the research implementation of the computing project in meeting the defined objectives.</td></tr>
<tr><td colspan="2">LO3 Review the outcomes of the ethical research proposal considering the effectiveness and limitations of the chosen methodology</td></tr>
<tr><td>P5 Review the research outcomes to obtain meaningful insights and conclusions regarding the ethical research proposal.<br><br>P6 Review the implementation of the research project and the research outcomes in the context of the effectiveness and limitations of the selected research methodology.</td><td>M3 Evaluate the extent to which the chosen methodology influenced the validity of the research outcomes.</td></tr>
</table>

 © Pearson Education Limited 2026

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Communicate the research findings to an intended audience | | |
| **P7** Communicate research findings effectively to an intended audience using appropriate academic or professional conventions.<br><br>**P8** Obtain feedback from the intended audience to identify strengths and limitations of the communication approach used in presenting the research findings to the intended audience. | **M4** Analyse, using the results of the stakeholder feedback, the effectiveness of the communication approach used in presenting the research findings to the intended audience. | **D2** Critically evaluate the effectiveness of the communication of research findings in meeting the needs and expectations of the intended audience. |

## Recommended Resources

### Textbooks

Bergin, T. (2024) *An Introduction to Data Analysis: Quantitative, Qualitative and Mixed Methods.* London: SAGE Publications.

Creswell, J.W. and Creswell, J.D. (2023) *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. 6th edn. London: Sage Publications.

Hair, J.F. *et al*. (2024) *Essentials of Business Research Methods*. 5th edn. London: Routledge.

Markel, M. and Selber, S.A. (2021) *Technical Communication*. 13th edn. New York: Macmillan Learning.

Saunders, M., Lewis, P. and Thornhill, A. (2024) *Research Methods for Business Students*. 9th edn. Harlow: Pearson Education.

Walliman, N. (2023) *Research Methods: The Basics*. 3rd edn. London: Routledge.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.ukri.org/councils/esrc/guidance-for-applicants/research-ethics-guidance/ | UK Research and Innovation (UKRI)<br>Research ethics guidance<br>(General reference) |
| www.computer.org/education/code-of-ethics | IEEE Computer Society<br>Code of Ethics for computing professionals<br>(General reference) |
| www.acm.org/code-of-ethics | Association for Computing Machinery (ACM)<br>Code of Ethics and Professional Conduct<br>(General reference) |
| www.nist.gov/programs-projects/research-data-framework-rdaf | National Institute of Standards and Technology (NIST)<br>Research Data Framework overview<br>(General reference) |
| www.ukdataservice.ac.uk/manage-data/plan | UK Data Service<br>Managing and planning research data<br>(Tutorial) |

 © Pearson Education Limited 2026

| | |
|---|---|
| scientific-publishing.webshop.elsevier.com/manuscript-preparation/how-to-write-the-results-section-of-a-research-paper/ | Elsevier Author Services<br>Writing a results section<br>(Tutorial) |
| research-rebels.com/blogs/get-research-done/research-methodology-made-simple-the-complete-step-by-step-guide | ResearchRebels<br>Research Methodology Made Simple: The Complete Step-by-Step Guide<br>(Blog) |
| www.geeksforgeeks.org/business-studies/introduction-to-research-methodology/ | GeeksforGeeks<br>Introduction to Research methodology<br>(Article) |
| www.theiet.org/media/5182/technical-report-writing.pdf | Institution of Engineering and Technology (IET)<br>Guide to technical report writing<br>(PDF) |
| pressbooks.umn.edu/techwriting/chapter/4-5-reports/ | University of Minnesota<br>Technical and professional reports<br>(Article) |

**Journals and articles**

Davidson, R. and Makanjee, C. (2022) 'Communicating research findings', in *Research for Medical Imaging and Radiation Sciences*. Springer. Available at: doi.org/10.1007/978-3-030-79956-4_7.

McNutt, M.K. *et al*. (2018) 'Transparency in authors' contributions and responsibilities to promote integrity in scientific publication', *Proceedings of the National Academy of Sciences*, 115(11), pp. 2557–2560. Available at: doi.org/10.1073/pnas.1715374115.

Mouza, C. *et al*. (2022) 'Investigating the impact of research-based professional development on teacher learning and classroom practice: Findings from computer science education', *Computers & Education*, 186, 104530. Available at: doi.org/10.1016/j.compedu.2022.104530.

Yan, S. and Jones, J. (2026) 'Bridging the Gaps: A Comprehensive Review of Data Management Capability Maturity Models in the Digital Era', *IEEE Transactions on Big Data*, 12(2), pp. 307–320. Available at: doi.org/10.1109/TBDATA.2025.3621149.

**Links**

This unit links to the following related units:

*Unit 506: Advanced Software Design*

*Unit 510: Advanced Data Engineering*

*Unit 512: Advanced Data Analytics*

*Unit 515: Secure System Design*

*Unit 516: Internet of Things*

*Unit 520: Applied AI.*

 © Pearson Education Limited 2026

# Unit 502: Applied Computing Project (Pearson-set)

**Unit code:** **M/652/3288**

**Unit level:** **5**

**Credit value:** **30**

---

## Introduction

This unit is a **Pearson-set unit**. The project brief will be set by the centre, based on a theme provided by Pearson (this will change annually). The theme and chosen project within the theme will enable students to explore and examine a relevant and topical aspect of Computing in the context of a professional environment.

Computing professionals are increasingly expected to deliver solutions that respond directly to real organisational needs. This unit allows students to bring together knowledge and skills developed across their programme and apply them to a substantial, work-related computing project. It provides an opportunity to operate in a professional context, engaging with realistic project constraints, stakeholder expectations and technical requirements that reflect current industry practice.

The aim of this unit is to enable students to plan, design, implement and review a computing solution based on a **small-scale computing project**. Through an end-to-end project life cycle, students will undertake requirements analysis, develop and justify design decisions, realise a functional solution using appropriate technologies, and evaluate its effectiveness against agreed objectives. The unit emphasises informed decision-making, critical reflection and professional practice throughout the project process.

On successful completion of this unit, students will be able to produce and evaluate a complete applied computing solution that demonstrates technical competence, analytical thinking and professional judgement. They will be equipped to evidence skills in requirements analysis, solution design, implementation and evaluation that are directly relevant to industry. This unit supports progression into roles such as Software Developer, Systems Analyst, Junior Solutions Architect, Technical Consultant and Computing Project Officer, as well as progression to further study where independent project work is required.

***Please refer to the accompanying Pearson-set Assignment Guide and Theme and Topic Release document on HN Global for further support and guidance on the delivery of the Pearson-set unit.**

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Carry out a formal requirements analysis for a computing project to meet a set of defined objectives

LO2 Develop a design for a computational solution based on the identified requirements of a computing project

LO3 Implement a computing solution based on the design developed for the computing project

LO4 Review the effectiveness of the developed solution in meeting the identified requirements of the computing project.

## Essential Content

LO1 **Carry out a formal requirements analysis for a computing project to meet a set of defined objectives**

*Project context and scope:*

Context definition: problem domain e.g. business system, computing application; organisational context e.g. company environment, institutional setting; client context e.g. internal client, external customer; stakeholder context e.g. management staff, technical staff

Objectives and scope: project rationale e.g. problem resolution, service improvement; project objectives e.g. system goals, user outcomes; scope definition e.g. included features, excluded features; project boundaries e.g. system limits, responsibility limits

Constraints and measures: constraints e.g. time limits, budget limits; assumptions e.g. available resources, skill availability; success criteria e.g. requirement fulfilment, user acceptance.

*Stakeholder and user considerations:*

Stakeholder analysis: stakeholder identification e.g. users, sponsors; stakeholder categories e.g. internal stakeholders, external stakeholders; engagement boundaries e.g. consultation limits, decision authority

User analysis: user roles e.g. administrators, end users; user profiles e.g. technical users, non-technical users; user needs e.g. functionality needs, usability needs; user expectations e.g. performance expectations, reliability expectations

Compliance and communication: communication needs e.g. reporting needs, feedback needs; ethical considerations e.g. data privacy, user consent; accessibility considerations e.g. inclusive design, usability.

*Requirements classification:*

Requirement types: functional requirements e.g. data entry, reporting functions; non-functional requirements e.g. performance standards, reliability standards; usability requirements e.g. intuitive interfaces, clear navigation;

Quality constraints: performance requirements e.g. response times, processing speed; security requirements e.g. authentication controls, access restrictions; technical requirements e.g. platform compatibility, software dependencies

Compliance constraints: operational constraints e.g. staffing limits, operational hours; compliance requirements e.g. organisational policies, industry standards; legal requirements e.g. data protection law, licensing law; regulatory requirements e.g. sector regulations, compliance codes.

*Requirements prioritisation and traceability:*

Prioritisation and alignment: requirement prioritisation e.g. essential features, optional features; priority classification e.g. high priority, low priority; requirement alignment e.g. project objectives, business goals

Dependencies and conflicts: dependency identification e.g. system dependencies, process dependencies; conflict identification e.g. requirement clashes, resource conflicts; conflict resolution e.g. requirement negotiation, priority adjustment

Traceability and validation: traceability mapping e.g. requirement matrices, audit trails; validation methods e.g. stakeholder review, requirement walkthroughs; completeness checks e.g. requirement coverage, missing requirements; clarity checks e.g. clear wording, precise definitions; ambiguity resolution e.g. requirement refinement, clarification discussions.

LO2 **Develop a design for a computational solution based on the identified requirements of a computing project**

*Design principles and approaches:*

Modularity and abstraction: modularity e.g. independent components, system modules; abstraction e.g. simplified representations, hidden complexity

Separation of concerns and maintainability: separation of concerns e.g. logic separation, functional layers; maintainability e.g. readable code, ease of modification

Scalability and reusability: scalability e.g. increased users, data growth; reusability e.g. shared functions, reusable modules.

*System and solution architecture:*

Architectural models: architectural models e.g. layered architecture, client–server architecture

Component structures: component structures e.g. software components, subsystem modules

Data and process interactions: data and process interactions e.g. data exchange, process communication

Interface and deployment considerations: interface considerations e.g. user interfaces, system interfaces; deployment contexts e.g. local deployment.

*Data and process design:*

Data structures: data structures e.g. tables, records; data flow e.g. input flow, output flow; data integrity considerations e.g. validation rules, consistency rules;

Processing logic: processing logic e.g. algorithms, conditional logic; system workflows e.g. user journeys, process sequences

Ethical considerations: ethical design e.g. data collection, informed consent, data minimisation; data sensitivity e.g. personal data, special category data; data protection e.g. anonymisation, pseudonymisation; privacy preservation e.g. confidentiality, access control; secure data handling e.g. secure storage, data retention; transparency e.g. data usage, user awareness; fairness e.g. bias mitigation, equitable processing; compliance e.g. regulatory compliance, legal requirements.

*Design constraints and justification:*

Requirement-driven design decisions: requirement-driven design decisions e.g. functional needs, user needs

Technical constraints: technical constraints e.g. hardware limits, software limits

Platform considerations: platform considerations e.g. operating systems, hosting platforms

Risk mitigation through design: risk mitigation through design e.g. error handling, fallback mechanisms.

LO3 **Implement a computing solution based on the design developed for the computing project**

*Implementation environment:*

Platform configuration: development platforms e.g. desktop environment, cloud environment; configuration considerations e.g. environment settings, configuration files

Technology selection: programming technologies e.g. programming languages, scripting languages; frameworks e.g. development frameworks, application frameworks

Tools e.g. development tools, testing tools; debugging support e.g. error tracing, breakpoint analysis; version control e.g. source repositories, change tracking; configuration management e.g. build settings, environment management; documentation support e.g. code comments, technical notes.

*Solution construction:*

Component assembly: component integration e.g. module integration, library integration

Behaviour implementation: system behaviour implementation e.g. functional behaviour, system responses; logic realisation e.g. processing logic, decision logic

Interface implementation: interface realisation e.g. user interfaces, system interfaces, navigation structures, input forms, output layouts, accessibility features.

*Quality and correctness considerations:*

Design compliance: alignment with design specifications e.g. design compliance, specification matching; compliance with requirements e.g. functional compliance, non-functional compliance

System consistency: consistency between components e.g. shared standards, interface consistency; reliability considerations e.g. error handling, stable operation

Requirement alignment: functional and non-functional alignment e.g. feature alignment, performance alignment; requirement conformity e.g. requirement matching, specification fulfilment.

*Implementation challenges:*

Technical barriers: technical limitations e.g. hardware restrictions, software constraints; platform constraints e.g. operating system limits, deployment limits

Integration risks: integration issues e.g. component mismatches, data incompatibility

Resource limitations: resource considerations e.g. time availability, skill availability.

LO4 **Review the effectiveness of the developed solution in meeting the identified requirements of the computing project**

*Requirements fulfilment review:*

Testing: test planning e.g. test scope, test schedule; test execution e.g. test runs, defect logging; test types e.g. unit test, integration test, regression test, system test, user acceptance test, performance test, security test, functional test, structural test; test evidence e.g. screenshots, output logs; test records e.g. test cases, test reports

Requirement evaluation: requirement compliance e.g. specification matching, requirement satisfaction; functional coverage e.g. implemented features, supported functions; non-functional performance alignment e.g. response times, reliability levels

Constraint evaluation: constraint satisfaction e.g. budget limits, time limits; evaluation criteria e.g. success measures, acceptance criteria; review measures e.g. test results, evaluation records.

*Strengths and limitations:*

Technical strengths: technical effectiveness e.g. system stability, correct operation; design suitability e.g. fit for purpose, design effectiveness

System limitations: maintainability considerations e.g. ease of updating, code structure; scalability considerations e.g. handling growth, future expansion.

*Risk and improvement considerations:*

Risk identification: identified weaknesses e.g. performance issues, design gaps; technical risks e.g. security risks, reliability risks

Improvement planning: improvement opportunities e.g. optimisation options, design refinements; future enhancement potential e.g. feature expansion, system upgrades.

*Reflective evaluation:*

Fitness and alignment: solution fitness for purpose e.g. meeting objectives, user satisfaction; alignment with original objectives e.g. project goals, intended outcomes

Future implications: implications for future project work e.g. lessons learned, development improvements.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Carry out a formal requirements analysis for a computing project to meet a set of defined objectives | | |
| **P1** Carry out a formal requirements analysis for a computing project to meet a set of defined objectives.<br><br>**P2** Review, with a range of identified stakeholders, the formal requirements analysis for a computing project to identify strengths and areas for improvement. | **M1** Analyse, using the results of the stakeholder feedback, the effectiveness of the requirements analysis in meeting the user needs of the project. | **D1** Evaluate the effectiveness of the requirements analysis in supporting the development of an appropriate computing solution that meets user needs. |

| Pass | Merit | Distinction |
|---|---|---|
| **LO2** Develop a design for a computational solution based on the identified requirements of a computing project | | **LO2, LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of the computational solution in how comprehensively it addresses the identified requirements. |
| **P3** Develop a design for a computational solution that addresses the identified requirements of a computing project.<br>**P4** Review, with a group of internal stakeholders, the design of the computational solution based on the identified requirements of a computing project to identify strengths and areas for improvement. | **M2** Optimise, using the results of the stakeholder feedback, the design of the computational solution so that it thoroughly addresses the identified requirements. | |
| **LO3** Implement a computing solution based on the design developed for the computing project | | |
| **P5** Implement a computing solution based on the design developed for the computing project.<br>**P6** Review, with a group of internal stakeholders, the implemented solution in terms of quality and correctness compared with the design. | **M3** Optimise, using the results of the stakeholder feedback, the implementation of the computing solution so that it aligns with the design and the requirements. | |
| **LO4** Review the effectiveness of the developed solution in meeting the identified requirements of the computing project | | |
| **P7** Implement a test plan to determine how completely the implemented system meets the identified requirements.<br>**P8** Review, with a range of external stakeholders, the developed solution against the identified requirements of the computing project brief. | **M4** Analyse, using the results of testing and the stakeholder review, the strengths and limitations of the developed solution in meeting the identified requirements. | |

## Recommended Resources

### Textbooks

Ammann, P. and Offutt, J. (2024) *Introduction to Software Testing*. 3rd edn. Cambridge: Cambridge University Press.

Dennis, A., Wixom, B.H. and Roth, R.M. (2021) *Systems Analysis and Design*. 8th edn. Hoboken, NJ: Wiley.

Laplante, P.A. and Kassab, M. (2022) *Requirements Engineering for Software and Systems*. 4th edn. Boca Raton, FL: CRC Press.

Maylor, H. and Turner, N. (2021) *Project Management*. 5th edn. Harlow: Pearson Education.

Richards, M. and Ford, N. (2025) *Fundamentals of Software Architecture*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Wiegers, K. and Beatty, J. (2024) *Software Requirements*. 4th edn. Redmond, WA: Microsoft Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.projectmanager.com/blog/requirements-traceability-matrix | ProjectManager<br>Explains how to create and use a requirements traceability matrix to link stakeholder needs, requirements and deliverables.<br>(Article) |
| instituteprojectmanagement.com/blog/stakeholder-analysis-process/ | Institute of Project Management<br>Stakeholder Analysis Process: What is It?<br>(Article) |
| creately.com/guides/use-case-diagram-tutorial/ | Creately<br>UML use case diagram tutorial<br>(Tutorial) |
| dev.to/yasmine_ddec94f4d4/understanding-the-layered-architecture-pattern-a-comprehensive-guide-1e2j | DEV Community<br>Understanding the Layered Architecture Pattern: A Comprehensive Guide<br>(Article) |

| | |
|---|---|
| restfulapi.net/rest-api-design-tutorial-with-example/ | RESTfulAPI.net<br>How to Design a REST API<br>(Tutorial) |
| www.geeksforgeeks.org/software-testing/software-engineering-integration-testing/ | GeeksForGeeks<br>Integration Testing<br>(Article) |
| www.qasphere.com/blog/software-test-plan-guide/ | QA Sphere<br>Software Test Plan: Definition, Examples & Best Practices.<br>(General reference) |
| www.testrail.com/blog/create-a-test-plan/ | TestRail<br>Creating a Test Plan<br>(Tutorial) |
| www.atlassian.com/work-management/project-management/post-implementation-review | Atlassian<br>Post-implementation review: what it is & how it works<br>(Article) |
| www.projectmanager.com/blog/lessons-learned-project-management | ProjectManager<br>Lessons Learned in Project Management: A Quick Guide<br>(Article) |

### Journals and articles

Balleisen, E.J., Howes, L. and Wibbels, E. (2024) 'The impact of applied project-based learning on undergraduate student development', *Higher Education*, 87, pp. 1141–1156. Available at: doi.org/10.1007/s10734-023-01057-1.

Budake, R., Bhoite, S. and Kharade, K. (2023) 'Identification of functional and non-functional software requirements using machine learning', *AIP Conference Proceedings*, 2946(1). Available at: doi.org/10.1063/5.0178116.

Chronéer, D., Runardotter, M. and Johansson, J. (2024) 'Logics alignment in agile software design processes', *International Journal of Agile Systems and Management*, 17(6), pp. 1–31. Available at: doi.org/10.1504/IJASM.2024.142139.

Hoy, Z. and Xu, M. (2023) 'Agile Software Requirements Engineering Challenges-Solutions—A Conceptual Framework from Systematic Literature Review', *Information*, 14(6), 322. Available at: doi.org/10.3390/info14060322.

### Links

This unit links to the following related units:

*Unit 506: Advanced Software Design*

*Unit 510: Advanced Data Engineering*

*Unit 512: Advanced Data Analytics*

*Unit 515: Secure System Design*

*Unit 516: Internet of Things*

*Unit 520: Applied AI.*

 © Pearson Education Limited 2026

# Unit 503: Risk Assessment and Change Management

**Unit code:** R/652/3289

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Modern organisations operate in environments shaped by rapid technological change, increasing regulatory demands, and growing exposure to operational and security risks. As digital systems, data-driven processes, and interconnected services become central to organisational performance, the ability to identify risk and manage change in a controlled and structured way is essential. This unit is designed to support students in developing an understanding of how organisations remain resilient, compliant, and effective while responding to change.

The aim of this unit is to introduce students to the principles and practical application of formal risk assessment and structured change management within organisational contexts. Students will develop an understanding of how risks are identified, analysed, documented, and managed, alongside how planned and unplanned changes are assessed, approved, implemented, and reviewed. The unit emphasises the importance of governance, accountability, and communication in ensuring that change is introduced in a controlled and auditable manner.

The unit covers key concepts of organisational risk assessment, including risk terminology, evaluation methods, and alignment with recognised standards and frameworks. Students explore a range of change management models and procedures used by organisations, considering how risk-based decision-making supports effective change control. The unit then brings these strands together by requiring students to design an integrated risk assessment and change management process for an identified scenario, and to communicate that process clearly to a defined audience.

On successful completion of this unit, students will be able to design and justify structured approaches to managing risk and organisational change, and explain their value in real-world settings. These skills are relevant to a range of entry-level and junior professional roles, including IT Service Technician, Risk Analyst, Change Co-ordinator, Systems Administrator, and Governance or Compliance Officer, and also support progression to further study in areas such as cybersecurity, IT governance, and project management.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the key concepts of the formal risk assessment process within an organisation

LO2 Describe a range of change management models and procedures used by organisations

LO3 Design a structured risk assessment and change management process for an identified scenario

LO4 Deliver a presentation on the structured process to an audience of identified stakeholders.

## Essential Content

LO1 **Explain the key concepts of the formal risk assessment process within an organisation**

*Risk assessment principles:*

Core risk terms: risk e.g. service failure, data loss; threat e.g. cyberattack, power outage; vulnerability e.g. weak passwords, outdated software; asset e.g. servers, customer data

Risk measurement: likelihood e.g. chance of occurrence, frequency; impact e.g. financial loss, service disruption; consequence e.g. downtime, reputational damage

Risk states: inherent risk e.g. unmitigated risk, initial exposure; residual risk e.g. remaining risk, accepted exposure

Risk thresholds: risk appetite e.g. acceptable risk level, organisational limits; risk tolerance e.g. maximum impact, tolerance boundaries.

*Risk identification and categorisation:*

Risk origins: threat sources e.g. hackers, suppliers; source category e.g. internal source, external source

Organisational context: internal risks e.g. staff error, system misconfiguration; internal factor e.g. process failure, resource limitation; organisational risks e.g. poor processes, limited resources; organisational issue e.g. governance weakness, policy gap

External context: external risks e.g. market changes, new legislation; external driver e.g. economic change, regulatory change; environmental risks e.g. fire, flooding; environmental hazard e.g. natural disaster, physical event

Operational exposure: technical risks e.g. hardware failure, software bugs; human factors e.g. lack of training, user mistakes; compliance risks e.g. data breaches, regulatory non-compliance.

*Risk analysis and evaluation:*

Probability analysis: likelihood assessment e.g. low likelihood, high likelihood; rating scale e.g. qualitative scale, numerical scale

Impact analysis: impact assessment e.g. minor impact, major impact; impact scale e.g. severity level, impact rating

Assessment methods: qualitative approaches e.g. risk ranking, expert judgement; quantitative approaches e.g. scoring scales, numerical ratings

Decision tools: risk matrices e.g. heat maps, risk grids; visual tool e.g. impact-likelihood matrix, scoring grid; prioritisation e.g. high-risk focus, urgent risks; priority level e.g. critical priority, medium priority; risk scoring e.g. numerical scores, weighted scores; score type e.g. composite score, weighted index.

*Risk documentation and standards:*

Risk records: risk registers e.g. risk logs, tracking documents; risk statements e.g. cause-impact statements, structured summaries

Governance evidence: documentation requirements e.g. policies, procedures; audit evidence e.g. audit logs, review records, assurance reports

Standards alignment: compliance alignment e.g. legal requirements, organisational policy alignment; ISO 31000 e.g. risk principles, management framework; ISO 27005 e.g. information security risk guidance

Risk frameworks: National Institute of Standards and Technology (NIST) SP 800-30 e.g. risk assessment guidance, methodological structure; NIST SP 800-37 e.g. risk management life cycle, process stages, contemporary knowledge, updates, evolving practice.

LO2 **Describe a range of change management models and procedures used by organisations**

*Change management models:*

Change models: Awareness, Desire, Knowledge, Ability and Reinforcement (ADKAR) model e.g. awareness, communication, urgency; desire e.g. motivation, engagement; knowledge e.g. training, guidance; ability e.g. capability, performance; reinforcement e.g. monitoring, rewards; Kotter's eight-step model e.g. urgency, coalition, vision, communication, empowerment, wins, acceleration, embedding

Behavioural change models: Lewin's change model e.g. unfreeze, change, refreeze; behavioural reinforcement e.g. feedback, habit formation

Service management models: ITIL 4 change enablement e.g. standard change, change authority, normal change, emergency change; ISO/IEC 20000 service management e.g. service delivery, service level management; DevOps practices e.g. continuous integration, continuous delivery

ISO/IEC 38500 IT governance e.g. accountability, responsibility, strategy; The Open Group Architecture Framework (TOGAF) governance framework e.g. architecture governance, compliance reviews; IT4IT reference architecture e.g. value streams, service life cycle; NIST cybersecurity framework e.g. identify, protect, detect; Committee of Sponsoring Organizations of the Treadway Commission (COSO) enterprise risk management e.g. risk assessment, control environment.

*Organisational change types:*

Change categories: standard change e.g. low-risk updates; change type e.g. pre-approved change, repeatable change; minor change e.g. routine adjustments; change scope e.g. limited impact, small-scale change

Planned change: normal change e.g. assessed and scheduled change; change planning e.g. impact assessment, scheduling process; project-driven change e.g. system upgrades; project activity e.g. implementation phase, deployment stage

Urgent change: emergency change e.g. incidents, outages; incident response e.g. service recovery, fault resolution; business-critical change e.g. regulatory deadlines; priority driver e.g. compliance requirement, operational urgency

Risk classification: risk-based categorisation e.g. impact and likelihood; change risk tiers e.g. low, medium, high.

*Risk-aligned change management procedures:*

Change initiation and assessment: change request initiation e.g. business need; change logging e.g. Request for Comments (RFC) submission; impact assessment e.g. service, user, compliance impact; change risk assessment e.g. likelihood and impact scoring; third-party and supply chain risk assessment e.g. Managed Service Provider (MSP) dependencies, vendor access, software supply chain vulnerabilities

Change approval and implementation: authorisation procedure e.g. change advisory board (CAB) review, delegated authority; implementation procedure e.g. deployment, configuration updates

Change review and closure e.g. success evaluation, issue identification; closure activities e.g. formal sign-off, documentation update.

*Change governance and control:*

Change advisory board (CAB) e.g. cross-functional oversight; governance role e.g. approval authority, decision body

Requests for Change (RFC) e.g. formal change documentation; documentation type e.g. change request, approval record

Control mechanisms: approval workflows e.g. staged approval; workflow stage e.g. initial approval, final approval; control checkpoints e.g. mandatory reviews; control point e.g. review gate, approval checkpoint

Governance controls: Segregation of duties (SoD) e.g. approval independence, audit protection; control principle e.g. role separation, accountability separation.

*Stakeholders and communication:*

Stakeholder roles e.g. change owner, service owner; role type e.g. ownership role, operational role

Communication planning e.g. stakeholder updates, implementation notices

Accountability e.g. approval accountability, escalation responsibility

Oversight activities: implementation oversight e.g. progress monitoring; governance reporting e.g. status updates.

LO3 **Design a structured risk assessment and change management process for an identified scenario**

*Process structure and integration:*

Process design: end-to-end workflow design e.g. start-to-finish process, defined stages; process model e.g. sequential workflow, staged process

Process integration: integration of risk assessment with change control e.g. embedded risk checks, aligned approval stages; integration method e.g. control integration, workflow alignment

Organisational alignment: alignment with organisational policies e.g. internal standards, governance requirements, operating procedures; alignment type e.g. policy alignment, regulatory alignment

Context definition: scenario context and constraints e.g. business requirements, resource limits; context factor e.g. operational constraint, organisational requirement.

*Risk-based change control:*

Embedded risk checks in RFCs e.g. risk sections, impact fields; embedding method e.g. risk integration, control embedding.

*Impact analysis:*

Impact analysis e.g. service disruption, compliance impact, operational effect; impact category e.g. service impact, regulatory impact

Risk estimation: likelihood and impact scoring e.g. low-high scales, scoring matrices; estimation method e.g. scoring model, risk calculation

Decision priority: prioritisation mechanisms e.g. risk ranking, approval thresholds; priority method e.g. ranking model, threshold control.

*Controls and mitigation:*

Control types: preventive controls e.g. access controls, approval rules; control category e.g. security control, procedural control

Detection methods: detective controls e.g. monitoring tools, audit checks; detection mechanism e.g. system monitoring, audit detection

Response actions: corrective controls e.g. rollback plans, incident resolution; response activity e.g. recovery action, issue resolution

Control alignment: technical controls e.g. system configuration, security tools; technical measure e.g. configuration setting, security mechanism

Administrative controls e.g. policies, procedures; administrative measure e.g. policy control, procedural safeguard.

*Performance and improvement:*

Risk performance: key risk indicators (KRIs) e.g. incident frequency, risk trends

Process performance: key performance indicators (KPIs) e.g. change success rate, failure rate, rework levels

Post-change review: Post-implementation review (PIR) e.g. outcome evaluation, lessons learned

Continuous improvement: continual improvement e.g. process refinement, control enhancement, maturity improvement.

LO4 **Deliver a presentation on the structured process to an audience of identified stakeholders**

*Presentation structure and artefacts:*

Presentation artefacts: process diagrams e.g. workflow diagrams, flowcharts; diagram type e.g. process map, system diagram

Operational views: system workflows e.g. change life cycle views, approval flows, hand-off stages; workflow type e.g. life cycle model, approval chain

Risk artefacts: risk registers e.g. documented risks, ownership records, status tracking; record component e.g. risk entry, ownership field

Control artefacts: control mappings e.g. control-to-risk links, assurance mapping, accountability alignment; mapping type e.g. control matrix, linkage model.

*Evaluation and measurement:*

Effectiveness measures: effectiveness metrics e.g. success rates, failure reduction; metric type e.g. performance metric, outcome indicator

Assurance results: audit findings e.g. non-conformities, improvement actions, assurance outcomes; finding type e.g. compliance issue, audit observation

Compliance results: compliance outcomes e.g. regulatory alignment, policy adherence; compliance status e.g. compliant state, non-compliant state.

*Risk and control justification:*

Proposed controls e.g. least privilege, monitoring, backup and disaster recovery; control purpose e.g. risk reduction, impact limitation

Decision balance: cost–benefit considerations e.g. implementation cost, operational impact, risk reduction value; decision factor e.g. cost factor, benefit factor

Risk acceptance: residual risk acceptance e.g. documented sign-off, management approval; acceptance condition e.g. approval threshold, tolerance level.

*Continuous improvement and assurance:*

Assurance feedback: audit feedback e.g. recommendations, corrective actions; feedback type e.g. improvement suggestion, remediation action

Process improvement: optimisation opportunities e.g. efficiency gains, control refinement; improvement type e.g. process enhancement, control optimisation

Governance confirmation: governance review e.g. formal sign-off, assurance reporting; review outcome e.g. approval decision, compliance confirmation.

*Stakeholder considerations:*

Stakeholder types: technical stakeholders e.g. developers, system architects, network administrators; non-technical stakeholders e.g. clients, managers, analysts

Communication quality: communication effectiveness indicators e.g. clarity, audience understanding.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the key concepts of the formal risk assessment process within an organisation</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of formal risk assessment approaches and change management processes in supporting the implementation of change within an organisation.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the principles of risk assessment within an organisation.<br><br>P2 Discuss, using a selection of real-world examples, how risks are identified, analysed and documented as part of the formal risk assessment process within an organisation.</td><td>M1 Assess, using a selection of real-world examples, how well the formal risk assessment processes of an organisation complies with identified standards and frameworks.</td></tr>
<tr><td colspan="2">LO2 Describe a range of change management models and procedures used by organisations</td></tr>
<tr><td>P3 Describe, using a selection of real-world examples, the recognised change management models used by organisations.<br><br>P4 Explore, using a selection of real-world examples, the recognised change types and management procedures used by organisations, considering governance and stakeholders.</td><td>M2 Analyse, using a selection of real-world examples, the effectiveness of change management models and procedures used by organisations to manage change.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a structured risk assessment and change management process for an identified scenario | | **LO3 and LO4**<br>**D2** Justify, using outcomes from the review and stakeholder feedback, the decisions made in the design of the structured risk assessment and change management process to demonstrate how the process minimises organisational impact in the identified scenario. |
| **P5** Design a structured risk assessment and change management process for an identified scenario.<br>**P6** Review the designed structured risk assessment and change management process to inform improvements. | **M3** Optimise, using the results of the review, the structured risk assessment and change management process design to minimise the impact on the organisation in the identified scenario. | |
| **LO4** Deliver a presentation on the structured process to an audience of identified stakeholders | | |
| **P7** Design a professional presentation on the risk assessment and change management process to be delivered to an audience of identified stakeholders.<br>**P8** Deliver a presentation on the risk assessment and change management process to an audience of identified stakeholders to obtain relevant feedback. | **M4** Evaluate, using the results of the audience feedback, the effectiveness of the designed risk assessment and change management process to identify strengths and areas for improvement. | |

## Recommended Resources

### Textbooks

Blokdyk, G. (2021) ISO 31000 *risk management: a complete guide*. Brisbane: The Art of Service.

Field, A. (2023) *Risk Management and ISO 31000*: A Pocket Guide. Ely: IT Governance Publishing.

Fraser, J.R.S. and Simkins, B.J. (2024) *Enterprise Risk Management: Today's Leading Research and Best Practices for Tomorrow's Executives*. 2nd edn. Hoboken, NJ: Wiley.

Hopkin, P. (2024) *Fundamentals of Risk Management: Understanding, Evaluating and Implementing Effective Risk Management*. 7th edn. London: Kogan Page.

ISACA (2019) *COBIT® 2019 Framework: Governance and Management Objectives*. Schaumburg, IL: ISACA.

Leflar Jr, J.J. (2021) *Change management for risk professionals*. Boca Raton: CRC Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| blog.pacificcert.com/iso-31000-risk-management-framework-explained/ | Pacific Certifications<br>ISO 31000 Risk Management Framework Explained, including principles, process and organisational application<br>(Article) |
| www.securityscientist.net/blog/guide-to-nist-risk-assessments/ | Security Scientist<br>The Complete Guide to NIST Risk Assessments<br>(Blog) |
| itsm.tools/change-enablement/ | ITSM<br>Change Enablement in ITIL 4: Definition, Practice & Best Approaches<br>(Article) |
| www.si-labs.com/en/articles/kotters-8-step-model/ | Service Innovation Labs<br>Kotter's 8-Step Model: Guide, Critique and Practical Example for Change Processes<br>(Article) |

| | |
|---|---|
| riskpublishing.com/essential-risk-management-process-flow-chart/ | Risk Publishing<br>Essential Risk Management Process Flow Chart: A Step-by-Step Guide<br>(Article) |
| www.projectmanager.com/training/it-risk-management-strategies | Project Manager<br>IT Risk Management Process, Frameworks & Templates<br>(Resource) |
| sbnsoftware.com/blog/how-can-risk-assessment-results-be-effectively-communicated-to-stakeholders/ | SBN<br>How can risk assessment results be effectively communicated to stakeholders?<br>(Article) |
| www.atlassian.com/itsm/change-management/change-advisory-board | Atlassian<br>Why companies need a change advisory board<br>(Article) |

## Journals and articles

Basri, W.S. and Ayu, A.L. (2024) 'Risk Management in Information Systems: Applying ISO 31000:2018 and ISO/IEC 27001:2022 Controls at PMI's Central Clinic', *International Journal for Applied Information Management*, 4(1), pp. 1–13. Available at: doi.org/10.47738/ijaim.v4i1.70.

Kraugusteeliana, K. *et al*. (2024) 'The Application of COBIT Framework to Evaluate Information System Governance in National Business Technology Transformation Companies', *Jurnal Informasi dan Teknologi*, 6(1), pp. 86–92. Available at: doi.org/10.60083/jidt.v6i1.479.

Mangoki, W., Manongga, D. and Iriani, A. (2024) 'IT Governance Design in XY University using Cobit 2019 Framework', *Jurnal Sistem Informasi Bisnis*, 14(2), pp. 111–122. Available at: doi.org/10.21456/vol14iss2pp111-122.

Park, J.M. *et al*. (2024) 'Application of the ISO 31000-based Quality Risk Management Process for the ISO 17025-Accredited Testing and Analysis Industry: Toward Enhanced Confidence in Safety and Quality in the Environmental, Agricultural, and Food Industries', *Journal of Analytical Science and Engineering*, 36(3), pp. 254–277. Available at: doi.org/10.22698/jales.20240021.

## Links

This unit links to the following related units:

*Unit 504: Security Testing*

*Unit 505: Application Development.*

# Unit 504: Security Testing

**Unit code:** **A/652/3290**

**Unit level:** **5**

**Credit value:** **15**

---

## Introduction

As organisations continue to expand their reliance on digital systems, applications and interconnected networks, ensuring the security of these environments has become a core operational requirement. Security testing is a critical activity used to identify weaknesses before they can be exploited, helping organisations protect data, maintain service availability, and meet regulatory and contractual obligations. This unit explores the role of security testing within modern organisational contexts, recognising its importance across sectors such as finance, healthcare, cloud services and public infrastructure.

The aim of this unit is to develop students' understanding of how security testing is planned, carried out and evaluated in a structured and ethical manner. Students will gain insight into how testing activities support risk management and organisational decision-making, and how recognised frameworks, standards and professional practices are applied in real-world scenarios. The unit is designed to bridge the gap between theoretical security principles and their practical application within constrained, operational environments.

The unit covers the full security testing life cycle, beginning with an examination of security testing principles, models and frameworks used to guide testing activities. Students will then design a security testing plan for a given organisational scenario, considering scope, risk prioritisation, compliance requirements and attack surface mapping. Building on this, students will implement ethical penetration testing procedures using appropriate tools and techniques, before reviewing testing results to assess impact, prioritise vulnerabilities and propose suitable mitigation strategies.

On successful completion of this unit, students will be able to contribute effectively to security testing and vulnerability assessment activities within an organisation. They will be equipped to support tasks such as test planning, penetration testing, reporting and remediation review. These skills are relevant to roles including Security Analyst, Penetration Tester, Cyber Security Consultant, Network Security Engineer and Vulnerability Assessment Specialist, and also support progression to further study in cyber security and secure systems design.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore a range of security testing principles and frameworks within organisational contexts

LO2 Design an ethical security testing plan for a given organisational scenario

LO3 Implement an ethical security testing plan for a given organisational scenario

LO4 Review the results of the ethical security testing plan for a given organisational scenario.

## Essential Content

LO1 **Explore a range of security testing principles and frameworks within organisational contexts**

*Security testing principles:*

Core security objectives: confidentiality e.g. data protection, access control; integrity e.g. data accuracy, tamper prevention, validation controls; availability e.g. system uptime, redundancy

Defensive design concepts: defence in depth e.g. layered controls, network segmentation, monitoring; least privilege e.g. role assignment, access limitation

Authentication, authorisation and non-repudiation: authentication (MFA) mechanisms e.g. password authentication, multi-factor authentication, biometric authentication; identity verification e.g. user identity validation, credential verification; authorisation controls e.g. role-based access control, permission allocation, privilege management; access management e.g. access policies, permission structures; non-repudiation mechanisms e.g. digital signatures, audit logs, transaction records; accountability measures e.g. activity logging, action tracking.

*Security testing models and approaches:*

Risk-driven approaches: risk-based testing e.g. asset value, impact analysis; threat-led testing e.g. attack scenarios, adversary behaviour, threat intelligence

Intelligence and compliance approaches: intelligence-led testing e.g. threat feeds, incident trends, sector intelligence; compliance-driven testing e.g. regulatory checks, audit alignment

Testing visibility models: functional testing e.g. external perspective, perimeter focus; structural testing e.g. source access, configuration review; grey-box testing e.g. partial knowledge, credentialed access

Penetration testing and vulnerability scanning: penetration testing e.g. exploit simulation, attack execution, security validation; testing scope e.g. network testing, application testing, social engineering; testing methodology e.g. structured approach, testing phases, engagement rules; vulnerability scanning e.g. automated scanning, vulnerability identification, exposure detection; scan type e.g. internal scanning, external scanning; vulnerability management e.g. vulnerability prioritisation, remediation tracking, risk ranking.

*Security testing frameworks and standards:*

Framework classifications: industry frameworks e.g. best practice models, sector guidance; organisational security frameworks e.g. internal policies, control structures; regulatory and compliance frameworks e.g. legal requirements, audit standards

Strategic alignment: alignment between security objectives and testing activities e.g. business risk, security goals, testing scope

Framework selection criteria: framework selection based on organisational context e.g. system complexity, regulatory environment, risk appetite

Standards and guidance examples: ISO/IEC standards e.g. ISO 27001, ISO 27002, ISO 27005; NIST guidance e.g. National Institute of Standards and Technology (NIST) CSF, SP 800 series, risk frameworks; Open Worldwide Application Security Project (OWASP) models e.g. Top 10, testing guide, maturity models.

*Test planning concepts:*

Planning foundations: testing objectives e.g. risk discovery, assurance goals; scope definition e.g. in-scope assets, system boundaries, exclusions

Constraints and expectations: assumptions and constraints e.g. time limits, system availability; stakeholder requirements e.g. business priorities, reporting needs

Risk handling processes: risk identification e.g. threat sources, vulnerabilities; prioritisation strategies e.g. impact ranking, likelihood weighting.

LO2 **Design an ethical security testing plan for a given organisational scenario**

*Testing scope and objectives:*

Scope definition: system boundaries e.g. network segments, application layers, data stores; in-scope assets e.g. servers, endpoints, cloud services; out-of-scope exclusions e.g. third-party systems, live production constraints

Objective setting: business objectives e.g. risk reduction, service protection; security goals e.g. vulnerability discovery, control validation.

Organisational context: organisational environment e.g. business operations, service delivery, organisational structure; operational constraints e.g. time constraints, resource limitations; risk landscape e.g. threat exposure, risk profile, vulnerability levels; compliance context e.g. regulatory requirements, policy obligations; stakeholder context e.g. stakeholder roles, stakeholder expectations; organisational priorities e.g. business continuity, service availability.

*Methodologies and testing approaches:*

Testing methodologies: structured testing e.g. phased testing, repeatable processes; adaptive testing e.g. scenario-driven testing, threat-informed testing

Testing techniques: automated testing e.g. vulnerability scanning, configuration checks; manual testing e.g. exploratory testing, validation activities

Ethical and legal considerations: authorisation e.g. written approval, scope agreement; legal compliance e.g. data protection, contractual boundaries.

*Risk prioritisation and compliance alignment:*

Risk assessment methods: impact evaluation e.g. business impact, service disruption; likelihood evaluation e.g. exploitability, exposure

Compliance requirements: regulatory requirements e.g. data protection laws, sector regulations and updates; organisational compliance e.g. internal policies, governance standards.

*Attack surface mapping and test structuring:*

Attack surface mapping: asset identification e.g. hosts, applications, interfaces; entry points e.g. network services, user inputs; trust boundaries e.g. network zones, authentication transitions

Test structuring: test suites e.g. grouped test cases, functional areas; execution sequencing e.g. logical order, dependency awareness.

*Documentation and reporting structures:*

Test documentation: test plans e.g. scope statements, objectives; evidence collection e.g. screenshots, logs, outputs

Reporting structures: reporting formats e.g. technical reports, executive summaries; stakeholder communication e.g. briefings, findings presentations.

LO3 **Implement an ethical security testing plan for a given organisational scenario**

*Ethical and professional testing practice:*

Authorisation: authorisation e.g. written approval, scope confirmation; rules of engagement e.g. testing boundaries, permitted techniques, escalation routes; legal boundaries e.g. contractual limits, data protection, jurisdictional constraints

Professional conduct: professional conduct e.g. ethical behaviour, accountability; confidentiality e.g. data handling, access control, information protection; responsible disclosure e.g. reporting timelines, stakeholder notification, remediation co-ordination.

*Penetration testing process stages:*

Pre-exploitation activities: reconnaissance e.g. information gathering, footprinting; enumeration e.g. service discovery, account identification

Exploitation activities: vulnerability identification e.g. weakness discovery, misconfiguration detection, exposure analysis; exploitation techniques e.g. exploit execution, proof-of-concept development, controlled access

Post-exploitation activities: post-exploitation activities e.g. impact confirmation, access validation; validation of findings e.g. result verification, false positive removal.

*Security testing tools and techniques:*

Assessment tools: vulnerability scanners e.g. automated scanning, configuration analysis; fuzzing tools e.g. input mutation, error detection, robustness testing

Analysis tools: static analysis tools e.g. source review, code inspection; dynamic analysis tools e.g. runtime testing, behaviour monitoring

Test environment definition: environment configuration e.g. test environments, staging environments, isolated environments; system setup e.g. system configurations, network configurations; environment isolation e.g. sandbox environments, virtual environments; operational safety e.g. production protection, service stability; data management e.g. test data sets, anonymised data; data control e.g. data separation, data security

Execution support tools: exploitation frameworks e.g. exploit libraries, payload delivery; automation tools e.g. scripting, repeatable testing.

*Information gathering and exploitation concepts:*

Information sources: open-source intelligence e.g. public records, exposed services, metadata analysis

Privilege escalation concepts: privilege escalation concepts e.g. permission abuse, credential reuse, misconfiguration exploitation

Movement techniques: lateral movement awareness e.g. network traversal, trust abuse.

*Testing documentation and artefact management:*

Evidence management: activity logs e.g. command history, timestamps; screenshots e.g. proof capture, validation evidence

Result handling: output validation e.g. result checking, consistency review; test result consolidation e.g. findings aggregation, report preparation.

LO4 **Review the results of the ethical security testing plan for a given organisational scenario**

*Analysis of testing results:*

Vulnerability identification: identified vulnerabilities e.g. software flaws, misconfigurations; false positives e.g. tool noise, non-exploitable findings

Validation outcomes: validation outcomes e.g. exploit confirmation, evidence verification, repeat testing

Impact understanding: exploitation impact e.g. privilege gain, data exposure; business and technical consequences e.g. service disruption, compliance risk.

*Impact assessment and risk evaluation:*

Risk evaluation factors: risk severity e.g. high impact, low tolerance; asset impact e.g. critical systems, sensitive data

Likelihood assessment: likelihood assessment e.g. ease of exploitation, exposure level, threat presence

Scoring and interpretation: vulnerability scoring models e.g. CVSS base scores, contextual scoring.

*Remediation and mitigation strategies:*

Remediation actions: patching approaches e.g. vendor updates, version upgrades; configuration changes e.g. hardening settings, access restrictions

Risk handling options: compensating controls e.g. monitoring controls, network segmentation; risk acceptance e.g. documented acceptance, business sign-off

Mitigation prioritisation: mitigation prioritisation e.g. risk ranking, resource availability.

*Review and improvement considerations:*

Evaluation outcomes: lessons learned e.g. process gaps, control weaknesses; testing effectiveness e.g. coverage adequacy, tool suitability

Future planning: residual risk e.g. remaining exposure, accepted risk; recommendations for future testing cycles e.g. scope adjustment, frequency changes

Posture development: security posture improvement e.g. maturity improvement, control enhancement.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explore a range of security testing principles and frameworks within organisational contexts | | |
| **P1** Explore, using a selection of real-world examples, a range of security testing principles and models used to assess system vulnerabilities and risks.<br><br>**P2** Review, using a selection of real-world examples, a range of security testing frameworks and test planning concepts used to investigate system vulnerabilities and risks. | **M1** Analyse, using a selection of real-world examples, the effectiveness of security testing principles, models and frameworks used to assess risk within different system contexts. | **D1** Critically evaluate, using a selection of real-world examples, the effectiveness of security testing principles, models and frameworks in addressing organisational security risks. |

| Pass | Merit | Distinction |
|---|---|---|
| **LO2** Design an ethical security testing plan for a given organisational scenario | | **LO2, LO3 and LO4**<br>**D2** Justify the decisions made in the design of the ethical security testing plan in determining how effectively it identified security vulnerabilities in the given organisational scenario. |
| **P3** Design and document an ethical security testing plan that defines the scope, objectives, attack surface mapping and testing approaches for a given organisational scenario.<br>**P4** Review the design of the ethical security testing plan with a range of identified key stakeholders to obtain feedback to identify and inform improvements. | **M2** Assess, using the results of the review and stakeholder feedback, the design of the ethical security testing plan in relation to risk prioritisation and compliance requirements. | |
| **LO3** Implement an ethical security testing plan for a given organisational scenario | | |
| **P5** Implement an ethical security testing plan using appropriate tools and techniques for a given organisational scenario.<br>**P6** Produce testing evidence and documentation that record the ethical activities performed, results gathered and information obtained. | **M3** Evaluate the effectiveness of the ethical security testing plan in relation to the identification and validation of security weaknesses. | |
| **LO4** Review the results of the ethical security testing plan for a given organisational scenario | | |
| **P7** Review the results of the ethical security testing plan for a given organisational scenario to produce a risk assessment and impact evaluation.<br>**P8** Develop, using the results of the risk assessment and impact evaluation, a set of mitigation strategies to improve the security of the given organisational scenario and to inform future improvements. | **M4** Analyse the effectiveness of the mitigation strategies in being able to secure the given organisational scenario against identified vulnerabilities. | |

## Recommended Resources

### Textbooks

Alsmadi, I. (2023) *The NICE Cyber Security Framework: Cyber Security Intelligence and Analytics*. Cham: Springer.

Andress, J. (2024) *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice*. 4th edn. Amsterdam: Elsevier.

Bhardwaj, R. (2025) *Vulnerability Assessment and Penetration Testing (VAPT)*. New Delhi: BPB Publications.

McNab, C. (2023) *Network Security Assessment: Know Your Network*. 4th edn. Sebastopol: O'Reilly Media.

Skoudis, E. and Liston, T. (2024) *Penetration Testing: A Hands-On Introduction to Hacking*. 3rd edn. Hoboken: Wiley.

### Websites

Website links were valid at the time of initial publication.

| | |
|---|---|
| hightable.io/iso27001-annex-a-8-29-security-testing-in-development-and-acceptance/ | High Table<br>ISO 27001 Annex A 8.29 Explained<br>(Article) |
| tkxel.com/blog/practical-guide-to-risk-based-cyber-security-testing/ | TKXEL<br>A Practical Guide to Risk-Based Cyber Security Testing<br>(Article) |
| secportal.io/blog/penetration-testing-methodology-guide | SecPortal<br>Penetration Testing Methodology: Steps, Frameworks & Complete Guide<br>(Blog) |
| www.on-sitetechnology.com/penetration-testing-methodology-guide/ | On-Site Technology<br>Penetration Testing Methodology: A Clear Step-by-Step Guide<br>(Article) |
| www.inspectiv.com/articles/vulnerability-remediation-how-to-prioritize-and-fix-what-matters | Inspectiv<br>Vulnerability Remediation: How to Prioritize and Fix What Matters<br>(Blog) |

| | |
|---|---|
| learn.microsoft.com/en-us/azure/well-architected/security/test | Microsoft Learn<br>Architecture strategies for security testing<br>(Resource) |
| www.sentinelone.com/cybersecurity-101/cybersecurity/attack-surface-mapping/ | SentinelOne<br>What is an Attack Surface Mapping?<br>(Article) |
| www.cyberly.org/en/what-are-the-legal-and-ethical-considerations-in-penetration-testing/index.html | Cyberly<br>Legal And Ethical Considerations In Penetration Testing<br>(Blog) |

**Journals and articles**

Alhamed, M. and Rahman, M.M.H. (2023) 'A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions', *Applied Sciences*, 13(12), 6986. Available at: doi.org/10.3390/app13126986.

Al Shebli, H.M.Z. and Beheshti, B.D. (2018) 'A study on penetration testing process and tools', *in 2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pp. 1–7. Available at: doi.org/10.1109/LISAT.2018.8378035.

Balsam, A. *et al*. (2025) 'Automatic CVSS-Based Vulnerability Prioritization and Response with Context Information and Machine Learning', *Applied Sciences*, 15(16), 8787. Available at: doi.org/10.3390/app15168787.

Vallabhaneni, R. and Veeramachaneni, V. (2024) 'Understanding Penetration Testing for Evaluating Vulnerabilities and Enhancing Cyber Security', *Engineering and Technology Journal*, 9(10), pp. 5335–5337. Available at: doi.org/10.47191/etj/v9i10.12.

**Links**

This unit links to the following related units:

*Unit 503: Risk Assessment and Change Management*

*Unit 505: Application Development.*

# Unit 505: Application Development

**Unit code:** D/652/3291

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Application development plays a central role in how organisations operate, innovate and deliver digital services in a rapidly evolving technology landscape. From internal business systems to customer-facing platforms, well-designed software applications support efficiency, scalability and informed decision-making across a wide range of sectors. This unit is designed to reflect contemporary industry practice, where technical capability must be combined with an understanding of business needs, quality standards and professional communication.

The aim of this unit is to develop students' ability to evaluate, design, build and present a functional software application for an identified business scenario. Students will be supported in making informed technical decisions, selecting appropriate tools and development approaches, and applying structured methods throughout the application life cycle. The unit emphasises the importance of aligning technical solutions with organisational requirements, quality expectations and stakeholder needs.

The unit covers key stages of the application development process, beginning with a review of tools, platforms and techniques used to support application creation and life cycle management. Students then progress to designing a software solution that addresses both functional and non-functional requirements within a defined business context. This is followed by the development and testing of a functional application that meets quality and acceptance criteria. The unit concludes with the presentation of the developed application to an intended audience, incorporating evaluation and feedback to support refinement and improvement.

On successful completion of this unit, students will have developed practical and transferable skills in software design, development, evaluation and professional presentation. These skills are directly relevant to a range of computing roles, including Application Developer, Software Engineer, Systems Analyst, Junior DevOps Engineer and Associate Technical Consultant. The unit also provides a strong foundation for progression to further Level 6 study or specialist units within software engineering and digital technologies.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Review the tools and techniques used in the creation of a business application for an identified business scenario

LO2 Design a software application for an identified business scenario

LO3 Develop a functional business application for an identified business scenario

LO4 Present the functional business application developed for an identified business scenario to an intended audience.

## Essential Content

LO1 **Review the tools and techniques used in the creation of a business application for an identified business scenario**

*Application development platforms and environments:*

Development environments: integrated development environments e.g. debugging tools, code editors; cloud-based development environments e.g. remote workspaces, hosted runtimes; examples: Visual Studio, Eclipse, AWS Cloud9, GitHub Codespaces

Application platforms: low-code platforms e.g. workflow builders, logic designers, rule engines; enterprise application platforms e.g. application servers, middleware services; examples: Microsoft Power Apps, Mendix, Salesforce Platform, ServiceNow

Platform evaluation: scalability e.g. load handling, resource scaling; maintainability e.g. modular structure, documentation quality, code updates; licensing models e.g. subscription licensing, enterprise agreements; examples: Microsoft Azure, Amazon Web Services, Google Cloud Platform.

*AI development tools:*

Model resources: pre-trained models e.g. classification models, language models; model repositories e.g. shared registries, versioned stores, model libraries; examples: Hugging Face Hub, TensorFlow Hub, Azure AI Studio, Amazon SageMaker

Prompt workflows: prompt engineering tools e.g. prompt editors, template managers; prompt testing tools e.g. response validation, scenario testing, output comparison; examples: OpenAI Playground, Azure OpenAI Studio, LangChain, PromptLayer

Evaluation tools: model evaluation tools e.g. accuracy metrics, recall analysis; model monitoring tools e.g. performance tracking, drift detection, usage monitoring; examples: MLflow, Azure Machine Learning, Amazon SageMaker Model Monitor.

*Application life cycle tools – requirements management and analysis:*

Elicitation methods: requirements elicitation techniques e.g. interviews, workshops, observation

Requirement definition: user stories e.g. acceptance criteria, outcome statements; use cases e.g. system interactions, scenario descriptions, actor flows; examples: Jira Software, Azure DevOps, Trello, Rally

Control mechanisms: requirements traceability e.g. requirement mapping, impact analysis; documentation standards e.g. specification templates, formatting guidelines, version conventions.

*Application Life cycle Tools – development, testing, and quality tools:*

Development assets: programming frameworks e.g. application frameworks, development stacks; libraries e.g. reusable components, shared modules, utility packages; examples: Spring Boot, .NET, React, Angular

Source control: code repositories e.g. central repositories, distributed repositories; version control systems e.g. branching strategies, merge control, commit history; examples: GitHub, GitLab, Bitbucket, Azure Repos

Quality assurance: automated testing tools e.g. test runners, regression tools; static code analysis tools e.g. code scanning, complexity analysis, rule checking.

LO2 **Design a software application for an identified business scenario**

*Functional requirements:*

Core functionality: business processes e.g. order handling, approval workflows; user interactions e.g. form submission, navigation flows; examples: Salesforce CRM, ServiceNow, Microsoft Dynamics 365

Logic and rules: system behaviours e.g. rule execution, event handling; validation rules e.g. input constraints, format checking; workflow logic e.g. conditional flows, task sequencing; examples: SAP S/4HANA, Oracle NetSuite, Workday.

*Non-functional requirements:*

Performance and reliability: performance expectations e.g. response times, throughput targets; availability e.g. uptime targets, service continuity; examples: Amazon Web Services, Microsoft Azure, Google Cloud Platform

Security and usability: security requirements e.g. access control, encryption; usability considerations e.g. accessibility features, user experience; compliance requirements e.g. data protection standards, regulatory alignment; examples: Microsoft Entra ID, Okta, Open Worldwide Application Security Project (OWASP) Application Projects.

*Application architecture and structure:*

Architectural design: architectural styles e.g. layered architecture, microservices; application layering models e.g. presentation layer, business logic layer; examples: Netflix Platform Architecture, Amazon Microservices Architecture, Spotify Backstage

Integration design: integration points e.g. application programming interfaces (APIs), messaging services; deployment architecture e.g. cloud deployment, container orchestration; examples: Kubernetes, Docker, Azure App Service.

*Design and modelling techniques:*

Design approaches: architectural patterns e.g. Model–View–Controller (MVC), event-driven architecture; component-based design e.g. modular components, reusable services; examples: Spring Framework, .NET Architecture Patterns, Apache Kafka

Interface design: interface design tools e.g. wireframing tools, prototyping tools; design standards e.g. UI guidelines, coding conventions; accessibility considerations e.g. Web Content Accessibility Guidelines (WCAG compliance), inclusive design; examples: Figma,
Adobe XD, Material Design.

*Data and information design:*

Data structures: data entities e.g. customer records, transaction records; relationships e.g. entity associations, data hierarchies; examples: Unified Modelling Language (UML) Tools, Draw.io.

*Operational considerations:*

Operational readiness: risk mitigation measures e.g. threat modelling, vulnerability management; support and maintenance considerations e.g. incident management, update planning; examples: Azure Monitor, AWS CloudWatch, ServiceNow.

LO3 **Develop a functional business application for an identified business scenario**

*Application development life cycle:*

Requirements implementation e.g. requirement breakdown, task definition

Design realisation e.g. component mapping, interface alignment

Build execution e.g. feature construction, integration sequencing

Testing execution e.g. verification cycles, defect resolution;

Deployment preparation e.g. release packaging, environment readiness

Operational transition e.g. handover activities, support enablement.

*Implementation and build considerations:*

Implementation approaches: programming languages e.g. object-oriented languages, scripting languages; frameworks e.g. application frameworks, development stacks; examples: Java with Spring Boot, C# with .NET, JavaScript with Node.js

Build structure: modular implementation e.g. reusable modules, component separation; integration of components e.g. service connections, interface binding; configuration management e.g. environment settings, parameter control; examples: Maven, Gradle, Docker, Kubernetes.

*Testing and validation approaches:*

Testing methods: functional testing e.g. feature verification, workflow testing; non-functional testing e.g. performance testing, security testing; examples: Selenium, JMeter, OWASP Zed Attack Proxy (ZAP), Postman

Validation control: defect management e.g. issue tracking, fault resolution; acceptance criteria alignment e.g. requirement validation, stakeholder sign-off; examples: Jira, Azure DevOps, TestRail, GitHub Issues.

*Quality standards and compliance:*

Quality frameworks: coding standards e.g. style guidelines, naming conventions; documentation standards e.g. technical documentation, user guides; examples: Google Style Guides, Microsoft Coding Standards, Confluence, SharePoint

Compliance requirements: testing standards e.g. test documentation, evidence records; maintainability criteria e.g. code readability, modular design.

*Deployment and operational readiness:*

Deployment preparation: deployment environments e.g. development environments, production environments; release management e.g. version releases, deployment scheduling

Operational support: operational documentation e.g. runbooks, support guides; maintenance planning e.g. update schedules, patch management.

LO4 **Present the functional business application developed for an identified business scenario to an intended audience**

*Presentation structure and communication:*

Audience focus: audience analysis e.g. stakeholder roles, user needs; presentation structure e.g. logical sequencing, clear sections; examples: Microsoft PowerPoint, Google Slides, Prezi

Communication clarity: clarity of explanation e.g. structured narratives, key messaging; professional terminology e.g. technical vocabulary, business language; examples: Microsoft Teams, Zoom, Webex.

*Demonstration and explanation of functionality:*

Application walkthrough: application features e.g. core functions, user actions; workflow demonstration e.g. process flows, task execution

Design justification: system behaviour explanation e.g. process outcomes, rule execution; design rationale e.g. design decisions, technology choices.

*Feedback and evaluation considerations:*

Feedback collection: audience feedback types e.g. verbal feedback, written feedback; usability observations e.g. navigation issues, interaction patterns; examples: Microsoft Forms, Google Forms, Miro

Performance review: performance observations e.g. response behaviour, execution speed; improvement identification e.g. enhancement areas, defect recognition.

*Optimisation and refinement considerations:*

Application improvement: application improvements e.g. feature refinement, functional enhancements; usability enhancements e.g. interface adjustments, accessibility fixes; examples: Jira, Azure DevOps, GitHub Issues

Performance optimisation: performance refinement e.g. efficiency improvements, response tuning; reliability considerations e.g. stability measures, error reduction.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Review the tools and techniques used in the creation of a business application for an identified business scenario</td><td rowspan="2"></td></tr>
<tr><td>P1 Review, using a selection of real-world examples, the development platforms, environments and AI tools used to develop business applications.<br><br>P2 Explore, using a selection of real-world examples, the tools used to help support and manage the application life cycle of business applications.</td><td>M1 Analyse, using a selection of real-world examples, the suitability of selected platforms, environments and tools in creating a business application for an identified business scenario.</td><td>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of identified platforms, environments and tools in creating a business application for an identified business scenario that comprehensively meets requirements.</td></tr>
<tr><td colspan="2">LO2 Design a software application for an identified business scenario</td><td rowspan="4">LO2 and LO3<br><br>D2 Justify, using the results of the stakeholder feedback and application testing, the decisions made in the design and development of the application for meeting functional and non-functional requirements for an identified business scenario.</td></tr>
<tr><td>P3 Design a software application and test plan that addresses functional and non-functional requirements for an identified business scenario.<br><br>P4 Review the designs and test plan with a range of key stakeholders to identify strengths and areas for improvement in addressing functional and non-functional requirements for an identified business scenario.</td><td>M2 Optimise, using the results of the stakeholder feedback, the software designs and test plan so that they comprehensively address functional and non-functional requirements for an identified business scenario.</td></tr>
<tr><td colspan="2">LO3 Develop a functional business application for an identified business scenario</td></tr>
<tr><td>P5 Develop a functional business application that satisfies defined requirements and quality standards for an identified business scenario.<br><br>P6 Implement the test plan to identify strengths and areas for improvement in the quality standards and acceptance criteria of the developed business application.</td><td>M3 Evaluate, using the results of the test plan, the effectiveness of the developed business application in comprehensively meeting functional, non-functional and quality requirements for an identified business scenario.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Present the functional business application developed for an identified business scenario to an intended audience | | |
| **P7** Deliver a presentation on the functional business application developed for an identified business scenario to an intended audience that demonstrates professionalism.<br><br>**P8** Review feedback from the intended audience to identify strengths and areas for improvement in the presented application. | **M4** Optimise the presentation using feedback from the intended audience to identify strengths and areas for improvement. | **D3** Evaluate the professionalism and effectiveness of the presentation made in response to audience feedback. |

## Recommended Resources

### Textbooks

Beyer, B. and Lhotka, L. (2024) Software Architecture: *The Hard Parts*. Sebastopol: O'Reilly Media.

Cervantes, H. and Kazman, R. (2024) *Designing Software Architectures: A Practical Approach*. 2nd edn. Boston, MA: Addison-Wesley Professional.

Hughes, B., Cotterell, M. and Mall, R. (2023) *Software Project Management*. 7th edn. London: McGraw-Hill Education.

Humble, J. and Farley, D. (2010) *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Boston, MA: Addison-Wesley.

Richards, M. and Ford, N. (2025) *Fundamentals of Software Architecture*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Sommerville, I. and Tilley, S. (2024) *Software Engineering*. 11th edn. Harlow: Pearson.

Stellman, A. and Greene, J. (2023) *Learning Agile*. Sebastopol: O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| github.com/resources/articles/what-is-version-control | GitHub<br>What is Version Control?<br>(Article) |
| www.jetbrains.com/guide/tags/debugging/ | JetBrains<br>Debugging – Investigate errors and trace code execution with a visual debugger<br>(Resource) |
| www.nngroup.com/articles/ten-usability-heuristics/ | NN/G<br>10 Usability Heuristics for User Interface Design<br>(Article) |
| www.lucidchart.com/blog/how-to-draw-architectural-diagrams | LucidChart<br>5 types of architecture diagrams<br>(Article) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.mindtools.com/c51fpv6/presentation-skills/ | MindTools<br>Presentation Skills<br>(Resource Hub) |
| www.geeksforgeeks.org/blogs/low-code-developement-best-practices/ | Geeks For Geeks<br>Best Practices to Low Code Development<br>(Article) |
| www.geeksforgeeks.org/blogs/top-10-no-code-development-platforms/ | Geeks For Geeks<br>Top 10 No-Code Development Platforms<br>(Article) |
| www.kuika.com/en/blog/what-is-visual-programming-everything-you-need-to-know-to-start | Kuika<br>Beginner's Guide to Visual Programming<br>(Article) |
| www.knack.com/blog/exploring-no-code-development-platforms/ | Knack<br>Exploring No-Code Development Platforms<br>(Article) |

**Journals and articles**

Azad, N. and Hyrynsalmi, S. (2024) 'Multivocal Literature Review on DevOps Critical Success Factors', *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024)*, pp. 1–8. Available at: doi.org/10.1145/3661167.3661236.

Ebert, C. and Hochstein, L. (2023) 'DevOps in Practice', *IEEE Software*, 40(1), pp. 29–36. Available at: doi.org/10.1109/MS.2022.3213285.

Razavian, M., Paech, B. and Tang, A. (2023) 'The vision of on-demand architectural knowledge systems as a decision-making companion', *Journal of Systems and Software*, 198, Article 111560. Available at: doi.org/10.1016/j.jss.2022.111560.

Sedano, T., Ralph, P. and Péraire, C. (2020) 'Dual-Track Development', *IEEE Software*, 37(6), pp. 58–64. Available at: doi.org/10.1109/MS.2020.3013274.

**Links**

This unit links to the following related units:

*Unit 503: Risk Assessment and Change Management*

*Unit 504: Security Testing.*

# Unit 506: Advanced Software Design

**Unit code:** F/652/3292

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Advanced software design plays a critical role across a wide range of modern computing sectors, including enterprise computing, cloud services and data-driven applications. As software systems grow in scale and sophistication, there is increasing demand for practitioners who can apply structured design approaches to create robust, maintainable and scalable solutions. This unit focuses on the principles and practices required to design effective software systems in a professional context.

The aim of this unit is to develop students' ability to design, analyse and justify advanced software solutions. Building on the skills and techniques studied at Level 4, students are required to engage with more complex design problems, apply recognised design patterns and algorithms, and make informed technical decisions in response to defined scenarios. The unit supports the development of higher-level design capability aligned with current industry practice.

The unit explores a range of topics central to the design and implementation of advanced software systems. Students will examine established software design patterns and principles to understand how reusable solutions address recurring design problems. They will analyse advanced algorithms and consider their suitability for specific computational contexts, including performance and complexity considerations. The unit culminates in the design and development of a programmed solution using appropriate modelling and documentation techniques, followed by the evaluation of the implemented solution against defined requirements.

On successful completion of this unit, students will have the knowledge and skills needed to design, analyse and evaluate advanced software solutions in realistic contexts. They will be equipped to make justified design decisions and reflect critically on solution quality and effectiveness. These capabilities support progression into roles such as Software Developer, Software Engineer, Systems Designer, Application Architect or Technical Analyst, as well as further study in software engineering or computing-related disciplines.

 © Pearson Education Limited 2026

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore a range of design patterns used in advanced software design

LO2 Analyse a range of advanced algorithms appropriate to specific computational problems

LO3 Design a programmed solution to meet a defined scenario

LO4 Develop a programmed solution that satisfies the scenario requirements.

## Essential Content

LO1 **Explore a range of design patterns used in advanced software design**

*Design pattern categories:*

Classification types: creational patterns e.g. object creation control, life cycle management; structural patterns e.g. class composition, system organisation; behavioural patterns e.g. interaction handling, responsibility flow

Conceptual purpose: intent and purpose e.g. design intent clarity, solution motivation; pattern classification e.g. pattern grouping, category identification; pattern selection contexts e.g. problem type matching, design constraints.

*Creational patterns:*

Instantiation control: object creation strategies e.g. instance reuse, controlled allocation; abstraction of instantiation e.g. interface-based creation, decoupled construction; life cycle control e.g. instance lifetime, reuse management

Creational examples: singleton e.g. shared instance access, global state control; factory method e.g. object creation delegation, subclass instantiation; abstract factory e.g. related object families, framework extension.

*Structural patterns:*

Structure organisation: class and object composition e.g. object grouping, structure reuse; interface adaptation e.g. interface compatibility, legacy integration; system modularity e.g. component separation, structural flexibility

Structural examples: adapter e.g. interface translation, system compatibility; decorator e.g. feature extension, responsibility layering; facade e.g. simplified access, subsystem co-ordination.

*Behavioural patterns:*

Interaction focus: object interaction e.g. message passing, collaboration control; responsibility distribution e.g. task allocation, role separation; communication flow e.g. request handling, control sequencing

Behavioural examples: observer e.g. event notification, state updates; strategy e.g. algorithm selection, behaviour switching; command e.g. request encapsulation, action management.

*Design principles:*

Design quality principles: SOLID principles e.g. responsibility separation, dependency control; separation of concerns e.g. functionality isolation, responsibility focus; coupling and cohesion e.g. dependency reduction, module clarity

Design styles: RESTful design e.g. resource orientation, stateless communication; extensibility e.g. feature addition, system expansion; maintainability e.g. code clarity, modification ease.

LO2 **Analyse a range of advanced algorithms appropriate to specific computational problems**

*Algorithmic problem domains:*

Problem types: optimisation problems e.g. resource usage, solution efficiency, cost reduction; search problems e.g. solution space, state exploration, path discovery; scheduling problems e.g. task ordering, time allocation, priority management; distributed computing problems e.g. workload distribution, co-ordination needs, node interaction

Domain characteristics: constraint handling e.g. rule limits, boundary conditions, feasibility checking; problem scale e.g. small datasets, large datasets, growth factors; solution goals e.g. efficiency targets, accuracy goals, completion criteria.

*Advanced algorithm types:*

Algorithm classes: dynamic programming e.g. subproblem reuse, optimal substructure, overlapping problems; heuristic search algorithms e.g. guided exploration, solution approximation, rule-based guidance; distributed algorithms e.g. parallel co-ordination, message exchange, process synchronisation; approximation algorithms e.g. near-optimal solutions, reduced computation, acceptable accuracy

Algorithm strategies: recursive approaches e.g. repeated breakdown, structured solving; iterative approaches e.g. stepwise refinement, OOP processing; greedy methods e.g. local choice, incremental building.

*Computational complexity:*

Performance measures: time complexity e.g. execution steps, input growth, processing effort; space complexity e.g. memory usage, storage needs, data allocation; Big O Notation e.g. upper bounds, growth comparison, performance classification; algorithm scalability e.g. performance increase, data volume, system expansion

Complexity factors: best-case behaviour e.g. minimal workload, favourable input; worst-case behaviour e.g. maximum workload, unfavourable input; average-case behaviour e.g. typical data, expected performance.

*Algorithm suitability factors:*

Selection considerations: input size considerations e.g. data volume, dataset range, growth rate; resource constraints e.g. memory limits, processing capacity, system availability; accuracy requirements e.g. exactness needs, acceptable error, result tolerance; execution environments e.g. hardware platforms, system contexts, deployment settings

Decision influences: performance trade-offs e.g. speed versus accuracy, memory versus time; implementation effort e.g. development complexity, maintainability needs; reliability expectations e.g. consistent output, predictable behaviour.

LO3 **Design a programmed solution to meet a defined scenario**

*Solution design approaches:*

Design structuring: problem decomposition e.g. requirement analysis, functional prioritisation, constraint resolution; modular design e.g. component responsibility, interface boundaries, change isolation; abstraction e.g. conceptual modelling, dependency hiding, design generalisation; architectural layering e.g. separation enforcement, responsibility distribution, cross-layer interaction

Design planning: workflow design e.g. process sequencing, control logic, data transformation; responsibility allocation e.g. role justification, dependency ownership, accountability clarity; dependency management e.g. coupling mitigation, dependency direction, impact containment.

*Design modelling techniques:*

Model representations: Unified Modelling Language (UML) diagrams e.g. design abstraction, system reasoning, structural validation; class diagrams e.g. cohesion analysis, responsibility allocation, reuse potential; sequence diagrams e.g. behavioural correctness, interaction dependencies, execution logic; component diagrams e.g. architectural coherence, dependency analysis, deployment readiness

Modelling purposes: design validation e.g. consistency evaluation, requirement traceability, design risk identification; stakeholder communication e.g. design justification, decision explanation, shared understanding.

*Data and interface design:*

Data structuring: data modelling e.g. entity justification, relationship rationale, integrity control; data structures e.g. efficiency consideration, access optimisation, structural suitability; data handling e.g. life cycle management, transformation logic, consistency assurance

Interface definition: Application Programming Interface (API) design e.g. contract clarity, abstraction enforcement, change tolerance; interface contracts e.g. precondition definition

*Design documentation:*

Design records: design specifications e.g. requirement linkage, solution scope, design constraints; assumptions and constraints e.g. design limitations, environmental conditions, operational boundaries; design rationale e.g. decision justification, alternative evaluation, trade-off analysis

Scenario alignment: alignment to scenario requirements e.g. requirement mapping, design traceability, coverage validation; design consistency e.g. logical coherence, requirement satisfaction, solution integrity.

LO4 **Develop a programmed solution that satisfies the scenario requirements**

*Program implementation concepts:*

Implementation structure: programming constructs e.g. logical control, design alignment, implementation consistency; control structures e.g. flow enforcement, decision handling, execution control; modular code organisation e.g. component separation, responsibility isolation, maintainable structure; integration of components e.g. interface linkage, dependency resolution, system coherence

Implementation decisions: implementation strategies e.g. incremental development, structured implementation, design adherence; code organisation e.g. logical grouping, responsibility clarity, reuse management.

*Development quality considerations:*

Code characteristics: code readability e.g. naming clarity, structural transparency, comprehension ease; maintainability e.g. change tolerance, defect localisation, evolution support; adherence to design specifications e.g. design conformity, requirement satisfaction, traceability enforcement; consistency with standards e.g. coding conventions, quality benchmarks, organisational guidelines

Quality management: technical debt awareness e.g. implementation shortcuts, maintenance impact, refactoring need; implementation discipline e.g. rule compliance, quality assurance, implementation control.

*Verification and validation aspects:*

Verification focus: functional correctness e.g. requirement fulfilment, output accuracy, behaviour alignment; logical accuracy e.g. condition handling, processing validity, rule consistency; alignment with scenario requirements e.g. requirement coverage, scenario traceability, implementation justification

Validation activities: test consideration e.g. input handling, boundary checking, scenario coverage; defect identification e.g. fault isolation, error detection, correction rationale.

*Evaluation considerations:*

Solution assessment: solution effectiveness e.g. requirement achievement, performance acceptability, functional completeness; limitations e.g. design constraints, implementation boundaries, scalability issues; scope for enhancement e.g. optimisation opportunities, feature extension, future improvement

Reflective judgement: reflection against original design e.g. design adherence, decision review, outcome evaluation; improvement justification e.g. change reasoning, enhancement prioritisation, evaluative judgement.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explore a range of design patterns used in advanced software design</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate, using a selection of practical programming examples, the effectiveness of selected design patterns and advanced algorithms in addressing advanced software design requirements and solving computational problems.</td></tr>
<tr><td>P1 Explore, using a selection of practical programming examples, the design patterns used in advanced software design.<br><br>P2 Explain, using a selection of practical programming examples, the design principles used in advanced software design.</td><td>M1 Analyse, using a selection of practical programming examples, the suitability of different design patterns and principles in advanced software design contexts.</td></tr>
<tr><td colspan="2">LO2 Analyse a range of advanced algorithms appropriate to specific computational problems</td></tr>
<tr><td>P3 Analyse, using a selection of practical programming examples, the advanced algorithms appropriate to specific computational problems.<br><br>P4 Discuss, using a selection of practical programming examples, the suitability and complexity of advanced algorithms appropriate to specific computational problems.</td><td>M2 Analyse, using a selection of practical programming examples, the impact of algorithm selection on software performance.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a programmed solution to meet a defined scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of the programmed solution in the context of meeting the requirements of the identified scenario. |
| **P5** Design a programmed solution that meets the requirements of a defined scenario.<br>**P6** Review the solution design with a range of identified stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the review, the design of the programmed solution so that it comprehensively addresses the requirements of the identified scenario. | |
| **LO4** Develop a programmed solution that satisfies the scenario requirements | | |
| **P7** Develop a programmed solution that satisfies the scenario requirements.<br>**P8** Implement a test plan to thoroughly test whether the programmed solution satisfies the solution requirements. | **M4** Evaluate, using the results of the testing, the effectiveness of the programmed solution in meeting the requirements of the identified scenario. | |

## Recommended Resources

### Textbooks


Buschmann, F., Henney, K. and Schmidt, D.C. (2021) *Pattern-Oriented Software Architecture Volume 5: On Patterns and Pattern Languages*. Chichester: Wiley.

Cervantes, H. and Kazman, R. (2024) *Designing Software Architectures: A Practical Approach*. 2nd edn. Boston, MA: Addison-Wesley Professional.

Freeman, E. and Robson, E. (2021) *Head First Design Patterns*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Gamma, E. *et al*. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA: Addison-Wesley.

Martin, R.C. (2021) *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Boston, MA: Pearson.

Richards, M. *et al*. (2022) *Software Architecture: The Hard Parts*. Sebastopol, CA: O'Reilly Media.

Uzayr, S.B. (2022) *Software Design Patterns: The Ultimate Guide*. Boca Raton, FL: CRC Press.


### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| refactoring.guru/design-patterns/catalog | Refactoring Guru<br>The Catalog of Design Patterns<br>(General Reference) |
| www.geeksforgeeks.org/system-design/software-design-patterns/ | GeeksforGeeks<br>Software Design Patterns<br>(Tutorial) |
| www.digitalocean.com/community/tutorials/gangs-of-four-gof-design-patterns | DigitalOcean<br>Gang of 4 Design Patterns Explained: Creational, Structural, and Behavioral<br>(General reference) |
| www.geeksforgeeks.org/dsa/analysis-algorithms-big-o-analysis/ | GeeksforGeeks<br>Algorithm complexity fundamentals explained using Big O notation<br>(Tutorial) |

| | |
|---|---|
| unwiredlearning.com/blog/big-o-notation-guide | Unwired Learning<br>Beginner guide to Big-O Notation<br>(Article) |
| www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/ | Visual Paradigm<br>UML Class Diagram Tutorial<br>(Tutorial) |
| www.uml-diagrams.org/index-examples.html | UML-Diagrams.org<br>Worked UML software examples<br>(General reference) |

**Journals and articles**

Koç, H. *et al*. (2021) 'UML Diagrams in Software Engineering Research: A Systematic Literature Review', *Proceedings*, 74(1), Article 13. Available at: doi.org/10.3390/proceedings2021074013.

Kuroiwa, R. and Beck, J.C. (2023) 'Solving Domain-Independent Dynamic Programming Problems with Anytime Heuristic Search', *Proceedings of the International Conference on Automated Planning and Scheduling*, 33(1), pp. 245–253. Available at: doi.org/10.1609/icaps.v33i1.27201.

Ramachandrappa, N.C. (2024) 'SOLID Design Principles in Software Engineering', *International Journal of Computer Trends and Technology*, 72(9), pp. 18–23. Available at: doi.org/10.14445/22312803/IJCTT-V72I9P104.

Syed-Mohamad, S.M. *et al*. (2025) 'Measuring Software Maintainability: An Exploration of Metrics and Continuous Practices', *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 63(2), pp. 181–195. Available at: doi.org/10.37934/araset.63.2.181195.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*

*Unit 507: Full Stack Development*

*Unit 508: Discrete Mathematics.*

# Unit 507: Full Stack Development

**Unit code:** H/652/3293

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Software applications underpin many of the services, platforms and systems organisations rely on to operate effectively in a digital economy. Modern applications are rarely built as isolated components; instead, they combine user-facing interfaces, application logic, data management and deployment infrastructure into integrated systems. Full stack development reflects this industry reality by focusing on the design and delivery of complete software solutions that are scalable, secure and maintainable across their entire life cycle.

The aim of this unit is to develop students' understanding of how front-end and backend technologies work together to support the delivery of modern enterprise applications. The unit emphasises architectural awareness and system integration rather than isolated programming techniques, enabling students to appreciate how design and technology choices influence performance, reliability and long-term sustainability.

The unit covers key concepts and practices used in full stack development. Students will explore common full stack architectures and the responsibilities of different system layers before developing a robust front-end user interface using contemporary frameworks. They will then implement backend services to support application logic, data management and security, culminating in the deployment of a scalable full stack solution within a configured environment. Together, these topics provide students with a coherent understanding of end-to-end application development.

On successful completion of this unit, students will be able to design, develop and deploy full stack applications that meet identified requirements and reflect current industry practice. The knowledge and skills developed support progression into roles such as Full Stack Developer, Software Engineer, Web Application Developer, Junior DevOps Engineer, and Application Support Engineer, as well as further study in software engineering, cloud computing and DevOps-related units.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore the range of full stack architectures used to deliver modern enterprise applications

LO2 Develop a robust front-end user interface using contemporary frameworks for an identified scenario

LO3 Implement robust backend services for an identified scenario

LO4 Deploy a scalable full stack solution for an identified scenario.

## Essential Content

LO1 **Explore the range of full stack architectures used to deliver modern enterprise applications**

*Full stack architecture models:*

Architectural styles: monolithic architectures e.g. single deployment units, shared codebases; layered architectures e.g. presentation separation, logical tiering; microservices architectures e.g. independently deployable services, decentralised components, service isolation

Service and execution models: service-oriented architectures e.g. reusable business services, standardised interfaces; event-driven architectures e.g. event producers, event consumers, message brokers; serverless architectures e.g. function execution, managed runtimes.

*System layers and responsibilities:*

Core system layers: front-end presentation layer e.g. user interaction logic, client rendering; backend application layer e.g. business logic processing, service co-ordination, request handling; data persistence layer e.g. structured data storage, transaction management

Design structure principles: separation of concerns e.g. responsibility isolation, logical separation; coupling and cohesion e.g. dependency control, modular organisation, internal consistency.

*Front-end and backend integration:*

Communication mechanisms: API-driven communication e.g. service endpoints, Hypertext Transfer Protocol (HTTP) interfaces; RESTful services e.g. resource representations, stateless interactions, standard methods

Interaction and data handling: Synchronous communication e.g. blocking requests, immediate responses; asynchronous communication e.g. background processing, message queues; data exchange formats e.g. JavaScript Object Notation (JSON) structures, serialised data.

*Scalability and enterprise considerations:*

Scalability strategies: horizontal scaling e.g. service replication, workload distribution; vertical scaling e.g. resource augmentation, capacity upgrades, performance enhancement

Enterprise quality attributes: stateless services e.g. request independence, session externalisation; fault tolerance e.g. failure mitigation, service recovery, redundancy mechanisms; maintainability e.g. code evolution, architectural flexibility.

LO2 **Develop a robust front-end user interface using contemporary frameworks for an identified scenario**

*Front-end frameworks and approaches:*

Framework architecture: single-page application frameworks e.g. client routing, dynamic views; component-based architectural approaches e.g. reusable components, interface modularity, state isolation

Rendering and design patterns: client–side rendering e.g. browser execution, dynamic updates, runtime rendering; declarative user interface (UI) patterns e.g. UI state descriptions, predictable rendering.

*User interface structure and components:*

Component organisation: reusable components e.g. shared UI elements, consistent interfaces; component hierarchies e.g. parent–child structure, nested components, interface composition

Interface behaviour: state management approaches e.g. local state, shared state; event handling e.g. user interactions, system events, input responses.

*Responsive and accessible design:*

Layout adaptability: responsive layouts e.g. flexible grids, adaptive containers; device adaptability e.g. screen variation, resolution support, orientation changes

Accessibility and usability: accessibility standards e.g. Web Content Accessibility Guidelines (WCAG) principles, accessibility compliance; semantic markup e.g. meaningful elements, structured content.

*Performance and optimisation:*

Performance techniques: client–side performance optimisation e.g. efficient rendering, reduced computation; code splitting e.g. modular bundles, selective loading, reduced payloads

Resource efficiency: lazy loading e.g. deferred resources, on-demand loading; bundle optimisation e.g. asset minimisation, dependency reduction.

LO3 **Implement robust backend services for an identified scenario**

*Backend architecture and services:*

Service structure: server-side application structure e.g. layered services, modular organisation; service layers e.g. business logic separation, processing tiers, service abstraction; middleware components e.g. request processing, service co-ordination

Request management: API services e.g. exposed endpoints, service interfaces, service contracts; routing mechanisms e.g. request mapping, URL patterns, endpoint resolution; application controllers e.g. request handling, control flow.

*Data management and persistence:*

Data organisation: database types e.g. relational databases, non-relational databases; data models e.g. entity structures, schema definitions, data relationships; schema design e.g. table structures, relationship mapping

Data operations: Create, Read, Update, and Delete (CRUD) operations e.g. data creation, data modification, data retrieval; data validation e.g. input validation, constraint checks, data rules; data integrity e.g. consistency enforcement, referential integrity.

*Authentication and security:*

Access control: authentication mechanisms e.g. credential verification, identity validation; authorisation models e.g. role-based access, permission control, access policies

Data protection: secure data access e.g. protected endpoints, controlled permissions; session management e.g. user sessions, session life cycles, session storage; token-based authentication e.g. access tokens, refresh tokens.

*Integration and reliability:*

System integration: API integration e.g. internal services, external services; service endpoints e.g. controlled access points, defined interfaces, request handlers

Service robustness: error handling e.g. exception management, failure responses; logging e.g. activity records, error logs, audit trails; retry mechanisms e.g. repeated requests, transient failure recovery.

LO4 **Deploy a scalable full stack solution for an identified scenario**

*Deployment environments and configuration:*

Environment types: development environments e.g. local testing, feature development; testing environments e.g. system testing, user testing; production environments e.g. live deployment, end-user access; environment isolation e.g. separated environments, deployment boundaries

Configuration management: environment variables e.g. runtime settings, configuration values; configuration management e.g. centralised configuration, versioned settings; dependency management e.g. library control, version dependencies.

*Deployment approaches and pipelines:*

Deployment processes: continuous integration e.g. automated builds, integration testing; continuous deployment e.g. automated releases, deployment automation; build processes e.g. compilation steps, artefact generation

Pipeline automation: automated pipelines e.g. workflow automation, deployment stages; version-control integration e.g. source repositories, change tracking; release management e.g. version releases, change co-ordination.

*Hosting and infrastructure:*

Infrastructure models: cloud hosting models e.g. public cloud, private cloud; application hosting platforms e.g. managed platforms, runtime environments; virtualisation e.g. virtual machines, abstracted hardware

Scalability support: containerisation concepts e.g. container images, isolated runtimes; infrastructure provisioning e.g. resource creation, environment provisioning; resource allocation e.g. compute resources, memory distribution.

*Operational scalability and maintenance:*

Operational monitoring: monitoring and logging e.g. system logs, operational metrics; performance monitoring e.g. response times, throughput measurement; system health checks e.g. service status, availability checks

Service maintenance: update strategies e.g. version updates, rolling updates; fault management e.g. issue resolution, failure handling; incident management e.g. incident response, escalation processes; operational documentation e.g. system records, operational guides.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explore the range of full stack architectures used to deliver modern enterprise applications | | |
| **P1** Explore, using a selection of real-world examples, the range of full stack architectures used in modern enterprise applications, considering the roles and responsibilities of the different layers.<br><br>**P2** Review, using a selection of real-world examples, the interaction and integration between front-end and backend layers of a full stack architecture and their impact on scalable enterprise applications. | **M1** Analyse, using a selection of real-world examples, the benefits and limitations of different full stack architectures in supporting the development of scalable and maintainable enterprise applications. | **D1** Critically evaluate, using a selection of real-world examples, the effectiveness of different full stack architectures in supporting enterprise application performance, scalability and long-term sustainability. |

| Pass | Merit | Distinction |
|---|---|---|
| **LO2** Develop a robust front-end user interface using contemporary frameworks for an identified scenario | | **LO2, LO3 and LO4**<br>**D2** Justify the decisions made in the design, development and deployment of a full stack solution in how comprehensively it meets the requirements of an identified scenario. |
| **P3** Implement a usable and accessible front-end user interface design using a contemporary framework that meets the requirements of an identified scenario.<br>**P4** Review the developed front-end user interface with a range of identified stakeholders to identify strengths and areas to improve in terms of accessibility, usability and performance. | **M2** Optimise, using the results of the stakeholder review, the developed front end user interface to improve accessibility, usability and performance. | |
| **LO3** Implement robust backend services for an identified scenario | | |
| **P5** Implement a backend services design that supports data handling, application logic and integration requirements for an identified scenario.<br>**P6** Review the developed back-end services with a range of identified stakeholders to identify strengths and areas to improve in terms of security, integration and reliability. | **M3** Optimise, using the results of the stakeholder review, the developed back-end services to improve security, integration and reliability. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Deploy a scalable full stack solution for an identified scenario | | **LO2, LO3 and LO4** *(continued)* |
| **P7** Integrate the developed front-end and back-end services to create a scalable full stack solution for an identified scenario.<br><br>**P8** Deploy the developed full stack solution that operates correctly within a configured deployment environment for an identified scenario.<br><br>**P9** Implement a developed test plan to determine whether the deployed full stack solution meets the requirements of an identified scenario. | **M4** Evaluate the deployed full stack solution to determine the effectiveness in meeting the requirements of the identified scenario. | |

## Recommended Resources

### Textbooks

Gorton, I. (2021) *Foundations of Scalable Systems: Designing Distributed Architectures*. 1st edn. Boston, MA: Addison-Wesley.

Grinberg, M. (2023) *Designing Web APIs*. Sebastopol: O'Reilly Media.

Kim, G. *et al*. (2021) *The DevOps Handbook*. 2nd edn. Portland: IT Revolution Press.

Knell, A. and Yadav, A. (2024) *Full Stack Development with Angular and Spring Boot*. 3rd edn. Birmingham: Packt Publishing.

Newman, S. (2024) *Building Microservices*. 2nd edn. Sebastopol: O'Reilly Media.

Richards, M. and Ford, N. (2025) *Fundamentals of Software Architecture*. 2nd edn. Sebastopol, CA: O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures | Microsoft Learn<br>Overview of common web application architectures including monolithic, layered and service-based models used in modern full stack systems.<br>(General reference) |
| www.geeksforgeeks.org/software-engineering/monolithic-vs-microservices-architecture/ | GeeksforGeeks<br>Comparative explanation of monolithic and microservices architectures.<br>(Tutorial/general reference) |
| docsallover.com/blog/ui-ux/building-a-spa-with-a-frontend-framework/ | DocsAllOver<br>Step-by-step explanation of single-page application development.<br>(Tutorial) |
| www.w3.org/WAI/tips/designing/ | W3C Web Accessibility Initiative Practical guidance on designing accessible and responsive user interfaces aligned with WCAG accessibility principles.<br>(General reference) |

 © Pearson Education Limited 2026

| | |
|---|---|
| dev.to/yuktisays/understanding-backend-architecture-a-complete-guide-to-request-life cycle-55gp | DEV Community<br>Walkthrough of backend request handling, routing, middleware and service layering in server-side applications.<br>(Tutorial) |
| www.w3schools.com/nodejs/nodejs_middleware.asp | W3Schools<br>Introduction to Node.js middleware concepts.<br>(Tutorial) |
| www.okta.com/identity-101/what-is-token-based-authentication/ | Okta<br>Explanation of token-based authentication, access tokens and secure API access.<br>(General reference) |
| www.deployhq.com/blog/ci-cd-pipelines-complete-guide | DeployHQ<br>Overview of CI/CD pipelines.<br>(General reference) |
| aws.amazon.com/what-is/containerization/ | Amazon Web Services<br>Explanation of containerisation concepts, benefits and scalability support for deploying full stack applications<br>(General reference) |
| www.braincuber.com/tutorial/deploy-full-stack-app-guide-beginner | Braincuber<br>Beginner-friendly guide to deploying full stack applications, covering environments, configuration and deployment strategies.<br>(Tutorial) |

## Journals and articles

Bucko, A. *et al*. (2023) 'Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History', *Computers*, 12(4), 78. Available at: doi.org/10.3390/computers12040078.

Dakkak, A. and Bosch, J. (2024) 'Towards Continuous Deployment at Scale in Software-Intensive Embedded Systems: A Maturity Model from the Telecommunications Domain', in *Proceedings of COMPSAC 2024*, pp. 1256–1267. Available at: doi.org/10.1109/COMPSAC61105.2024.00166.

Velepucha, V. and Flores, P. (2023) 'A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges', *IEEE Access*, 11, pp. 88339–88358. Available at: doi.org/10.1109/ACCESS.2023.3305687.

Vepsäläinen, J., Hellas, A. and Vuorimaa, P. (2025) 'Overview of Web Application Performance Optimization Techniques', in *WEBIST 2023 Revised Selected Papers*. Cham: Springer, pp. 45–62. Available at: doi.org/10.1007/978-3-031-89621-7_3.

## Links

This unit links to the following related units:

*Unit 506: Advanced Software Design.*

*Unit 508: Discrete Mathematics*

*Unit 542: API Design and Development.*

# Unit 508: Discrete Mathematics

**Unit code:** J/652/3294

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Discrete mathematics forms a core part of modern computing, providing the structures and techniques used to describe, organise and analyse non-continuous systems. It underpins many aspects of digital technologies, including data representation, algorithm design, system modelling and logical decision-making. Unlike continuous mathematics, discrete mathematics focuses on finite or countable structures, making it particularly suited to computational problems where precision, structure and formal reasoning are required.

The aim of this unit is to develop students' understanding of key discrete mathematical concepts and how they are applied within computing contexts. Students will build confidence in working with formal mathematical structures and in interpreting abstract concepts in a way that supports analysis and problem solving. The unit is designed to strengthen analytical thinking and provide a mathematical foundation for further study and practice within computing and digital technology disciplines.

This unit covers a range of discrete mathematics topics aligned to the learning outcomes. Students will explore set theory to formalise data structures and relationships, alongside functions, relations and mappings used to represent associations within discrete systems. Graph theory is examined as a method for modelling networks, dependencies and pathways, while combinatorics and discrete probability are used to analyse possible outcomes, decision paths and the implications of computational complexity.

On successful completion of this unit, students will be able to apply discrete mathematical techniques to model and analyse computational problems, evaluate different representations and support structured decision-making. These skills are relevant to a range of technical roles, including Software Developer, Data Analyst, Systems Analyst, Cybersecurity Analyst, Network Engineer and Algorithm Engineer, and support progression to higher-level study in computing-related subjects.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Apply set theory concepts relevant to discrete computational structures

LO2 Apply the mathematical concepts of functions, relations, and mappings to discrete mathematical contexts and structures

LO3 Apply the key concepts and principles of graph theory relevant to problems in computing

LO4 Apply the theory of combinatorics and discrete probability to computational decision-making.

## Essential Content

LO1 **Apply set theory concepts relevant to discrete computational structures**

*Set theory fundamentals:*

Membership concepts: elements e.g. objects, values; membership e.g. element inclusion, set membership; subsets e.g. proper subsets, improper subsets

Notation and size measures: cardinality e.g. finite size, infinite size; empty set e.g. ∅; universal set e.g. defined universe of discourse

Classification and representation: power sets e.g. sets of subsets; classification e.g. set categories, set types; representation e.g. symbolic notation, set-builder notation.

*Set operations and representations:*

Operation types: union e.g. combined sets; intersection e.g. common elements; difference e.g. relative complement; complement e.g. set complement with respect to a universal set

Expressions and abstractions: operations e.g. formal set operations; expressions e.g. algebraic set expressions; abstraction e.g. generalised set representation

Diagrammatic modelling methods: diagrams e.g. Venn diagrams; modelling e.g. abstract set models; relationships e.g. set relationships, inclusion relationships.

*Relations and partitions:*

Relation definition concepts: pairs e.g. ordered pairs; binary relations e.g. relations on sets; equivalence e.g. equivalence relations

Equivalence structure components: equivalence e.g. equivalence properties; classes e.g. equivalence classes; partitions e.g. partitions of a set

Grouping and organisation structures: grouping e.g. element grouping; classification e.g. relational classification; structure e.g. partitioned set structures.

*Applications to computing:*

Access control problems: role sets e.g. administrator roles, user roles, guest roles; permission sets e.g. read access, write access, execute access; inclusion e.g. role inheritance, permission inheritance, privilege escalation

Data validation and constraint checking: valid value sets e.g. permitted input ranges, accepted data types, mandatory fields; invalid value sets e.g. null values, out-of-range inputs, prohibited characters; constraints e.g. domain restrictions, format rules, boundary conditions

Database querying problems: record sets e.g. query result sets, filtered tables, materialised views; intersections e.g. multi-condition queries, join conditions, overlapping criteria; differences e.g. excluded records, filtered exclusions, omitted results

State-space problems: state sets e.g. operational states, error states, idle states; configuration sets e.g. permitted configurations, default configurations, custom configurations; exclusions e.g. invalid combinations, conflicting states, forbidden transitions.

LO2 **Apply the mathematical concepts of functions, relations, and mappings to discrete mathematical contexts and structures**

*Functions and mappings:*

Definition components: domain e.g. set of inputs; codomain e.g. target set; range e.g. image of a function

Function types: functions e.g. defined rules between sets; mappings e.g. assignments from elements of one set to another

Function properties: injective functions e.g. one-to-one mappings; surjective functions e.g. onto mappings; bijective functions e.g. one-to-one correspondences.

*Function composition and inversion:*

Function operations: composition e.g. composition of functions; inversion e.g. inverse functions

Dependency relationships: dependency e.g. functional dependency between sets; structure e.g. ordered functional structures

Input and output flow: inputs e.g. elements of the domain; outputs e.g. corresponding elements of the codomain; transformation e.g. set-to-set mapping.

*Relations in discrete structures:*

Relation characteristics: relations e.g. defined subsets of Cartesian products; pairs e.g. ordered pairs; binary relations e.g. relations on a single set

Relation properties: reflexive relations e.g. self-related elements; symmetric relations e.g. mutual relations; transitive relations e.g. chained relations

Structural consistency: equivalence relations e.g. relations satisfying reflexive, symmetric and transitive properties; structure e.g. relational structures.

*Applications to computing:*

Database design applications: functional dependencies e.g. normalisation constraints, key dependencies, attribute dependencies; relations e.g. table relationships, join relations, referential links; mappings e.g. primary keys, foreign keys, reference mappings

Software logic applications: input functions e.g. method inputs, function parameters, API inputs; output functions e.g. return values, response objects, output streams; composition e.g. chained operations, composed functions, nested calls

Data integration applications: mappings e.g. schema mappings, data mappings, field mappings; correspondences e.g. field alignment, attribute alignment, schema alignment; transformations e.g. data conversion

System interaction applications: relational models e.g. component relationships, service relationships, interface relationships; mappings e.g. interaction pathways, message routing, event mappings; structures e.g. modelled system behaviour.

LO3 **Apply the key concepts and principles of graph theory relevant to problems in computing**

*Graph theory foundations:*

Graph components: graphs e.g. abstract graphs; vertices e.g. nodes, points; edges e.g. connections, links

Graph types: directed graphs e.g. directed edges, arcs; undirected graphs e.g. symmetric connections; weighted graphs e.g. graphs with edge weights

Structural representation: structures e.g. graph structures; representations e.g. adjacency matrices, adjacency lists; models e.g. abstract graph models

*Graph structures:*

Tree forms: trees e.g. connected acyclic graphs; rooted trees e.g. trees with a distinguished root; spanning trees e.g. minimal connected subgraphs

Organisational patterns: hierarchies e.g. hierarchical graph structures; networks e.g. connected graph systems; topology e.g. graph topology

Relationship modelling: relationships e.g. adjacency relationships; structures e.g. connected structures, subgraphs.

*Connectivity and traversal:*

Path analysis: paths e.g. simple paths; cycles e.g. cyclic paths; connectivity e.g. connected and disconnected graphs

Traversal methods: traversal e.g. graph traversal; depth-first search e.g. depth-first traversal; breadth-first search e.g. breadth-first traversal

Reachability assessment: reachability e.g. reachable vertices; navigation e.g. movement within a graph.

*Applications to computing:*

Network routing problems: nodes e.g. routers, switches, endpoints; edges e.g. communication links, transmission paths, connections; weights e.g. latency values, bandwidth costs, hop counts

Dependency resolution problems: dependency graphs e.g. package dependencies, module dependencies, service dependencies; ordering e.g. execution order, build order, deployment order; cycles e.g. circular dependencies, dependency loops, invalid chains

Resource allocation problems: graph models e.g. allocation graphs, flow graphs, matching graphs; constraints e.g. capacity limits, availability limits, scheduling limits; conflicts e.g. contention points, resource clashes, deadlocks

Path optimisation problems: paths e.g. shortest paths, optimal paths, alternative paths; criteria e.g. minimum cost, minimum time, minimum distance; exclusions e.g. blocked routes, failed nodes, restricted edges.

LO4 **Apply the theory of combinatorics and discrete probability to computational decision-making**

*Combinatorial principles:*

Counting methods: permutations e.g. ordered arrangements; combinations e.g. unordered selections; counting principles e.g. sum rule, product rule

Discrete selection structures: selections e.g. element selection from sets; arrangements e.g. ordered arrangements of elements; outcomes e.g. possible results

Enumeration approaches: possibilities e.g. finite cases; grouping e.g. combinatorial groupings.

*Discrete probability models:*

Probability components: sample spaces e.g. finite sample spaces; events e.g. simple events, compound events; probability distributions e.g. discrete probability distributions

Probability measures: conditional probability e.g. probability given an event; likelihood e.g. numerical probability values; outcomes e.g. event outcomes

Evaluation techniques: measures e.g. probability measures; evaluation e.g. probability calculation; comparison e.g. relative probability comparison.

*State spaces and event modelling:*

State representations: finite state spaces e.g. sets of possible states; states e.g. defined conditions; transitions e.g. state transitions

Event modelling: event modelling e.g. formal event description; decision paths e.g. event sequences; scenarios e.g. possible outcome scenarios

Structural analysis: representation e.g. formal representations; structure e.g. state-space structure; analysis e.g. outcome analysis.

*Computational complexity considerations:*

Growth and scale: combinatorial growth e.g. factorial growth, exponential growth; solution space size e.g. size of combinatorial spaces; scalability e.g. growth behaviour of combinations

Efficiency and resources: complexity concepts e.g. asymptotic growth, order of magnitude; performance e.g. comparative growth rates; resources e.g. abstract resource measures.

*Applications to computing:*

System reliability applications: probability models e.g. failure models, reliability models, risk models; events e.g. fault events, recovery events, outage events; outcomes e.g. system availability, failure likelihood, reliability estimates

Decision support applications: outcome sets e.g. alternative decisions, possible actions, decision options; likelihood sets e.g. probability values, confidence levels, risk estimates; evaluations e.g. decision comparisons, expected outcomes, optimal choices

Search and optimisation applications: combinatorial spaces e.g. search spaces, solution spaces, candidate spaces; constraints e.g. search limits, feasibility constraints, optimisation bounds; results e.g. optimal solutions, feasible solutions, approximate solutions.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Apply set theory concepts relevant to discrete computational structures | | |
| **P1** Apply, using a selection of mathematical examples, the core set theory concepts relevant to discrete computational structures.<br>**P2** Discuss, using a selection of real-world examples, how set theory concepts are applied to identified computational problems. | **M1** Analyse, using a selection of real-world examples, the benefits and limitations of set theory concepts when applied to discrete computational structures. | **D1** Critically evaluate, using a selection of real-world examples, the effectiveness of set theory concepts when applied to computational structures. |
| **LO2** Apply the mathematical concepts of functions, relations, and mappings to discrete mathematical contexts and structures | | |
| **P3** Apply, using a selection of mathematical examples, the mathematical concepts of functions, relations, and mappings within discrete mathematical contexts and structures.<br>**P4** Review, using a selection of real-world examples, how functions, relations, and mappings concepts are applied to identified computational problems. | **M2** Assess, using a selection of real-world examples, the benefits and limitations of functions, relations, and mappings when applied to discrete mathematical contexts and structures. | **D2** Critically evaluate using a selection of real-world examples, the suitability of functions, relations and mappings used in modelling discrete mathematical contexts and structures. |

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Apply the key concepts and principles of graph theory relevant to problems in computing | | |
| **P5** Apply, using a selection of mathematical examples, the key concepts and principles of graph theory relevant to problems in computing.<br>**P6** Explore, using a selection of real-world examples, how graph theory concepts and principles are applied to identified computing problems. | **M3** Assess, using a selection of real-world examples, the benefits and limitations of graph theory concepts and principles when applied to modern computing problems. | **D3** Critically evaluate, using a selection of real-world examples, the use of graph theory principles in addressing modern computing problems. |
| **LO4** Apply the theory of combinatorics and discrete probability to computational decision-making | | |
| **P7** Apply, using a selection of mathematical examples, the theory of combinatorics and discrete probability to computational decision-making.<br>**P8** Explain, using a selection of real-world examples, how the theory of combinatorics and discrete probability are applied to identified computing problems. | **M4** Analyse, using a selection of real-world examples, how combinatorics and discrete probability support structured computational decision-making. | **D4** Critically evaluate, using a selection of real-world examples, the contribution of combinatorics and discrete probability theory to effective computational decision-making. |

## Recommended Resources

### Textbooks

Goodaire, E. and Parmenter, M. (2023) *Discrete Mathematics with Graph Theory*. 4th edn. London: Pearson Education.

Grami, A. (2022) *Discrete Mathematics: Essentials and Applications*. 1st edn. Oxford: Elsevier.

Lewis, H.R. and Zax, R. (2021) *Essential Discrete Mathematics for Computer Science*. Princeton, NJ: Princeton University Press.

Roberts, F.S. and Tesman, B. (2024) *Applied Combinatorics*. 3rd edn. Boca Raton: CRC Press.

Rosen, K.H. (2023) *Discrete Mathematics and Its Applications*. 9th edn. New York: McGraw-Hill.

Saoub, K.R. (2021) Graph Theory: *An Introduction to Proofs, Algorithms, and Applications.* 1st edn. Boca Raton, FL: CRC Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.csd.uwo.ca/~abrandt5/teaching/DiscreteStructures/Chapter2/set-theory.html | Western University<br>Sets and Set Theory<br>(General reference) |
| www.geeksforgeeks.org/maths/set-theory/ | GeeksforGeeks<br>Set Theory<br>(Tutorial) |
| math4cs.university/notes/01-sets_relations_functions/ | University of Pennsylvania<br>Sets, Relations, and Functions in Discrete Mathematics<br>(General reference) |
| fiveable.me/discrete-mathematics/unit-3 | Fiveable – Discrete Mathematics<br>Discrete mathematics study guide<br>(Tutorial) |
| www.csd.uwo.ca/~abrandt5/teaching/DiscreteStructures/Chapter7/index.html | Western University<br>Graph Theory<br>(General reference) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.geeksforgeeks.org/dsa/graph-theory-tutorial/ | GeeksforGeeks<br>Graph Theory Tutorial<br>(Tutorial) |
| www.csd.uwo.ca/~abrandt5/teaching/DiscreteStructures/Chapter6/index.html | Western University<br>Combinatorics and Probability<br>(General reference) |
| www.tutorialspoint.com/discrete_mathematics/discrete_mathematics_combinatorics.htm | TutorialsPoint<br>Combinatorics in Discrete Mathematics<br>(Tutorial) |
| www.tutorialspoint.com/discrete_mathematics/discrete_mathematics_probability.htm | TutorialsPoint<br>Introductory guide to discrete probability concepts, sample spaces and probability calculations.<br>(Tutorial) |
| notes.suhaib.in/docs/code/graph/theory/graph-traversals---bfs--dfs/ | Suhaib Notes<br>Graph Traversals<br>(Article) |

**Journals and articles**

Cristiá, M., Fois, A. and Rossi, G. (2023) 'Declarative Programming with Intensional Sets in Java Using JSetL', *The Computer Journal*, 66(3), pp. 763–784. Available at: doi.org/10.1093/comjnl/bxab195.

Golumbic, M.C., Hoffman, F. and Rosen, Z. (2022) 'Combinatorics, graph theory, and computing', *Journal of Algebraic Combinatorics*, 56(1), pp. 1–3. Available at: doi.org/10.1007/s10801-022-01148-9.

Majeed, A. and Rauf, I. (2020) 'Graph Theory: A Comprehensive Survey about Graph Theory Applications in Computer Science and Social Networks', *Inventions*, 5(1), 10. Available at: doi.org/10.3390/inventions5010010.

Savnik, I. *et al*. (2021) 'Data structure *set-trie* for storing and querying sets: Theoretical and empirical analysis', *PLOS ONE*, 16(2), e0245122. Available at: doi.org/10.1371/journal.pone.0245122.

**Links**

This unit links to the following related units:

*Unit 506: Advanced Software Design.*

*Unit 507: Full Stack Development.*

# Unit 509: Enterprise Database Management Systems

**Unit code:** K/652/3295

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Modern organisations rely on enterprise-scale database management systems to support critical business processes, large-scale data handling and real-time decision-making. Across sectors such as finance, healthcare, cloud services and digital platforms, enterprise databases must operate reliably at scale while maintaining performance, security and regulatory compliance. As systems become increasingly distributed and automated, database professionals are expected to work with complex architectures that underpin resilient and data-driven organisations.

The aim of this unit is to develop an advanced understanding of enterprise database management systems and the practices required to design, evaluate and implement them in organisational contexts. Students move beyond standalone or desktop databases to explore enterprise architectures, performance optimisation and automation approaches that reflect current industry expectations. The unit places an emphasis on designing and developing secure, scalable and compliant database solutions aligned to real business requirements.

This unit covers key principles of enterprise database architecture, including multi-tier, distributed and cloud-based models that support availability, scalability and resilience. Students review techniques for enterprise-level data storage, retrieval and optimisation, focusing on performance monitoring, indexing strategies, backup and recovery. Building on this foundation, students design and develop an automated, regulatory-compliant enterprise DBMS solution, applying automation, security and operational controls within a defined scenario.

On successful completion of this unit, students will be able to explain and evaluate enterprise database architectures, analyse optimisation and automation techniques, and justify the effectiveness of enterprise DBMS solutions in meeting organisational needs. These skills support progression into roles such as Database Administrator, Data Engineer, DevOps Engineer, Cloud Database Engineer and Solutions Architect, as well as further study in areas including cloud computing, data engineering and large-scale systems management.

 © Pearson Education Limited 2026

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the principles of enterprise database architectures

LO2 Review the techniques for enterprise-level data storage, retrieval and optimisation

LO3 Design a regulatory compliant, automated Enterprise DBMS solution to meet the business requirements of an identified scenario

LO4 Develop an automated Enterprise DBMS Solution to meet the regulatory and business requirements of an identified scenario.

## Essential Content

LO1 **Explain the principles of enterprise database architectures**

*Enterprise database architecture models:*

Architecture tiers: single-tier architectures e.g. standalone databases, local data processing; two-tier architectures e.g. client applications, database servers; three-tier architectures e.g. presentation layers, application logic tiers, database layers

Architecture models: client–server models e.g. database servers, client terminals; presentation layers e.g. user interfaces, web front ends; application layers e.g. business logic components, transaction processing; data layers e.g. database instances, storage subsystems

Layer separation: presentation separation e.g. user interaction logic, interface rendering; application separation e.g. business rules, workflow control; data separation e.g. data storage, data access logic

Tier responsibilities: presentation responsibilities e.g. data display, user input handling; application responsibilities e.g. processing logic, transaction management; data responsibilities e.g. data storage, data integrity management.

*Distributed and cloud-based database systems:*

Distributed concepts: distributed database concepts e.g. data location transparency, distributed transactions; horizontal distribution e.g. sharding, workload distribution

Replication and clustering: replication models e.g. primary-replica replication, read replicas; clustering e.g. failover clusters, high-availability clusters

Cloud platforms: cloud-hosted database management system (DBMS) platforms e.g. virtual machine databases, cloud database instances; managed database services e.g. database-as-a-service, automated backup services.

*Scalability and availability principles:*

Scaling approaches: vertical scaling e.g. CPU upgrades, memory expansion; horizontal scaling e.g. additional nodes, distributed workloads

Availability mechanisms: load balancing e.g. traffic distribution, workload balancing; high availability configurations e.g. failover systems, redundant nodes

Resilience architecture: fault tolerance e.g. component redundancy, error handling mechanisms; redundancy e.g. duplicate servers, mirrored storage; disaster recovery architectures e.g. backup sites, recovery environments.

*Security architectures in enterprise DBMS:*

Access control: role-based access control (RBAC) e.g. user roles, permission sets; authentication models e.g. password authentication, identity verification; authorisation models e.g. privilege assignment, access policies

Data protection: encryption at rest e.g. encrypted disks, secure storage; encryption in transit e.g. Secure Sockets Layer (SSL) connections, Transport Layer Security (TLS) protocols

Network security: network segmentation e.g. virtual local area networks (VLANs), subnet isolation;
secure connectivity between tiers e.g. protected network channels,
secure communication paths

Compliance architecture: compliance-driven architecture considerations e.g. regulatory standards, organisational security policies.

LO2 **Review the techniques for enterprise-level data storage, retrieval and optimisation**

*Enterprise data storage techniques:*

Data storage models: logical data storage e.g. schemas, relational structures; physical data storage e.g. disk layouts, storage devices, I/O subsystems

Storage organisation: partitioning strategies e.g. range partitioning, hash partitioning, workload segmentation; storage engines e.g. transactional engines, analytical engines

Capacity planning: capacity planning e.g. storage forecasting, resource allocation, demand analysis; data growth management e.g. trend monitoring,
storage life cycle control.

*Data retrieval and query processing:*

Query processing stages: query execution life cycle e.g. parsing stages, execution stages, result handling; query optimisation e.g. query rewriting, optimisation rules

Execution analysis: execution plans e.g. query plans, operator trees;
cost-based optimisation e.g. resource estimation, cost modelling

Database logic: stored procedures e.g. reusable queries, controlled execution logic; triggers e.g. event-driven logic, automated enforcement rules.

*Indexing and optimisation strategies:*

Index structures: index types e.g. B-tree indexes, hash indexes, clustered indexes; composite indexes e.g. multi-column indexes, combined key structures

Query optimisation indexes: covering indexes e.g. index-only queries, reduced table access; index selection e.g. workload analysis, query pattern evaluation

Performance considerations: index tuning e.g. structure refinement, performance adjustment; write performance impact e.g. insert overhead, update cost trade-offs.

*Performance monitoring and maintenance:*

Performance measurement: database performance metrics e.g. response time, throughput, resource utilisation; monitoring tools e.g. performance dashboards, monitoring platforms

Performance analysis: query performance analysis e.g. slow query identification, execution profiling; routine maintenance tasks e.g. reindexing, cleanup operations.

*Backup, recovery and resilience:*

Backup methods: backup strategies e.g. scheduled backups, automated backups; incremental backups e.g. change-based backups, delta backups

Recovery techniques: restore procedures e.g. database restoration, system recovery; point-in-time recovery e.g. transaction logs, recovery checkpoint.

LO3 **Design a regulatory compliant, automated Enterprise DBMS solution to meet the business requirements of an identified scenario**

*Requirements analysis for enterprise DBMS design:*

Requirements classification: functional requirements e.g. data processing needs, user access needs; non-functional requirements e.g. performance expectations, scalability targets, resilience constraints

Operational constraints: performance constraints e.g. response time thresholds, throughput limits; availability requirements e.g. uptime targets, service continuity

Risk and compliance: security considerations e.g. threat exposure, access risk; regulatory compliance: data protection principles e.g. confidentiality, integrity, accountability; GDPR principles e.g. lawful processing, purpose limitation; knowledge currency

Monitoring and auditing: access auditing e.g. access logs, audit trails; compliance monitoring e.g. policy checks, regulatory reporting.

*Enterprise DBMS solution architecture design:*

Architectural structure: logical architecture design e.g. component relationships, system boundaries, data flows; physical architecture design e.g. server placement, storage configuration

Deployment strategy: deployment topologies e.g. centralised deployment, distributed deployment; environment separation e.g. development environments, testing environments, production environments

System integration: integration with enterprise systems e.g. application platforms, identity services.

*Database schema and structural design:*

Data modelling: schema modelling e.g. entity definitions, attribute structures; entity relationships e.g. one-to-many relationships, many-to-many relationships, dependency mappings

Data integrity: constraints e.g. primary keys, foreign keys; integrity rules e.g. referential integrity, domain integrity

Design validation: normalisation considerations e.g. redundancy reduction, dependency resolution; acceptance criteria e.g. structural correctness, business rule alignment.

*Automation and infrastructure design:*

Automation approach: infrastructure-as-code concepts e.g. declarative definitions, configuration templates; automated provisioning e.g. environment creation, resource allocation

Deployment consistency: configuration management e.g. version-controlled configurations, parameter management; repeatable deployment processes e.g. standardised builds, automated rollout.

*Design documentation and validation:*

Design representation: architecture diagrams e.g. logical diagrams, deployment diagrams; design specifications e.g. technical descriptions, system assumptions, constraint definitions

Validation activities: schema documentation e.g. data dictionaries, structural descriptions; validation against requirements e.g. requirements traceability, design justification.

LO4 **Develop an automated Enterprise DBMS solution to meet the regulatory and business requirements of an identified scenario**

*Enterprise DBMS implementation:*

Database implementation: database creation e.g. instance deployment, database initialisation; schema implementation e.g. table creation, constraint definition, index deployment

Access configuration: role and permission configuration e.g. user roles, access privileges, least-privilege enforcement; environment setup e.g. runtime environments, configuration parameters.

*Security and data protection implementation:*

Security configuration: role-based security configuration e.g. access roles, permission mapping; access control enforcement e.g. policy application, authentication checks

Data protection mechanisms: encryption mechanisms e.g. data-at-rest encryption, data-in-transit encryption; secure credential management e.g. credential storage, secrets handling

Compliance implementation: compliance implementation e.g. policy alignment, regulatory configuration.

*Automation and deployment processes:*

Automation tooling: automation scripts e.g. deployment scripts, configuration scripts, provisioning scripts; version control of database assets e.g. schema repositories, change tracking

Deployment pipelines: CI/CD integration for databases e.g. automated testing stages, deployment gating; deployment pipelines e.g. staged deployment, controlled releases.

*Testing, monitoring and operational readiness:*

Testing activities: functional testing e.g. requirement validation, data integrity testing; performance testing e.g. load testing, stress testing, baseline comparison

Monitoring configuration: monitoring and alerting configurations e.g. performance alerts, threshold monitoring; operational dashboards e.g. system status views, performance summaries.

*Operational documentation and evaluation:*

Operational documentation: runbooks e.g. support procedures, recovery steps; operational procedures e.g. maintenance schedules, incident handling

Solution evaluation: test evidence e.g. test results, validation outputs; evaluation of solution effectiveness e.g. requirement satisfaction, operational suitability.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the principles of enterprise database architectures</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate the effectiveness of different enterprise database architectures in supporting the organisational requirements of enterprise-level data storage, retrieval and optimisation.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the principles of enterprise database architectures and how they can be applied to distributed or cloud based systems.<br><br>P2 Review, using a selection of real-world examples, the principles of scalability and security in enterprise database architectures.</td><td>M1 Analyse, using a selection of real-world examples, the benefits and limitations of different enterprise architectures in supporting scalability, security and performance in organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Review the techniques for enterprise-level data storage, retrieval and optimisation</td></tr>
<tr><td>P3 Review, using a selection of real-world examples, the techniques used for data storage and retrieval used within enterprise database systems.<br><br>P4 Explore, using a selection of real-world examples, the techniques used for performance, optimisation and backups within enterprise database systems.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of the techniques of enterprise level data storage, retrieval and optimisation for organisations using these database systems.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a regulatory compliant, automated Enterprise DBMS solution to meet the business requirements of an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of the enterprise DBMS in how comprehensively it meets the regulatory and business requirements of the identified scenario. |
| **P5** Design a regulatory compliant, automated enterprise DBMS solution to meet the business requirements of an identified scenario.<br>**P6** Review the design documentation developed for the enterprise DBMS scenario with a range of stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the stakeholder feedback, the enterprise database design so that it comprehensively addresses the business requirements of the identified scenario. | |
| **LO4** Develop an automated Enterprise DBMS solution to meet the regulatory and business requirements of an identified scenario | | |
| **P7** Develop an automated Enterprise DBMS solution to meet the regulatory and business requirements of an identified scenario.<br>**P8** Implement a test plan that thoroughly tests the developed enterprise DBMS solution in the context of meeting the regulatory and business requirements of the identified scenario. | **M4** Evaluate, using the results of the system testing, the effectiveness of the enterprise DBMS solution in meeting the regulatory and business requirements of the identified scenario. | |

## Recommended Resources

### Textbooks

Coronel, C. and Morris, S. (2024) *Database Systems: Design, Implementation, and Management*. 13th edn. Boston, MA: Cengage Learning.

Elmasri, R. and Navathe, S.B. (2016) *Fundamentals of Database Systems*. 7th edn. Harlow: Pearson.

Hoffer, J.A., Topi, H. and Venkataraman, R. (2025) *Modern Database Management*. 14th edn. Harlow: Pearson Education.

Lahdenmäki, T. and Leach, M. (2023) *Relational Database Index Design and the Optimizers: SQL Server, Oracle, and DB2*. Hoboken, NJ: Wiley.

Patni, J.C. *et al*. (2022) *Database Management System: An Evolutionary Approach*. Boca Raton, FL: CRC Press.

### Websites

*Website links were valid at the time of initial publication*.

| | |
|---|---|
| www.geeksforgeeks.org/system-design/client-server-architecture-system-design/ | GeeksforGeeks<br>Client–Server Architecture – System Design.<br>(Article) |
| em360tech.com/tech-articles/client-server-architecture-enterprise-it | EM360Tech<br>What Is Client–Server Architecture and Why It Still Matters in Modern Enterprise IT<br>(Article) |
| www.geeksforgeeks.org/sql/index-optimization/ | GeeksforGeeks<br>Index Optimisation<br>(Reference) |
| www.liquibase.com/resources/guides/gdpr-compliance-for-database-management-privacy-by-design-best-practices | Liquibase<br>GDPR Compliance for Database Management: Privacy by Design Best Practices<br>(Guide) |
| blogs.vmware.com/cloud-foundation/2026/02/03/achieve-infrastructure-as-code-for-databases-a-guide-for-platform-engineers/ | VMware<br>Achieve Infrastructure-as-Code for Databases: A Guide for Platform Engineers<br>(Tutorial) |

| | |
|---|---|
| www.dbmaestro.com/blog/database-ci-cd/how-to-automate-database-deployments-with-ci-cd-pipelines/ | DBmaestro<br>How to Automate Database Deployments with CI/CD Pipelines<br>(Blog) |
| usavps.com/blog/database-encryption-explained-protecting-data-at-rest-and-in-transit/ | USAVPS<br>Database Encryption Explained: Protecting Data At Rest and In Transit<br>(Article) |
| docs.aws.amazon.com/whitepapers/latest/logical-separation/encrypting-data-at-rest-and--in-transit.html | Amazon Web Services<br>Encrypting Data-at-Rest and Data-in-Transit<br>(White Paper) |

**Journals and articles**

Chilakala, S. (2025) 'Enterprise Data Architectures: A Comprehensive Analysis of Modern Solutions, Market Trends, and Implementation Frameworks', *International Journal of Research in Computer Applications and Information Technology*, 8(1), pp. 3388–3399. Available at: doi.org/10.34218/IJRCAIT_08_01_243.

Gupta, N.V.R.S.C.L. (2023) 'Database Optimization Strategies: Enhancing Performance and Scalability', *International Journal of Computer Science and Mobile Computing*, 12(11), pp. 69–89. Available at: doi.org/10.47760/ijcsmc.2023.v12i11.006.

Jalukuri, M.B. (2025) 'Infrastructure as Code: Revolutionizing Cloud Deployment and Management', *World Journal of Advanced Engineering Technology and Sciences*, 15(2), pp. 2394–2401. Available at: doi.org/10.30574/wjaets.2025.15.2.0799.

Saxena, A. *et al*. (2025) 'DevOps Automation Pipeline Deployment with IaC (Infrastructure as Code)', *IEEE Silchar Subsection Conference Proceedings*. Available at: doi.org/10.48550/arXiv.2503.16038.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 510: Advanced Data Engineering*

*Unit 511: Big Data Analytics and Visualisation.*

# Unit 510: Advanced Data Engineering

**Unit code:** **L/652/3296**

**Unit level:** **5**

**Credit value:** **15**

---

## Introduction

The continued growth of data-driven organisations has placed data engineering at the centre of modern computing practice. Across sectors such as finance, healthcare, retail, technology and the public sector, organisations depend on complex data systems to collect, process and manage large volumes of data in a reliable and scalable way. This unit reflects the practical and architectural challenges faced when designing and operating large-scale data platforms in real organisational contexts.

The aim of this unit is to develop students' understanding of advanced data engineering concepts that underpin contemporary data architectures. It focuses on how data engineering decisions support scalability, reliability and long-term data management, while responding to business, technical and operational requirements. Through this unit, students move beyond foundational data handling to consider how complex data systems are structured, integrated and justified within professional environments.

This unit covers a range of advanced topics, including data engineering architectures, data pipelines, processing models, storage approaches and data life cycle management. Students will explore how large-scale data systems achieve high availability, fault tolerance and performance, and how data can be integrated and governed across cloud, on-premise and hybrid platforms. The unit also provides opportunities to apply these concepts by designing and developing a prototype data engineering solution for a given scenario, linking architectural theory with practical design and testing considerations.

On successful completion of this unit, students will be able to analyse and evaluate advanced data engineering solutions, justify architectural and integration decisions, and design data pipelines that meet defined requirements. These skills support progression into roles such as Data Engineer, Data Platform Engineer, Analytics Engineer, Cloud Data Specialist and Big Data Developer, as well as further study in data engineering, data architecture and related computing disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore the concepts of advanced data engineering architectures and data life cycle models

LO2 Discuss the scalability and reliability of large-scale data systems

LO3 Review the techniques for integrating and managing data from different platforms

LO4 Develop a prototype data engineering solution for a given scenario.

## Essential Content

LO1 **Explore the concepts of advanced data engineering architectures and data life cycle models**

*Data engineering architectures:*

Architecture types: centralised architectures e.g. single data platforms, shared databases; decentralised architectures e.g. distributed data ownership, multiple data stores; lambda architectures e.g. batch processing layer, real-time processing layer, serving layer; kappa architectures e.g. streaming pipelines, continuous processing

Data platform architectures: data lake architectures e.g. raw data storage, flexible schemas; data warehouse architectures e.g. structured data storage, reporting systems, analytical queries; lakehouse architectures e.g. combined storage models, analytics support.

*Data pipelines and ingestion models:*

Ingestion models: batch ingestion models e.g. scheduled data loads, periodic updates, file transfers; streaming ingestion models e.g. live data feeds, continuous ingestion, real-time updates; event-driven ingestion e.g. system events, message triggers

Ingestion management: data ingestion orchestration e.g. workflow co-ordination, task scheduling, process control; ingestion reliability e.g. error handling, data completeness; ingestion latency considerations e.g. processing delays, response time, system performance.

*Processing and computation models:*

Processing frameworks: batch processing frameworks e.g. large dataset processing, offline analysis; stream processing frameworks e.g. continuous processing, event handling, real-time analytics

Processing characteristics: stateful processing e.g. session tracking, stored context; stateless processing e.g. independent requests, simple transformations, no stored memory.

*Storage layers and data organisation:*

Data layers: raw data layers e.g. unprocessed data, source data; curated data layers e.g. cleaned datasets, transformed data; serving data layers e.g. reporting datasets, dashboards

Data structuring approaches: schema-on-read e.g. flexible formats, late validation, data exploration; schema-on-write e.g. predefined structures, validation rules.

*Data life cycle management:*

Life cycle stages: data life cycle stages e.g. data creation, data usage, data disposal; data retention policies e.g. storage duration, organisational rules; archival strategies e.g. long-term storage, reduced access, cost-efficient storage

Compliance controls: data deletion e.g. secure removal, data disposal; life cycle automation e.g. automated workflows, rule-based actions.

LO2 **Discuss the scalability and reliability of large-scale data systems**

*Scalability principles:*

Scaling approaches: horizontal scaling e.g. adding servers, parallel resources, shared workloads; vertical scaling e.g. increased memory, processor upgrades, hardware enhancement; elastic compute e.g. on-demand resources, automatic scaling, flexible capacity

Resource management: resource provisioning e.g. capacity planning, resource allocation, system availability; workload variability e.g. fluctuating demand, peak usage, usage patterns.

*Distributed storage and processing:*

Storage systems: distributed file systems e.g. shared file access, network storage, scalable storage; distributed databases e.g. replicated data, shared access, fault tolerance

Data handling models: data replication models e.g. duplicated data, redundancy, data availability; data locality considerations e.g. data placement, access speed, network efficiency.

*Fault tolerance and resilience:*

Reliability strategies: replication strategies e.g. multiple copies, backup systems, data safety; failover mechanisms e.g. automatic switching, service continuity, alternative resources

System robustness: redundancy models e.g. duplicate components, backup infrastructure, system reliability; graceful degradation e.g. reduced functionality, partial service, controlled performance loss.

*System availability and performance:*

Availability design: high availability architectures e.g. continuous operation, minimal downtime, system uptime; load balancing strategies e.g. traffic distribution, workload sharing, performance stability

Performance management: performance optimisation e.g. efficiency improvements, faster processing, reduced delays; latency management e.g. response times, transmission delays, user experience.

*Disaster recovery and continuity:*

Recovery planning: backup strategies e.g. data copies, scheduled backups, recovery readiness; recovery time objectives e.g. acceptable downtime, service restoration targets, operational limits; recovery point objectives e.g. data loss limits, restore points, backup frequency

Business resilience: disaster recovery planning e.g. recovery procedures, system restoration, incident response; business continuity alignment e.g. operational continuity, service priorities, organisational resilience.

LO3 **Review the techniques for integrating and managing data from different platforms**

*Data integration techniques:*

Integration approaches: extract, transform and load (ETL) e.g. data extraction, data transformation, data loading; extract, load and transform (ELT) e.g. cloud data loading, in-platform processing, data preparation; real-time data integration e.g. live data updates, continuous flows, event handling; batch integration e.g. scheduled transfers, periodic updates, grouped processing.

*Platform interoperability:*

Deployment environments: cloud environments e.g. hosted services, remote infrastructure, scalable platforms; on-premise environments e.g. local servers, internal networks, organisational infrastructure; hybrid architectures e.g. combined cloud and local systems, integrated platforms, shared resources; multi-cloud integration e.g. multiple providers, distributed services, vendor diversity.

*Integration technologies:*

Communication tools: Application Programming Interfaces (APIs) e.g. system interfaces, data access points, service communication; middleware platforms e.g. message handling, system co-ordination, data routing; message brokers e.g. message queues, event delivery, system messaging

Synchronisation methods: data synchronisation mechanisms e.g. data updates, consistency checks, aligned datasets.

*Data standardisation and optimisation:*

Data alignment: schema mapping e.g. field matching, structure alignment, format consistency; data transformation rules e.g. data conversion, value changes, format standardisation; data normalisation e.g. reduced redundancy, consistent structures, organised datasets

Performance improvement: optimisation techniques e.g. efficient queries, reduced storage usage, improved processing.

*Data governance and quality management:*

Governance structures: data governance frameworks e.g. organisational policies, management structures, compliance models; data ownership e.g. responsibility allocation, data accountability, assigned roles

Quality controls: data quality dimensions e.g. accuracy, completeness, consistency; validation rules e.g. data checks, rule enforcement, error detection; monitoring processes e.g. ongoing review, performance tracking, issue identification.

LO4 **Develop a prototype data engineering solution for a given scenario**

*Scenario analysis and requirements specification:*

Scenario interpretation: analysis of organisational context e.g. business objectives, operational constraints, stakeholder needs; identification of data requirements e.g. data sources, data types, data volumes

Requirements definition: functional requirements e.g. ingestion needs, processing objectives, output expectations; non-functional requirements e.g. performance, scalability, reliability, data quality constraints; success criteria e.g. acceptance conditions, measurable outcomes, scenario alignment.

*Data pipeline design and implementation:*

Pipeline structure: ingestion pipeline design e.g. batch pipelines, streaming pipelines, event-driven flows; transformation stages e.g. data cleansing, enrichment, aggregation; output layers e.g. analytical datasets, reporting structures

Design choices: processing model selection e.g. batch vs streaming; orchestration considerations e.g. workflow sequencing, dependency management; design trade-offs e.g. performance versus complexity, flexibility versus control.

*Integration, testing, and validation:*

System integration: platform connectivity e.g. cloud services, on-premise systems, hybrid environments; data interoperability e.g. format compatibility, interface alignment, integration points

Testing and validation: data validation checks e.g. accuracy, completeness, consistency; pipeline testing e.g. test data flows, failure scenarios, error handling; validation against requirements e.g. meeting scenario objectives, confirming expected outcomes.

*Deployment considerations and change management:*

Deployment context: deployment models e.g. cloud-based deployment, on-premise deployment, hybrid deployment; environment configuration e.g. development, testing, production separation

Change management: version control e.g. change tracking, controlled updates; maintainability considerations e.g. documentation, configurability, supportability; adaptability to change e.g. evolving requirements, future enhancements, system scalability.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explore the concepts of advanced data engineering architectures and data life cycle models</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of advanced data architectures and models in integrating data from multiple sources and supporting the development of scalable and reliable data systems.</td></tr>
<tr><td>P1 Explore, using a selection of real-world examples, how advanced data engineering architectures help support data life cycle management.<br><br>P2 Discuss, using a selection of real-world examples, the concepts of data pipelining, processing and storage within advanced data engineering architectures.</td><td>M1 Analyse, using a selection of real-world examples, the applications of advanced data engineering architectures and concepts in a range of organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Discuss the scalability and reliability of large-scale data systems</td></tr>
<tr><td>P3 Discuss, using a selection of real-world examples, the principles of scalability as applied to large-scale distributed data systems.<br><br>P4 Describe, using a selection of real-world examples, the principles of reliability as applied to large-scale distributed data systems.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of implementing scalable and reliable large-scale distributed data systems in a range of organisational contexts.</td></tr>
<tr><td colspan="2">LO3 Review the techniques for integrating and managing data from different platforms</td></tr>
<tr><td>P5 Review, using a selection of real-world examples, data integration and management techniques used across identified platforms.<br><br>P6 Explore, using a selection of real-world examples, how data governance and quality management is maintained when integrating data from different platforms.</td><td>M3 Analyse, using a selection of real-world examples, the application of data integration and management techniques across a range of different platforms and organisations.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Develop a prototype data engineering solution for a given scenario | | |
| **P7** Design a prototype data engineering solution that addresses the requirements of a given scenario.<br><br>**P8** Develop a prototype data engineering solution that addresses the requirements of a given scenario.<br><br>**P9** Implement a test plan that thoroughly tests the developed data engineering solution in the context of meeting the identified requirements of the given scenario. | **M4** Evaluate, using the results of the system testing, the effectiveness of the data engineering solution in meeting the identified requirements of the given scenario. | **D2** Justify the decisions made in the design and development of the data engineering solution in how comprehensively it meets the identified requirements of the given scenario. |

## Recommended Resources

### Textbooks

Burns, B. (2022) *Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services.* 2nd edn. Sebastopol: O'Reilly Media.

Kleppmann, M. (2023) *Designing Event-Driven Systems: Concepts and Patterns for Streaming Services with Apache Kafka*. Sebastopol, CA: O'Reilly Media.

Kleppmann, M. and Riccomini, C. (2026) *Designing Data-Intensive Applications*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Marz, N. and Warren, J. (2022) *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. 2nd edn. Shelter Island: Manning Publication.

Reis, J. and Housley, M. (2023) *Fundamentals of Data Engineering*. Sebastopol: O'Reilly Media.

Vohra, D. (2024) *Practical Distributed Data Systems*. 2nd edn. New York: Apress.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| lumenalta.com/insights/mastering-data-engineering-architecture-for-scalable-solutions | Lumenalta<br>Mastering data engineering architecture<br>(Article) |
| cloud.google.com/architecture | Google Cloud Architecture Centre<br>Foundational architecture guidance for enterprise deployments in Google Cloud<br>(Resource Hub) |
| databricks.com/glossary/data-lakehouse | Databricks<br>What is a Data Lakehouse?<br>(Article) |
| www.ibm.com/topics/data-governance | IBM<br>What is data governance?<br>(General reference) |
| learn.microsoft.com/en-gb/azure/architecture/browse/ | Microsoft Learn<br>Azure architectures<br>(General reference) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.geeksforgeeks.org/system-design/scaling-distributed-systems/ | GeeksforGeeks<br>Scaling Distributed Systems<br>(General reference) |
| learn.microsoft.com/en-us/azure/architecture/data-guide/relational-data/etl | Microsoft Learn<br>Extract, transform, and load (ETL)<br>(General reference) |
| www.datacamp.com/blog/etl-vs-elt | DataCamp<br>ETL vs ELT: Understanding the Differences and Making the Right Choice<br>(Blog) |
| qualytics.ai/data-governance-and-quality | Qualytics<br>Data governance and data quality<br>(General reference) |
| dataengineeracademy.com/blog/building-data-pipelines-a-step-by-step-guide-2023/ | Data Engineer Academy<br>Building Data Pipelines: A Step-by-Step Guide<br>(Tutorial) |

**Journals and articles**

Armbrust, M. *et al*. (2021) 'Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics', *CIDR 2021 Conference*. Available at: www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf.

Pahl, C., Jamshidi, P. and Zimmermann, O. (2018) 'Architectural Principles for Cloud Software', *ACM Transactions on Internet Technology*, 18(2), Article 17. Available at: doi.org/10.1145/3104028.

Zaharia, M. *et al*. (2016) 'Apache Spark: a unified engine for big data processing', *Communications of the ACM*, 59(11), pp. 56–65. Available at: doi.org/10.1145/2934664.

Zárate, G. *et al*. (2024) 'Evolution of Extract-Transform-Load (ETL) processes towards data product pipelines', *Proceedings of the Eclipse Security, AI, Architecture and Modelling Conference on Data Space (eSAAM 2024)*. Available at: doi.org/10.1145/3685651.3686662.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 509: Enterprise Database Management Systems*

*Unit 511: Big Data Analytics and Visualisation.*

# Unit 511: Big Data Analytics and Visualisation

**Unit code:** M/652/3297

**Unit level:** 5

**Credit value:** 15

---

## Introduction

As organisations continue to operate in increasingly data-driven environments, the volume, speed and variety of data being generated has grown significantly. Across sectors such as finance, healthcare, retail, manufacturing and digital services, organisations rely on the ability to analyse large datasets and communicate insights clearly to support operational and strategic decision-making. Big data analytics and visualisation are therefore essential skills for individuals working in modern computing and data-focused roles, where both technical understanding and effective communication are required.

The aim of this unit is to introduce students to the core concepts and technologies that underpin big data analytics and data visualisation. It focuses on developing an understanding of how large-scale data systems operate, how data is processed and analysed at scale, and how insights can be presented in ways that are meaningful to different audiences. The unit places equal emphasis on analytical techniques and visual communication, recognising that data-driven insights only have value when they can be clearly interpreted and applied.

This unit begins by exploring the fundamental characteristics and architectures of big data systems, including how large and complex datasets are stored and processed. Students then apply big data analytics principles to the processing and analysis of large-scale datasets, considering scalability, performance and analytical workflows. The unit also examines key data visualisation principles, tools and techniques used to communicate insights, including dashboards and storytelling approaches. The unit culminates in the design and development of a data visualisation prototype for an identified scenario, bringing together analytical, technical and design skills.

On successful completion of this unit, students will be able to analyse large datasets, apply big data analytics approaches, evaluate data visualisation tools and develop solutions that support informed decision-making. These skills are directly relevant to roles such as Data Analyst, Junior Data Engineer, Business Intelligence Analyst, Reporting Analyst and Analytics Technician, and also support progression to further study in data analytics, data science and related computing disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core concepts, characteristics and architectures underpinning big data systems

LO2 Apply the principles of big data analytics to the processing and analysis of large-scale datasets

LO3 Review a range of data visualisation tools and techniques to communicate insights obtained from large datasets

LO4 Develop a prototype of a data visualisation solution that meets the requirements of an identified scenario.

## Essential Content

LO1 **Explain the core concepts, characteristics and architectures underpinning big data systems**

*Big Data concepts and characteristics:*

Data dimensions: volume e.g. large datasets, data size measurement; velocity e.g. data generation speed, real-time data; variety e.g. different data formats, mixed data types; veracity e.g. data accuracy, data reliability; value e.g. business benefit, decision support

Data scale and growth: data scale e.g. organisational data size, enterprise datasets; data growth patterns e.g. increasing data volumes, data expansion trends; data complexity e.g. multiple data relationships, processing difficulty

Data structure and composition: structured data e.g. tables, spreadsheets; semi-structured data e.g. JavaScript Object Notation (JSON) files, eXtensible Markup Language (XML) documents; unstructured data e.g. text files, images; data heterogeneity e.g. mixed data sources, varied data formats.

*Big data system architectures:*

Processing architecture: distributed architecture e.g. multiple machines, shared processing; cluster computing e.g. server clusters, grouped nodes; parallel processing e.g. simultaneous tasks, concurrent execution

System performance and scalability: scalability e.g. system expansion, resource scaling; processing efficiency e.g. faster computation, reduced processing time; workload distribution e.g. task allocation, load sharing

System reliability and resilience: fault tolerance e.g. system redundancy, failure recovery; system resilience e.g. continued operation, system stability; data locality e.g. processing near storage, reduced data movement

Resource control and management: resource management e.g. memory allocation, processor usage; workload distribution e.g. balanced processing, task scheduling; processing efficiency e.g. optimised performance, reduced overhead.

*Big data platforms and ecosystems:*

Platform frameworks: Hadoop ecosystem e.g. Hadoop Distributed File System (HDFS) components, MapReduce framework; data processing frameworks e.g. batch frameworks, stream frameworks; query engines e.g. Structured Query Language (SQL) queries, data retrieval tools

Storage and access: distributed storage e.g. data replication, shared storage; file systems e.g. distributed file systems, large file handling; data access e.g. user access methods, data retrieval processes

Platform services and co-ordination: ecosystem tools e.g. monitoring tools, management utilities; resource co-ordination e.g. job scheduling, resource allocation; platform integration e.g. system compatibility, tool connectivity

Deployment environments: cloud platforms e.g. public cloud services, hosted environments; platform integration e.g. multi-platform environments, service integration; resource co-ordination e.g. cloud scheduling, shared resources.

*Big data data sources and analytics types:*

Data sources: transactional data e.g. sales records, purchase logs; sensor data e.g. Internet of Things (IoT) devices, environmental sensors; log data e.g. system logs, application logs

Types of analytics: descriptive analytics e.g. summary reports, historical analysis; predictive analytics e.g. trend forecasting, future predictions

Analytical interpretation: data patterns e.g. recurring trends, usage patterns; data grouping e.g. category clustering, data segmentation.

LO2 **Apply the principles of big data analytics to the processing and analysis of large-scale datasets**

*Big data analytics workflows:*

Data pipeline inputs: data ingestion e.g. log files, sensor readings; data preprocessing e.g. missing values, duplicate records; data transformation e.g. format conversion, value scaling

Data pipeline management: data storage e.g. data lake, distributed storage; data integration e.g. merged datasets, combined sources; data sequencing e.g. event order, batch order

Workflow structure: processing stages e.g. ingestion stage, processing stage; workflow design e.g. workflow steps, workflow order; pipeline structure e.g. pipeline stages, pipeline flow; analytics output e.g. summary reports, insight outputs.

*Scalability and processing models:*

Scalability approaches: horizontal scalability e.g. extra nodes, cluster growth; vertical scalability e.g. more memory, faster processor; system capacity e.g. storage capacity, compute capacity

Processing models: batch processing e.g. scheduled jobs, bulk processing; stream processing e.g. live streams, continuous data; real-time processing

Performance measures: processing latency e.g. response delay, system lag; throughput e.g. records per second, processing rate; workload handling e.g. peak loads, concurrent tasks; performance efficiency e.g. resource use, runtime performance.

*Distributed processing frameworks:*

Distributed execution: distributed processing e.g. cluster processing, multi-node processing; parallel execution e.g. parallel tasks, concurrent jobs; execution control e.g. job control, task control

Co-ordination mechanisms: Processing frameworks e.g. framework tools, framework services; Task co-ordination e.g. task scheduling, task assignment; System co-ordination e.g. node co-ordination, resource co-ordination

Reliability handling: Data partitioning e.g. data shards, split datasets; workload distribution e.g. load balancing, worker assignment; fault handling e.g. error recovery, task retry; processing reliability e.g. stable processing, consistent results.

*Analytical feature handling:*

Feature methods: feature extraction e.g. key variables, useful field; feature selection e.g. selected variables, reduced features; pattern identification e.g. trends, clusters

Data shaping: data aggregation e.g. grouped totals, summary metrics; data reduction e.g. reduced dataset, sampled data; data transformation e.g. normalised data, encoded values; data representation e.g. tables, feature sets

Data readiness: attribute handling e.g. data types, attribute formats; Input preparation e.g. prepared inputs, cleaned inputs; analytical readiness e.g. validated dataset, analysis-ready data.

LO3 **Review a range of data visualisation tools and techniques to communicate insights obtained from large datasets**

*Data visualisation principles:*

Visual perception: visual perception e.g. human vision, pattern recognition; visual clarity e.g. clear labels, readable scales; data accuracy e.g. correct values, reliable data

Visual design quality: visual scalability e.g. large datasets, high data volumes; visual encoding e.g. position, colour, size; colour usage e.g. colour schemes, contrast; accessibility e.g. readable text, colour contrast

Interpretation support: consistency e.g. consistent scales, consistent symbols; interpretability e.g. easy interpretation, clear meaning; visual effectiveness e.g. clear insights, effective communication.

*Visualisation techniques and chart types:*

Comparison techniques: comparative charts e.g. bar charts, column charts; Distribution techniques: distribution charts e.g. histograms, box plots; Relationship techniques: relational charts e.g. scatter plots, network charts

Time and location techniques: temporal charts e.g. line charts, time series; geospatial charts e.g. maps, heat maps; hierarchical charts e.g. tree maps, hierarchy diagrams

Interaction techniques: interactive charts e.g. filters, drill-down; summary visuals e.g. summary charts, overview panels; trend visuals e.g. trend lines, progression charts; categorical visuals e.g. category charts, grouped data.

*Dashboards and storytelling approaches:*

Dashboard structure: dashboard design e.g. dashboard layout, dashboard components; information hierarchy e.g. key metrics, priority ordering; layout structure e.g. grid layouts, section organisation

Storytelling flow: narrative flow e.g. logical sequence, insight progression; contextual framing e.g. business context, analytical context; insight communication e.g. key findings, messages

Audience focus: audience focus e.g. user needs, stakeholder needs; visual sequencing e.g. visual order, reading flow; decision support e.g. informed decisions, action support; message emphasis e.g. highlighted insights, key points.

*Big data visualisation tools:*

Tool categories: business intelligence tools e.g. Power BI, SAP BusinessObjects; data visualisation tools e.g. Tableau, Power BI; open-source tools e.g. Apache Superset, Grafana

Deployment environments: cloud-based tools e.g. Power BI Service, Tableau Cloud; platform integration e.g. Microsoft ecosystem, enterprise platforms; data connectivity e.g. SQL databases, cloud data sources

Operational capabilities: tool interoperability e.g. common data formats, shared connectors; reporting tools e.g. Power BI reports, Tableau dashboards; analytics platforms e.g. Power BI platform, Tableau platform; visualisation software e.g. Tableau Desktop, Power BI Desktop; AI tools e.g. Power BI Copilot, Tableau Einstein.

LO4 **Develop a prototype of a data visualisation solution that meets the requirements of an identified scenario**

*Solution design and requirements analysis:*

User needs: user needs e.g. information needs, usability needs; stakeholder requirements e.g. business goals, reporting needs; functional requirements e.g. required features, system functions

Design constraints: non-functional requirements e.g. performance limits, security needs; data requirements e.g. data sources, data formats; usage context e.g. user environment, operational context

Design definition: solution scope e.g. project boundaries, solution limits; requirement priorities e.g. high priority needs, key requirements; success criteria e.g. measurable outcomes, acceptance criteria.

*Tool selection and solution architecture:*

Tool evaluation: tool selection e.g. appropriate tools, suitable platforms; selection criteria e.g. capability fit, ease of use; platform compatibility e.g. supported platforms, system compatibility

System integration: data connectivity e.g. source connections, data links; system integration e.g. integrated components, connected systems; deployment context e.g. deployment environment, operational setting

Architecture considerations: scalability considerations e.g. future growth, increased users; performance constraints e.g. response time limits, processing limits; architectural structure e.g. system layout, solution structure; solution alignment e.g. requirement alignment, organisational fit.

*Prototyping and development approaches:*

Prototype development: low-fidelity prototypes e.g. sketches, wireframes; high-fidelity prototypes e.g. interactive mock-ups, detailed layouts; Iterative design e.g. design cycles, repeated refinement

Design construction: layout structure e.g. visual layout, screen structure; visual composition e.g. visual balance, element placement; interaction design e.g. user interactions, navigation

Development process: design refinement e.g. layout adjustment, visual improvement; prototype evaluation e.g. feedback sessions, usability review; development workflow e.g. build stages, development steps; solution iteration e.g. revised versions, incremental improvement.

*Evaluation and refinement:*

Evaluation methods: usability e.g. ease of use, learnability; user feedback e.g. user comments, stakeholder feedback; effectiveness e.g. insight clarity, communication effectiveness

Alignment checks: communication clarity e.g. message clarity, visual clarity; requirement alignment e.g. matched requirements, fulfilled needs; performance review e.g. system performance, response behaviour

Improvement actions: issue identification e.g. identified problems, detected gaps; improvement actions e.g. design changes, solution updates; solution refinement e.g. final improvements, adjusted solution; design validation e.g. validated design, approved solution.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the core concepts, characteristics and architectures underpinning big data systems</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of big data architectures and analytics in supporting large-scale enterprise data processing requirements and making effective business decisions.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the core concepts and characteristics of big data systems.<br><br>P2 Review, using a selection of real-world examples, the architectures, platforms and ecosystems used by big data systems.</td><td>M1 Assess, using a selection of real-world examples, the benefits and limitations of big data concepts, characteristics and architectures when applied in different organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Apply the principles of big data analytics to the processing and analysis of large-scale datasets</td></tr>
<tr><td>P3 Review, using a selection of real-world examples, the range of big data analytics principles that can be applied to the processing of large-scale datasets.<br><br>P4 Apply the principles of big data analytics to the processing and analysis of datasets to obtain meaningful insights and conclusions.</td><td>M2 Analyse the effectiveness of applied big data analytics principles in processing and analysing large-scale datasets to obtain meaningful insights and conclusions.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Review a range of data visualisation tools and techniques to communicate insights obtained from large datasets | | **LO3 and LO4**<br>**D3** Justify the decisions made in the design, tool selection and implementation of the data visualisation prototype in addressing user needs and supporting insight communication. |
| **P5** Review, using a selection of real-world examples, the data visualisation principles and techniques used to communicate insights from large datasets.<br>**P6** Explore, using a selection of real-world examples, the dashboard and storytelling approaches used to communicate insights from large datasets. | **M3** Analyse, using a selection of real-world examples, the effectiveness of data visualisation tools and techniques in communicating insights from large datasets. | |
| **LO4** Develop a prototype of a data visualisation solution that meets the requirements of an identified scenario | | |
| **P7** Design a data visualisation solution that meets the requirements of an identified scenario.<br>**P8** Develop a prototype of the data visualisation solution that meets the requirements of an identified scenario.<br>**P9** Review the data visualisation prototype with a group of identified internal stakeholders to see how well the prototype meets the requirements of the identified scenario and to identify strengths and limitations. | **M4** Evaluate, using the results of the stakeholder feedback, the effectiveness of the data visualisation prototype in meeting requirements of an identified scenario. | |

## Recommended Resources

### Textbooks


Akhtar, N. and Khan, A. (2023) *Big Data Analytics: Principles*, Technologies and Applications. 2nd edn. Cham: Springer.

Chaudhary, K. and Alam, M. (2022) *Big Data Analytics: Applications in Business and Marketing*. Boca Raton: CRC Press.

Knaflic, S.N. (2024) *Storytelling with Data: Let's Practice!*. Hoboken, NJ: Wiley.

Provost, F. and Fawcett, T. (2013) *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. Sebastopol, CA: O'Reilly Media.

Shamsi, J.A. and Khojaye, M.A. (2023) *Big Data Systems: A 360-Degree Approach*. Boca Raton: CRC Press.

Wilke, C.O. (2020) *Fundamentals of Data Visualization: A Primer on Making Informative and Compelling Figures*. Sebastopol, CA: O'Reilly Media.


### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| algoscale.com/blog/big-data-architecture/ | Algoscale<br>What is Big Data Architecture? Components, Layers & Examples<br>(Blog) |
| aws.amazon.com/big-data/datalakes-and-analytics/ | Amazon Web Services (AWS)<br>Analytics on AWS<br>(Reference Hub) |
| learn.microsoft.com/en-us/azure/architecture/data-guide/big-data/ | Microsoft Learn<br>Big data architectures<br>(Reference) |
| www.databricks.com/glossary/data-pipeline | Databricks<br>What is Data Collection?<br>(Article) |
| www.tableau.com/learn/articles/data-visualization | Tableau<br>What Is Data Visualization? Definition, Examples, And Learning Resources<br>(Article) |

| | |
|---|---|
| www.interaction-design.org/literature/topics/data-visualization | Interaction Design Foundation<br>Data Visualization (DV)<br>(Article) |
| learn.microsoft.com/en-us/power-bi/guidance/ | Microsoft<br>Power BI guidance documentation<br>(Reference Hub) |
| www.storytellingwithdata.com/chart-guide | Storytelling with Data<br>Practical guidance on choosing appropriate charts and visuals to support insight communication.<br>(Article) |
| www.datacamp.com/tutorial/dashboard-design-tutorial | DataCamp<br>Effective Dashboard Design: Principles, Best Practices, and Examples<br>(Tutorial) |

**Journals and articles**

Chen, M., Mao, S. and Liu, Y. (2014) 'Big Data: A survey', *Mobile Networks and Applications*, 19(2), pp. 171–209. Available at: doi.org/10.1007/s11036-013-0489-0.

Dean, J. and Ghemawat, S. (2008) 'MapReduce: simplified data processing on large clusters', *Communications of the ACM*, 51(1), pp. 107–113. Available at: doi.org/10.1145/1327452.1327492.

Munzner, T. (2009) 'A Nested Model for Visualization Design and Validation', *IEEE Transactions on Visualization and Computer Graphics*, 15(6), pp. 921–928. Available at: doi.org/10.1109/TVCG.2009.111.

Sedlmair, M., Meyer, M. and Munzner, T. (2012) 'Design Study Methodology: Reflections from the Trenches and the Stacks', *IEEE Transactions on Visualization and Computer Graphics*, 18(12), pp. 2431–2440. Available at: doi.org/10.1109/TVCG.2012.213.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 510: Advanced Data Engineering*

*Unit 512: Advanced Data Analytics*

*Unit 513: AI for Data Analysis.*

# Unit 512: Advanced Data Analytics

**Unit code:** R/652/3298

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Advanced data analytics has become a critical capability for modern organisations operating in data-rich environments. Across sectors such as finance, healthcare, manufacturing, retail, digital services and the public sector, organisations increasingly rely on advanced analytical approaches to interpret complex datasets, identify patterns and support informed decision-making. This unit is part of the broader field of data analytics and data science and reflects the growing demand for practitioners who can move beyond basic reporting to deliver meaningful analytical insights using advanced methods and tools.

The aim of this unit is to develop students' ability to apply advanced analytical thinking and techniques within realistic organisational contexts. It is designed to build students' confidence in working with complex data problems, selecting appropriate analytical approaches, and applying methods in a structured, justified and compliant manner. The unit supports both centres and employers by developing students who can critically evaluate analytical options and deliver solutions that align with business, technical and regulatory requirements.

This unit covers a focused range of advanced topics that directly support this aim. Students explore the mathematical and statistical foundations that underpin advanced analytics, including inference, multivariate analysis and optimisation methods. They then review a range of advanced analytical tools and techniques, including predictive modelling, classification approaches and model validation methods. The unit also addresses the design of regulatory-compliant analytics programmes, before moving on to the practical implementation, validation and operationalisation of advanced data analytics solutions using modern tools and platforms.

On successful completion of this unit, students will be able to analyse complex analytical problems, evaluate the suitability of advanced tools and techniques, and design and implement robust data analytics solutions within defined constraints. These capabilities support progression into technical and analytical roles such as Data Analyst, Business Intelligence Analyst, Data Scientist, Analytics Consultant and Digital Transformation Specialist, as well as further study in data science, artificial intelligence or related subject areas.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the mathematical and statistical concepts used in advanced data analytics

LO2 Review a selection of advanced tools and techniques used in advanced data analytics

LO3 Design a regulatory compliant advanced data analytics workflow that meets the requirements of an identified scenario

LO4 Implement a regulatory compliant advanced data analytics workflow using modern tools for an identified data analytics scenario.

## Essential Content

LO1 **Discuss the mathematical and statistical concepts used in advanced data analytics**

*Statistical inference and probability foundations:*

Probability concepts: probability e.g. chance events, likelihood estimation; distributions e.g. normal distribution, skewed distribution; likelihood functions e.g. parameter likelihoods, model likelihoods

Sampling and estimation: sampling e.g. random sampling, stratified sampling; estimation e.g. parameter estimation, point estimation; variability e.g. data spread, value variation

Hypothesis testing measures: hypothesis testing e.g. null hypothesis, alternative hypothesis; significance e.g. significance levels, *p*-values; confidence intervals e.g. interval range, estimation bounds

Risk and uncertainty factors: risk e.g. outcome risk, decision risk; bias e.g. sampling bias, measurement bias; uncertainty e.g. data uncertainty, model uncertainty; inference e.g. statistical conclusions, analytical reasoning

Bayesian approaches: Bayesian inference principles e.g. prior beliefs, posterior updating; likelihood functions e.g. evidence weighting, probability updating.

*Multivariate and high-dimensional analysis:*

Multivariate relationships: multivariate analysis e.g. multiple variables, combined analysis; correlation e.g. positive correlation, negative correlation; covariance e.g. variable dependence, joint variation; relationships e.g. variable interaction, data association

Dimensionality techniques: dimensionality reduction e.g. feature reduction, data simplification; principal component analysis e.g. principal components, variance explanation; factor analysis e.g. latent factors, variable grouping

Clustering structures: clustering e.g. data grouping, cluster similarity; pattern recognition e.g. data patterns, trend identification; relationships e.g. similarity relationships, grouped behaviour

Modelling challenges: multicollinearity e.g. correlated variables, redundant predictors; pattern recognition e.g. repeated structures, data regularities; relationships e.g. dependent relationships, variable links.

*Optimisation and mathematical methods:*

Optimisation techniques: optimisation e.g. best solutions, performance improvement; linear optimisation e.g. linear constraints, linear objectives; non-linear optimisation e.g. curved functions, complex objectives

Objective formulation: objectives e.g. target outcomes, performance goals; loss e.g. prediction error, cost minimisation; functions e.g. mathematical functions, objective functions; constraints e.g. resource limits, condition restrictions

Computational methods: gradient-based methods e.g. gradient descent, iterative adjustment; approximation e.g. estimated solutions, simplified models; convergence e.g. stable solutions, repeated iteration

Model performance evaluation: efficiency e.g. computational efficiency, resource usage; evaluation e.g. solution assessment, outcome comparison; trade-offs e.g. accuracy trade-offs, speed trade-offs.

LO2 **Review a selection of advanced tools and techniques used in advanced data analytics**

*Regression and predictive modelling techniques:*

Regression models: regression e.g. linear relationships, variable relationships; linear regression models e.g. straight-line models, trend lines; non-linear regression models e.g. curved models, complex relationships

Prediction methods: prediction e.g. outcome prediction, value prediction; forecasting e.g. trend forecasting, future values; modelling e.g. mathematical models, data models

Variables and relationships: variables e.g. dependent variables, independent variables; relationships e.g. variable relationships, data relationships; estimation e.g. parameter estimation, coefficient estimation

Model quality factors: accuracy e.g. prediction accuracy, model accuracy; assumptions e.g. model assumptions, data assumptions; validation e.g. model validation, result checking.

*Classification and machine learning approaches:*

Classification principles: classification e.g. group assignment, category allocation; categories e.g. data categories, output classes; labels e.g. class labels, output labels

Decision structures: decision trees e.g. branching decisions, rule paths; rules e.g. decision rules, classification rules; features e.g. input features, data attributes

Pattern analysis: patterns e.g. data patterns, behaviour patterns; comparison e.g. model comparison, result comparison; interpretation e.g. result interpretation, output understanding

Application contexts: application e.g. practical use, scenario application; accuracy e.g. classification accuracy, result accuracy; comparison e.g. method comparison, outcome comparison.

*Model evaluation and validation methods:*

Evaluation processes: evaluation e.g. performance evaluation, model assessment; validation e.g. result validation, model checking; testing e.g. test datasets, test results

Performance measures: metrics e.g. performance metrics, evaluation measures; accuracy e.g. predictive accuracy, classification accuracy; error e.g. prediction error, model error

Reliability factors: reliability e.g. consistent results, dependable outputs; consistency e.g. repeatable results, stable performance; improvement e.g. model improvement, performance refinement

Analytical comparisons: comparison e.g. model comparison, metric comparison; performance e.g. system performance, model performance; Receiver Operator Characteristic (ROC) and precision–recall analysis e.g. ROC curves, precision recall.

LO3 **Design a regulatory compliant advanced data analytics workflow that meets the requirements of an identified scenario**

*Requirements and problem definition:*

Requirements: requirements e.g. business needs, data needs; objectives e.g. project goals, analysis goals; constraints e.g. time limits, resource limits

Stakeholder considerations: stakeholders e.g. data users, decision makers; scope e.g. project scope, analysis scope; priorities e.g. task priorities, outcome priorities

Assumptions and risks: assumptions e.g. data assumptions, process assumptions; risks e.g. data risks, project risks; success criteria e.g. measurable outcomes, acceptance criteria

Context and compliance: context e.g. organisational context, analytical context; compliance e.g. legal compliance, organisational compliance.

*Analytical workflow and architecture design:*

Workflow structure: workflow e.g. process flow, task sequence; stages e.g. workflow stages, process phases; sequencing e.g. task order, process sequence.

Data handling: inputs e.g. raw data, source data; outputs e.g. analysis results, reports; preparation e.g. data cleaning, data formatting

Processing and analysis: processing e.g. data processing, transformation steps; analysis e.g. analytical methods, data evaluation; documentation e.g. process documentation, analysis records

Co-ordination and efficiency: co-ordination e.g. team co-ordination, task co-ordination; efficiency e.g. process efficiency, time efficiency.

*Tool selection and compliance considerations:*

Tool evaluation: tools e.g. Excel, Structured Query Language (SQL), Python; capabilities e.g. data manipulation, analytical functions; suitability e.g. task suitability, data suitability

Programming and analysis tools: Python e.g. data analysis scripts, pandas workflows; R e.g. statistical analysis, data visualisation; Jupyter Notebook e.g. interactive analysis, documented workflows

AI and machine learning tools: TensorFlow e.g. model building, neural networks; scikit-learn e.g. classification models, regression models; AutoML platforms e.g. automated model selection, automated tuning

Visualisation and reporting platforms: Power BI e.g. dashboards, report visuals; Tableau e.g. interactive charts, visual dashboards; Google Data Studio e.g. online reports, shared dashboards

Development and management environments: VS Code e.g. script editing, code management; GitHub e.g. version control, collaborative development; Anaconda e.g. package management, environment setup

Governance and security: governance e.g. data governance, process governance; security e.g. data security, access security; policies e.g. organisational policies, usage policies

Controls and legislation: controls e.g. access controls, process controls; legislation e.g. data legislation, regulatory requirements and ongoing awareness.

LO4 **Implement a regulatory compliant advanced data analytics workflow using modern tools for an identified data analytics scenario**

*Analytics solution implementation:*

Implementation: implementation e.g. solution setup, process execution; models e.g. analytical models, predictive models; tools e.g. analytics tools, software tools

Configuration and data: configuration e.g. system configuration, model configuration; data e.g. input data, processed data; scripts e.g. code scripts, execution scripts

Process execution: processes e.g. analytical processes, workflow steps; workflows e.g. processing workflows, analysis workflows; execution e.g. task execution, process running

Outputs and adaptation: outputs e.g. analytical results, generated outputs; adaptation e.g. solution adjustment, model refinement.

*Validation and reliability assurance:*

Validation activities: validation e.g. result validation, model checking; accuracy e.g. output accuracy, result accuracy; testing e.g. test data, validation tests

Error and reliability: error e.g. prediction error, processing error; reliability e.g. reliable outputs, dependable results; consistency e.g. consistent behaviour, stable performance

Verification and review: verification e.g. result verification, process verification; results e.g. analysis results, performance results; comparison e.g. output comparison, result comparison

Assurance processes: assurance e.g. quality assurance, reliability assurance; review e.g. solution review, performance review.

*Operationalisation and automation:*

Deployment and execution: deployment e.g. solution deployment, system release; execution e.g. operational execution, routine running; scheduling e.g. task scheduling, job timing

Automation and monitoring: automation e.g. automated processing, automated tasks; monitoring e.g. system monitoring, performance tracking; maintenance e.g. routine maintenance, issue resolution

Scalability and usage: scalability e.g. system scaling, workload scaling; usage e.g. operational usage, user access; support e.g. technical support, user support.

Continuity management: continuity e.g. service continuity, operational continuity; updates e.g. system updates, model updates; maintenance e.g. ongoing maintenance, system upkeep.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Discuss the mathematical and statistical concepts used in advanced data analytics | | |
| **P1** Discuss, using a selection of mathematical examples, the advanced statistical concepts used in advanced data analytics.<br>**P2** Explain, using a selection of mathematical examples, the advanced mathematical concepts used in advanced data analytics. | **M1** Analyse, using a selection of mathematical examples, how advanced statistical and mathematical concepts can be applied in supporting complex data analytics activities. | **D1** Critically evaluate, using a selection of mathematical examples, the effectiveness of advanced statistical and mathematical approaches in addressing complex data analytics problems. |
| **LO2** Review a selection of advanced tools and techniques used in advanced data analytics | | |
| **P3** Review, using a selection of real-world examples, the modelling and machine learning approaches that can be used in advanced data analytics.<br>**P4** Explore, using a selection of real-world examples, the techniques used to evaluate and validate models used in advanced data analytics. | **M2** Assess, using a selection of real-world examples, the benefits and limitations of the application of advanced analytical tools and techniques for different advanced data analytics scenarios. | **D2** Critically evaluate, using a selection of real-world examples, the effectiveness of selected advanced tools and techniques in delivering reliable and valid analytical outcomes. |

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a regulatory compliant advanced data analytics workflow that meets the requirements of an identified scenario | | **LO3 and LO4**<br>**D3** Justify the decisions made in the design and implementation of the regulatory compliant advanced data analytics workflow in relation to comprehensively meeting the requirements of the identified data analytics scenario. |
| **P5** Design a regulatory compliant advanced data analytics workflow that meets the requirements of an identified scenario.<br>**P6** Review the regulatory compliant advanced data analytics workflow for an identified scenario with a range of identified internal stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the stakeholder feedback, the regulatory compliant advanced data analytics workflow design so that it comprehensively meets the requirements of an identified scenario. | |
| **LO4** Implement a regulatory compliant advanced data analytics workflow using modern tools for an identified data analytics scenario | | |
| **P7** Implement a regulatory compliant advanced data analytics workflow using modern tools for an identified data analytics scenario.<br>**P8** Review implementation of the advanced data analytics workflow with a range of identified stakeholders to identify strengths and areas for improvements in terms of accuracy, reliability, and automation. | **M4** Evaluate the effectiveness of the implemented regulatory compliant advanced data analytics workflow in addressing the requirements of the identified data analytics scenario. | |

## Recommended Resources

### Textbooks


Aggarwal, C.C. (2023) *Neural Networks and Deep Learning: A Textbook*. 2nd edn. Cham: Springer.

Evans, J.R. (2021) *Business Analytics*. Global edn. Harlow: Pearson Education.

Evans, J.R. (2021) *Statistics, Data Analysis, and Decision Modeling*. 5th edn. Harlow: Pearson Education.

Géron, A. (2022) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd edn. Sebastopol, CA: O'Reilly Media.

James, G. *et al*. (2021) *An Introduction to Statistical Learning*. 2nd edn. Cham: Springer.

Provost, F. and Fawcett, T. (2013) *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. Sebastopol, CA: O'Reilly Media.

Shmueli, G. *et al*. (2023) *Data Mining for Business Analytics*. 2nd edn. Hoboken, NJ: Wiley.


### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| mindfulchase.com/deep-dives/data-science-pathway/advanced-statistics-for-data-science-probability-distributions-and-hypothesis-testing.html | Mindful Chase<br>Advanced Statistics for Data Science: Probability Distributions and Hypothesis Testing<br>(Article) |
| www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/ | GeeksforGeeks<br>Principal Component Analysis (PCA)<br>(Tutorial) |
| www.statology.org/how-to-choose-and-validate-a-predictive-model-in-r-with-example/ | Statology<br>How to Choose and Validate a Predictive Model in R (With Example)<br>(Tutorial) |
| developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall | Google Developers<br>Classification: Accuracy, recall, precision, and related metrics<br>(General reference) |

| | |
|---|---|
| ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/toolkit-for-organisations-considering-using-data-analytics/toolkit-for-organisations-considering-using-data-analytics-uk-gdpr/ | Information Commissioner's Office (ICO)<br>Toolkit for organisations considering using data analytics – UK GDPR<br>(Toolkit) |
| www.ey.com/en_gl/insights/trust/gdpr-compliance-how-data-analytics-can-help | EY<br>GDPR compliance: how data analytics can help<br>(Article) |
| www.lifebit.ai/blog/regulatory-compliant-data-analytics/ | Lifebit<br>Regulatory Compliant Data Analytics: The Complete Guide for Healthcare and Life Sciences Leaders<br>(Article) |
| www.databricks.com/blog/enterprise-data-governance-complete-modern-framework | Databricks<br>Enterprise Data Governance: A Complete Modern Framework<br>(General reference) |
| scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html | scikit-learn<br>Pipeline documentation<br>(Reference Documentation) |
| www.ibm.com/think/insights/monitoring-machine-learning | IBM<br>A guide to monitoring machine learning<br>(Article) |

**Journals and articles**

Breck, E. *et al*. (2017) 'The ML test score: A rubric for ML production readiness', *IEEE Big Data*. Available at: doi.org/10.1109/BigData.2017.8258038.

Bzdok, D., Altman, N. and Krzywinski, M. (2018) 'Statistics versus machine learning', *Nature Methods*, 15(4), pp. 233–234. Available at: doi.org/10.1038/nmeth.4642.

Saltz, J.S. and Shamshurin, I. (2016) 'Big data team process methodologies: A literature review and the identification of key factors for a project's success', *Proceedings of the IEEE International Conference on Big Data*, pp. 2872–2879. Available at: doi.org/10.1109/BigData.2016.7840936.

Sculley, D. *et al*. (2015) 'Hidden Technical Debt in Machine Learning Systems', *Advances in Neural Information Processing Systems*, 28, pp. 2503–2511. Available at: proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 510: Advanced Data Engineering*

*Unit 511: Big Data Analytics and Visualisation*

*Unit 513: AI for Data Analysis.*

# Unit 513: AI for Data Analysis

**Unit code:** **T/652/3299**

**Unit level:** **5**

**Credit value:** **15**

---

## Introduction

Artificial Intelligence has become an integral part of modern data analysis across many sectors, including business, healthcare, finance and the public sector. Organisations increasingly use AI-driven techniques to process large datasets, identify patterns and support evidence-based decision-making. This unit introduces students to the role of AI within data analysis, placing it within the wider computing and analytics discipline and reflecting current industry practice.

The aim of this unit is to develop students' understanding of how AI techniques are applied to real-world data analysis scenarios. It focuses on building a clear link between foundational concepts and practical application, enabling students to work confidently with AI-based analytical approaches while operating within regulatory, ethical and organisational constraints.

This unit covers key principles of artificial intelligence used in data analysis, including machine learning categories and neural network concepts. Students explore how AI tools support data preparation, model training, inference and performance measurement. The unit then moves into the design of a regulatory-compliant AI analytics workflow for an identified scenario, before progressing to implementation and evaluation of that workflow in practice.

On successful completion of this unit, students will be able to explain and apply AI techniques for data analysis, design and implement compliant analytical workflows, and evaluate their effectiveness in supporting decision-making. These skills are relevant to a range of junior and entry-level technical roles, including Data Analyst, Junior Data Scientist, AI Analyst, Business Intelligence Analyst, and Analytics Technician, as well as providing a foundation for further study in data analytics or artificial intelligence.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain core principles of AI techniques used in data analysis

LO2 Review the use of AI applications for data analysis tasks

LO3 Design a regulatory compliant AI analytics workflow for an identified scenario

LO4 Implement a regulatory compliant AI analytics workflow for an identified scenario.

## Essential Content

LO1 **Explain core principles of AI techniques used in data analysis**

*AI fundamentals:*

Concepts: intelligence e.g. human reasoning, problem solving; analytics e.g. descriptive summaries, performance reporting; datasets e.g. spreadsheets, transaction logs

Systems: systems e.g. input–process–output, interconnected components; automation e.g. scheduled processing, rule-based workflows; optimisation e.g. faster execution, resource efficiency

Analytical outputs: prediction e.g. demand forecasting, risk estimation; classification e.g. spam detection, fault categorisation; clustering e.g. customer groups, similar-item grouping

Decision context: insights e.g. key findings, summary statements; decision-making e.g. choosing actions, prioritising options; conclusions e.g. justified outcomes, evidence-led summaries; recommendations e.g. suggested next steps, improvement actions.

*Machine learning categories:*

Learning paradigms: supervision e.g. labelled examples, known outputs; reinforcement e.g. rewards, penalties; clustering e.g. similarity measures, natural groupings

Analytical methods: classification e.g. category labels, binary outcomes; regression e.g. numeric prediction, trend lines; prediction e.g. future values, likely outcomes

Data processes: training e.g. learning patterns, updating weights; validation e.g. checking performance, tuning settings; labelling e.g. tagging records, assigning classes

Analytical results: patterns e.g. repeated behaviour, common sequences; trends e.g. upward movement, seasonal change; anomalies e.g. outliers, unusual spikes; correlations e.g. linked variables, positive relationship.

*Neural network concepts:*

Model structure: neurons e.g. processing units, weighted inputs; layers e.g. input layer, hidden layer; architectures e.g. simple networks, deep networks

Model behaviour: activations e.g. activation functions, output signals; training e.g. backpropagation, learning rate; features e.g. input variables, extracted attributes

Model performance: overfitting e.g. memorising training data, poor generalisation; underfitting e.g. oversimplified model, weak accuracy; interpretability e.g. explainable outputs, understandable decisions

Network forms: feedforward e.g. one-way flow, layered processing; convolution e.g. filters, feature maps.

LO2 **Review the use of AI applications for data analysis tasks**

*Data preparation processes:*

Data quality: cleaning e.g. error removal, duplicate handling; missing-values e.g. null replacement, record omission; noise e.g. outlier smoothing, irrelevant data removal;

Feature handling: extraction e.g. attribute derivation, variable creation; selection e.g. relevance filtering, importance ranking; transformation e.g. scaling, normalisation

Data representation: encoding e.g. numeric encoding, categorical encoding; transformation e.g. data reshaping, format conversion; datasets e.g. training datasets, analysis datasets

Real-world examples: cleaning e.g. AWS Glue DataBrew, Talend Data Preparation; selection e.g. scikit-learn, Amazon SageMaker; transformation e.g. Apache Spark, Google Dataprep; extraction e.g. DataRobot, H2O.ai.

*Model training and inference:*

Model datasets: training e.g. learning sets, labelled data; testing e.g. unseen data, evaluation sets; validation e.g. model tuning, parameter adjustment

Inference process: inference e.g. model execution, output generation; prediction e.g. future estimation, outcome calculation; outputs e.g. scores, classifications

Operational scale: scalability e.g. increased data volume, performance growth; deployment e.g. production release, system integration; systems e.g. analytics platforms, processing environments

Real-world examples: training e.g. Amazon SageMaker, Azure Machine Learning; prediction e.g. Amazon SageMaker, IBM Watson Studio; deployment e.g. Azure Machine Learning, Amazon SageMaker.

*Performance measurement and decision support:*

Evaluation metrics: accuracy e.g. correct predictions, overall correctness; precision e.g. positive prediction reliability, relevance; recall e.g. detection rate, sensitivity; F1 e.g. balanced score, combined metric

Confusion analysis: confusion matrices e.g. true positives, false negatives; classification e.g. predicted categories, actual labels; prediction e.g. expected outputs, model results

Analytical outputs: patterns e.g. recurring results, common behaviours; trends e.g. gradual change, directional movement; anomalies e.g. unusual results, unexpected values; correlations e.g. variable relationships, linked behaviour

Decision systems: effectiveness e.g. decision quality, outcome suitability; integration e.g. system embedding, workflow alignment; decision-support e.g. recommendation tools, insight dashboards; limitations e.g. data constraints, model assumptions

Real-world examples: precision e.g. scikit-learn, Azure Machine Learning; classification e.g. Google Vertex AI, Amazon SageMaker; anomalies e.g. Azure Anomaly Detector, Amazon Lookout for Metrics; decision-support e.g. Power BI, Tableau.

LO3 **Design a regulatory compliant AI analytics workflow for an identified scenario**

*Scenario and analytical requirements:*

Business context: organisation e.g. company structure, operational purpose; sector e.g. industry type, market area; data stakeholders e.g. data owners, data users

Analytical objectives: analytical questions e.g. problem statements, investigation focus; decision requirements e.g. choice criteria, action needs; success criteria e.g. measurable outcomes, performance targets

Constraints: data availability e.g. limited records, data access; data quality e.g. incomplete data, inconsistent values; resource limitations e.g. time constraints, computing capacity.

*Data and feature design:*

Data preparation planning: data collection e.g. data gathering methods, source identification; data cleaning e.g. error correction, duplicate removal; data preprocessing e.g. formatting data, scaling values

Feature engineering: feature extraction e.g. derived variables, calculated fields; feature selection e.g. relevant attributes, importance ranking; feature transformation e.g. normalisation, encoding

Data governance: data quality e.g. accuracy checks, consistency control; bias e.g. unbalanced data, representation issues; provenance e.g. data origin, data history; ownership e.g. responsibility definition, access rights.

*Design decisions and analytical approach:*

Analytical approach: predictive analytics e.g. forecasting outcomes, estimating values; descriptive analytics e.g. summarising data, reporting patterns; prescriptive analytics e.g. recommended actions, optimisation guidance

Technique selection rationale: model suitability e.g. task alignment, complexity match; data characteristics e.g. data volume, variable type; analytical goals e.g. insight generation, decision support

Analytical priorities: accuracy e.g. correct outputs, error reduction; robustness e.g. stable performance, noise tolerance; reliability e.g. consistent results, repeatable outcomes.

*Governance, ethics and evaluation planning:*

Ethical considerations: fairness e.g. non-discrimination, equal treatment; bias mitigation e.g. data balancing, bias reduction; transparency e.g. clear justification, accountable decisions

Legal requirements: data protection e.g. personal data handling, security controls; privacy e.g. confidentiality protection, consent management; compliance e.g. regulatory adherence and updates, policy alignment

Evaluation planning: evaluation metrics e.g. success measures, performance indicators; validation strategy e.g. testing approach, assessment method; success thresholds e.g. acceptance criteria, minimum standards

Risk considerations: assumptions e.g. data expectations, model conditions; limitations e.g. known constraints, reduced coverage; analytical risks e.g. incorrect predictions, misleading outcomes.

LO4 **Implement a regulatory compliant AI analytics workflow for an identified scenario**

*Workflow implementation processes:*

Analytics pipeline execution: data ingestion e.g. data loading, source import; processing stages e.g. transformation steps, analysis phases; workflow sequencing e.g. task order, dependency flow

Analytical system integration: data sources e.g. databases, data feeds; analytics platforms e.g. cloud services, analytics tools; model services e.g. deployed models, prediction services

Operational analytics setup: output monitoring e.g. result tracking, performance observation; result logging e.g. execution records, output storage.

*Model training and inference execution:*

Model training: training runs e.g. repeated execution, experiment cycles; parameter optimisation e.g. tuning values, configuration adjustment

Model inference: prediction generation e.g. output creation, value estimation; classification execution e.g. category assignment, label generation

Analytical outputs: results generation e.g. model outputs, calculated scores; insight production e.g. interpreted results, actionable findings

Automation processes: scheduled runs e.g. timed execution, recurring analysis.

*Performance measurement and validation:*

Evaluation metrics: accuracy e.g. correct outcomes, prediction correctness; precision e.g. relevant results, positive reliability; recall e.g. detection success, sensitivity rate; F1 e.g. balanced scoring, combined measure

Validation techniques: testing datasets e.g. unseen data, evaluation samples; cross-validation e.g. data folds, repeated testing

Performance monitoring: throughput e.g. processing rate, data volume handling; latency e.g. response time, execution delay

Result interpretation: analytical outputs e.g. model results, generated insights; patterns e.g. frequent outcomes, consistent behaviour.

*Operational review and improvement:*

Effectiveness review: analytical objectives e.g. goal achievement, requirement fulfilment; decision support value e.g. improved decisions, actionable insight; limitations e.g. performance gaps, known weaknesses

Analytical issues: data drift e.g. changing input data, shifting distributions; model degradation e.g. reduced accuracy, performance decline; bias emergence e.g. skewed results, unfair outcomes

Improvement actions: model retraining e.g. updated datasets, revised learning; feature adjustment e.g. variable refinement, feature removal

Sustainability considerations: maintainability of analytics e.g. ease of updates, long-term support.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain core principles of AI techniques used in data analysis</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of AI techniques and applications for data analysis contexts within identified organisational contexts.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the core principles of AI used in data analysis.<br><br>P2 Discuss, using a selection of real-world examples, how machine learning and neural network concepts can be applied to data analysis.</td><td>M1 Analyse, using a selection of real-world examples, the application of different AI techniques in a range of organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Review the use of AI applications for data analysis tasks</td></tr>
<tr><td>P3 Review, using a selection of real-world examples, the use of AI applications for data analysis tasks such as data preparation and model training.<br><br>P4 Explore, using a selection of real-world examples, the use of AI applications for data analysis tasks such as performance measurement and decision support.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of identified AI applications in supporting data analysis within organisational contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a regulatory compliant AI analytics workflow for an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and implementation of the regulatory compliant AI analytics workflow in relation to comprehensively meeting the requirements of the identified scenario. |
| **P5** Design a regulatory compliant AI analytics workflow for an identified scenario.<br>**P6** Review the design of a regulatory compliant AI analytics workflow with a range of identified stakeholders to identify strengths and limitations and how well it meets the requirements of an identified scenario. | **M3** Optimise, using the results of the stakeholder feedback, the design of a regulatory compliant AI analytics workflow so that it fully meets the requirements of an identified scenario. | |
| **LO4** Implement a regulatory compliant AI analytics workflow for an identified scenario | | |
| **P7** Implement a regulatory compliant AI analytics workflow for an identified scenario.<br>**P8** Review the implementation of the regulatory compliant AI analytics workflow with a range of identified stakeholders to determine workflow effectiveness and identify any issues and areas for improvement. | **M4** Evaluate the effectiveness of the implemented regulatory compliant AI analytics workflow in addressing the requirements of the identified scenario. | |

## Recommended Resources

### Textbooks

Eryurek, E. *et al*. (2021) *Data Governance: The Definitive Guide*. Sebastopol, CA: O'Reilly Media.

Kelleher, J.D., Mac Namee, B. and D'Arcy, A. (2021) *Fundamentals of Machine Learning for Predictive Data Analytics*. 2nd edn. Cambridge, MA: MIT Press.

Provost, F. and Fawcett, T. (2013) *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. Sebastopol, CA: O'Reilly Media.

Russell, S. and Norvig, P. (2021) *Artificial Intelligence: A Modern Approach*. 4th edn. Harlow: Pearson.

Sharda, R., Delen, D. and Turban, E. (2024) *Business Intelligence, Analytics, Data Science, and AI*. 5th edn. Harlow: Pearson Education.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.datacamp.com/tracks/ai-fundamentals | DataCamp<br>AI Fundamentals<br>(General reference) |
| www.geeksforgeeks.org/data-analysis/data-analysis-tutorial/ | GeeksforGeeks<br>Data Analytics Tutorial<br>(Tutorial) |
| www.pecan.ai/blog/data-preparation-for-machine-learning/ | Pecan AI<br>Data Preparation for Machine Learning: The Ultimate Guide to Doing It Right<br>(Blog) |
| www.datacamp.com/blog/data-preprocessing | DataCamp<br>Data Preprocessing Guide: A Complete Guide with Python Examples<br>(Tutorial) |
| ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/guidance-on-ai-and-data-protection/ | Information Commissioner's Office (ICO)<br>Guidance on AI and data protection<br>(General reference) |

| | |
|---|---|
| www.oecd.org/en/topics/ai-principles.html | OECD<br>AI principles<br>(General reference) |

**Journals and articles**

Brasse, J. *et al*. (2023) 'Explainable artificial intelligence in information systems: A review of the status quo and future research directions', *Electronic Markets*, 33, Article 26. Available at: doi.org/10.1007/s12525-023-00644-5.

Jobin, A., Ienca, M. and Vayena, E. (2019) 'The global landscape of AI ethics guidelines', *Nature Machine Intelligence*, 1, pp. 389–399. Available at: doi.org/10.1038/s42256-019-0088-2.

Rahmani, A.M. *et al*. (2021) 'Artificial intelligence approaches and mechanisms for big data analytics: a systematic study', *PeerJ Computer Science*, 7, e488. Available at: doi.org/10.7717/peerj-cs.488.

Schmitt, M. (2023) 'Automated machine learning: AI-driven decision making in business analytics', *Intelligent Systems with Applications*, 18, 200188. Available at: doi.org/10.1016/j.iswa.2023.200188.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 510: Advanced Data Engineering*

*Unit 511: Big Data Analytics and Visualisation*

*Unit 512: Advanced Data Analytics.*

 © Pearson Education Limited 2026

# Unit 514: Quality of Service (QoS)

**Unit code:** H/652/3300

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Modern organisations rely on computer networks to support real-time communication, cloud services and business-critical applications. As network traffic becomes more complex and demand continues to increase, maintaining reliable and predictable service performance has become a key challenge for networked systems. Quality of Service (QoS) provides the principles and mechanisms used to manage traffic behaviour and ensure that essential services receive appropriate levels of performance within enterprise, service provider and cloud-based network environments.

The aim of this unit is to develop students' understanding of how QoS is used to manage, control and optimise network traffic. The unit focuses on the reasons for implementing QoS, the performance challenges caused by congestion and competing traffic demands, and the role QoS plays in supporting organisational objectives and service level expectations.

The unit covers the core principles that underpin QoS, including latency, jitter, bandwidth, packet loss and traffic prioritisation. Students will examine and compare commonly used QoS mechanisms and algorithms, such as queuing, scheduling, traffic shaping, policing and Differentiated Services (DiffServ). Building on this knowledge, students will implement QoS configurations for an identified network and review network performance using appropriate monitoring, analysis and evaluation techniques.

On successful completion of this unit, students will be able to design, implement and evaluate QoS strategies within real-world networked environments. They will be equipped with skills relevant to roles such as Network Engineer, Network Operations Technician, Infrastructure Engineer, Systems Administrator and Network Performance Analyst, and will be prepared for progression into more advanced study or employment within networking and infrastructure management.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the principles of Quality of Service

LO2 Compare QoS mechanisms and algorithms used to manage and control network traffic

LO3 Implement QoS configurations to optimise the performance of an identified network

LO4 Review the network performance using a range of QoS techniques.

## Essential Content

LO1 **Discuss the principles of Quality of Service**

*Core QoS principles:*

Network performance measures: latency e.g. transmission delay, data delivery time, signal travel, jitter e.g. delay variation, timing inconsistency, packet spacing, bandwidth e.g. data capacity, link speed, transmission rate, throughput e.g. data transfer rate, successful delivery rate, network efficiency, packet loss e.g. dropped packets, transmission errors, congestion loss

Traffic management concepts: congestion e.g. network overload, traffic bottlenecks, queue buildup, prioritisation e.g. traffic importance, service ranking, access preference, traffic sensitivity e.g. delay tolerance, loss tolerance, application behaviour

Service delivery concepts: service differentiation e.g. service classes, traffic separation, performance levels.

*Performance impact on networked services:*

Traffic and application types: Real-time traffic requirements e.g. low delay needs, time-critical delivery, continuous flow, delay-sensitive applications e.g. interactive services, response-dependent software, live communication, bandwidth-intensive services e.g. high data usage, large file transfer, media distribution

Service expectations: reliability expectations e.g. consistent performance, dependable service, fault resistance, service continuity e.g. uninterrupted operation, session persistence, availability maintenance, quality perception e.g. perceived performance, user satisfaction, service impression

User-focused outcomes: user experience e.g. ease of use, perceived quality, service satisfaction, Voice over IP (VoIP) e.g. voice calls, audio communication, internet telephony, video conferencing e.g. virtual meetings, live video interaction, remote collaboration, streaming services e.g. media playback, online video delivery, content streaming.

*Traffic classification and prioritisation concepts:*

Traffic identification: traffic classes e.g. application groups, service categories, traffic types, application awareness e.g. service recognition, application identification, traffic purpose, protocol identification e.g. protocol types, communication standards, data formats

Priority handling: priority levels e.g. importance ranking, service precedence, access order, fairness e.g. equal access, balanced allocation, shared resources

Traffic treatment models: best-effort traffic e.g. unmanaged delivery, no guarantees, default handling, managed traffic e.g. controlled delivery, assigned priority, performance management, business-critical traffic handling e.g. essential services, operational importance, critical application support.

LO2 **Compare QoS mechanisms and algorithms used to manage and control network traffic**

*Queuing mechanisms:*

Queue models: first-in-first-out (FIFO) queuing e.g. arrival order, first processing, simple sequencing, priority queuing e.g. importance ordering, urgent traffic, service precedence, weighted fair queuing e.g. proportional allocation, weight assignment, balanced servicing, class-based queuing e.g. traffic grouping, class separation, policy control

Queue behaviour characteristics: congestion handling characteristics e.g. queue buildup, packet delay, buffer overflow, resource allocation behaviour e.g. bandwidth distribution, processing allocation, queue fairness.

*Scheduling techniques:*

Scheduling models: packet scheduling models e.g. transmission ordering, scheduling logic, processing sequence, priority scheduling e.g. urgent packet handling, precedence control, service ranking, round-robin approaches e.g. cyclic servicing, equal turns, sequential processing

Scheduling optimisation: weighted scheduling e.g. assigned importance, proportional service, controlled distribution, delay management e.g. wait reduction, timing control, response improvement, fairness considerations e.g. equal opportunity, balanced access, shared use.

*Traffic shaping and policing:*

Traffic control principles: traffic shaping principles e.g. traffic smoothing, rate control, burst management, traffic policing principles e.g. traffic enforcement, rule compliance, rate restriction

Traffic regulation methods: rate limiting e.g. maximum transmission rate, speed control, bandwidth caps, burst control e.g. traffic bursts, buffer allowance, temporary excess, compliance enforcement e.g. policy adherence, rule application, traffic regulation

Impact on performance: impact on throughput e.g. transmission rate effects, data flow changes, delivery consistency, impact on latency e.g. delay changes, response time, queue delay.

*Differentiated Services (DiffServ):*

DiffServ framework: DiffServ architecture e.g. network model, service structure, traffic framework, Differentiated Services Code Point (DSCP marking) e.g. traffic tagging, packet identification, code points, traffic classes e.g. service groups, class types, priority groupings

Forwarding behaviour: per-hop behaviours e.g. node actions, forwarding treatment, packet handling, scalability considerations e.g. network growth, traffic volume, performance maintenance

Deployment contexts: enterprise use e.g. organisational networks, internal services, business traffic, service provider use e.g. public networks, customer traffic, large-scale deployment.

LO3 **Implement QoS configurations to optimise the performance of an identified network**

*QoS configuration models:*

Configuration approaches: policy-based QoS e.g. rule-based control, policy definition, traffic rules, class-map e.g. traffic classification, match conditions, traffic grouping, policy-map concepts e.g. action definition, policy behaviour, traffic treatment, interface-level configuration e.g. device interfaces, port settings, access points

QoS consistency and compatibility: end-to-end QoS consistency e.g. consistent policies, path continuity, service alignment, device interoperability e.g. equipment compatibility, vendor interaction, system co-ordination.

*Policy design and optimisation:*

Traffic policy design: traffic classification rules e.g. traffic identification, rule matching, service recognition, prioritisation policies e.g. traffic importance, service ranking, access preference, bandwidth allocation strategies e.g. bandwidth division, capacity assignment, resource sharing

Performance optimisation: Congestion avoidance settings e.g. queue management, overload prevention, traffic control, optimisation objectives e.g. performance improvement, efficiency goals, service enhancement.

*Service level agreements (SLAs):*

Performance targets: latency targets e.g. acceptable delay, timing limits, service response, jitter thresholds e.g. variation limits, delay stability, timing control, packet loss tolerance e.g. acceptable loss, delivery reliability, error allowance

Service assurance: availability metrics e.g. uptime measurement, service availability, operational time, performance guarantees e.g. service commitments, agreed performance, delivery standards, compliance measurement e.g. performance checking, target verification, service monitoring.

*Network tuning considerations:*

Performance assessment: performance baselines e.g. normal operation levels, reference performance, standard conditions, capacity planning e.g. growth forecasting, resource planning, demand estimation;

Configuration refinement: configuration refinement e.g. adjustment processes, parameter changes, policy updates, interoperability constraints e.g. compatibility limits, system restrictions, device interaction, hardware and software limitations e.g. equipment capability, processing limits, system resources.

LO4 **Review the network performance using a range of QoS techniques**

*QoS monitoring and analytics:*

Traffic and utilisation metrics: performance metrics e.g. key indicators, performance values, measurement points, traffic statistics e.g. usage data, packet counts, flow records

Delay, loss and variation analysis: latency measurement e.g. delay monitoring, timing analysis, response tracking, jitter analysis e.g. variation assessment, consistency checking, timing comparison, packet loss reporting e.g. loss records, dropped packets, transmission errors

Trend and capacity monitoring: utilisation trends e.g. capacity usage, traffic growth, network load, performance metrics e.g. key indicators, performance values, measurement points.

*Diagnostics and performance evaluation:*

Congestion and capacity diagnosis: bottleneck identification e.g. congestion points, capacity limits, constrained links, congestion analysis e.g. traffic overload, queue buildup, packet delay

Traffic distribution and balance analysis: traffic imbalance detection e.g. uneven distribution, priority dominance, resource skew

Configuration and fault identification: misconfiguration indicators e.g. incorrect settings, policy errors, configuration issues, root-cause indicators e.g. fault sources, underlying causes, problem origins.

*Reporting and review methods:*

Performance and compliance reporting: performance reports e.g. summary documents, metric reporting, performance records, compliance reporting e.g. policy adherence, performance compliance, service reporting

Service agreement and baseline review: SLA validation e.g. agreement checking, target confirmation, service verification, comparison against baseline expectations e.g. performance comparison, reference levels, expected standards;

Trend and comparative analysis: trend analysis e.g. pattern identification, long-term changes, usage direction.

*Continuous performance improvement:*

Configuration and policy improvement: configuration refinement e.g. parameter adjustment, setting updates, policy tuning, policy adjustment e.g. rule modification, priority changes, configuration updates

Feedback and optimisation cycles: feedback loops e.g. performance feedback, monitoring results, corrective input

Adaptation and scalability planning: scalability considerations e.g. growth planning, capacity expansion, traffic increase, evolving traffic demands e.g. usage changes, application growth, service expansion.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Discuss the principles of Quality of Service (QoS)</td><td rowspan="4">LO1 and LO2<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of QoS principles, mechanisms and algorithms in managing network performance for an identified system or service.</td></tr>
<tr><td>P1 Discuss, using a selection of real-world examples, the principles of QoS and traffic classification in a range of identified network systems or contexts.<br><br>P2 Describe, using a selection of real-world examples, the impact of performance on a range of identified network services.</td><td>M1 Analyse, using a selection of real-world examples, the impact of QoS principles on the network performance of a range of identified network systems or services.</td></tr>
<tr><td colspan="2">LO2 Compare QoS mechanisms and algorithms used to manage and control network traffic</td></tr>
<tr><td>P3 Compare, using a selection of real-world examples, the QoS queuing and scheduling algorithms used to manage and control network traffic in a range of identified network systems or contexts.<br><br>P4 Compare, using a selection of real-world examples, the QoS traffic management mechanisms and DiffServ architectures used to manage and control network traffic in a range of identified network systems or contexts.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of QoS mechanisms and algorithms in controlling network traffic in a range of identified network systems or contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Implement QoS configurations to optimise the performance of an identified network | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and implementation of the QoS configurations in the context of how effectively it meets the requirements of the defined service level agreement (SLA). |
| **P5** Design a QoS configuration to optimise the performance of an identified network that meets a defined service level agreement (SLA).<br>**P6** Implement QoS configurations to optimise the performance of an identified network. | **M3** Assess how closely the QoS design agrees with the implemented QoS configuration to identify strengths and areas for improvement. | |
| **LO4** Review the network performance using a range of QoS techniques | | |
| **P7** Review the performance of the optimised network by gathering data using a range of QoS techniques.<br>**P8** Review the optimised network with a range of internal stakeholders to determine how well the network meets the terms of the SLA. | **M4** Evaluate, using results of the network performance data and the stakeholder review, the effectiveness of the implemented QoS design in meeting the terms of the agreed SLA. | |

## Recommended Resources

### Textbooks

Almomani, I. and Gupta, B.B. (2023) *Quality of Service and Quality of Experience in Networked Systems*. Cham: Springer.

Kliazovich, D., Bouvry, P. and Khan, S.U. (2024) *Energy-Efficient and QoS-Aware Networking*. Hoboken, NJ: Wiley.

Stallings, W. (2015) *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*. Boston, MA: Pearson.

Tanenbaum, A.S. and Wetherall, D.J. (2024) *Computer Networks*. 6th edn. Harlow: Pearson Education.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.fortinet.com/resources/cyberglossary/qos-quality-of-service | Fortinet<br>What Is Quality Of Service (QoS) In Networking?<br>(Article) |
| www.splunk.com/en_us/blog/learn/qos-quality-of-service.html | Splunk<br>QoS Explained: Quality of Service in Networks<br>(Article) |
| networklessons.com/quality-of-service/introduction-qos-quality-service | NetworkLessons<br>Introduction to QoS (Quality of Service)<br>(Tutorial) |
| study-ccna.com/qos-queues-queuing/ | Study CCNA<br>Quality of Service (QoS) Queues and Queuing Explained.<br>(Tutorial) |
| scienceinsights.org/what-is-diffserv-differentiated-services-explained/ | ScienceInsights<br>What Is DiffServ? Differentiated Services Explained<br>(Article) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html | Cisco<br>Compare Traffic Policy and Traffic Shape to Limit Bandwidth<br>(Reference) |
| study-ccna.com/qos-traffic-shaping-policing/ | Study CCNA<br>Quality of Service (QoS) Traffic Shaping and Policing.<br>(Article) |
| docs.fortinet.com/document/fortigate/7.6.4/administration-guide/297431/traffic-shaping | Fortinet Documentation<br>Traffic shaping.<br>(Reference) |
| www.kentik.com/kentipedia/understanding-latency-packet-loss-and-jitter-in-networking/ | Kentik<br>Understanding Latency, Packet Loss, and Jitter in Network Performance<br>(Reference) |

**Journals and articles**

Isaev, R., Mirjasurova, K. and Jurabekov, A. (2024) 'About one model for providing QoS in multimedia communication networks', *AIP Conference Proceedings*, 3244(1). Available at: doi.org/10.1063/5.0241607.

Strzęciwilk, D. (2025) 'Differentiated service quality analysis based on QoS traffic prioritisation', *International Journal of Electronics and Telecommunications*, 71(1), pp. 285–292. Available at: doi.org/10.24425/ijet.2025.153572.

Tandi, M.R. and Shrirao, N.M. (2025) 'Comparative Analysis of Network Measurement Tools for QoS Evaluation in 5G Communication Environments', *International Journal of Communication and Computer Technologies*, 13(2). Available at: doi.org/10.31838/IJCCTS.13.02.07.

Toghyani, M., Khorsand, R. and Khaksar, H. (2025) 'QoS-SLA-aware Optimization Framework for IoT-Service Placement in Integrated Fog-Cloud Computing', *Journal of Grid Computing*, 23. Available at: doi.org/10.1007/s10723-024-09787-x.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 515: Secure System Design*

*Unit 516: Internet of Things.*

# Unit 515: Secure System Design

**Unit code:** J/652/3301

**Unit level:** 5

**Credit value:** 15

---

## Introduction

The design of secure systems is a fundamental requirement within modern computing, underpinning the safe operation of digital services, platforms and infrastructures across both public and private sectors. As organisations increasingly depend on interconnected networks, cloud-based services and data-driven systems, weaknesses introduced at the design stage can lead to significant security, operational and regulatory risks. Secure system design therefore focuses on embedding protection measures early, ensuring systems are resilient, trustworthy and aligned with recognised security principles and standards.

The aim of this unit is to develop students' ability to design and evaluate secure systems using structured, industry-recognised approaches. Students are encouraged to adopt a security-by-design mindset, enabling them to identify threats, assess risks and apply appropriate controls as part of the system development process. This approach reflects current industry practice, where security considerations are treated as a core design requirement rather than an afterthought.

The unit covers a coherent set of topics that support the development of secure system design skills. Students explore the principles of secure system design across the secure development life cycle, including trust relationships, threat surfaces and threat-focused design approaches. They carry out structured threat and risk assessments using recognised frameworks before producing secure system designs that incorporate cryptography, access control and secure communications. The unit also examines the implementation and testing of security controls to ensure that system designs effectively address identified risks.

On successful completion of this unit, students will be able to analyse security risks, design secure system architectures and evaluate the effectiveness of implemented controls within defined scenarios. These skills are directly applicable to roles such as Secure Systems Designer, Cybersecurity Analyst, Security Architect, DevSecOps Engineer and Information Security Consultant, as well as supporting progression to further study and professional certification pathways within the cybersecurity field.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the principles of secure system design across the secure development life cycle

LO2 Develop a structured threat and risk assessment for an identified scenario

LO3 Design a secure system according to an identified scenario using the principles of security by design

LO4 Implement a range of security controls and technologies to meet the system design requirements of the identified scenario.

## Essential Content

LO1 **Discuss the principles of secure system design across the secure development life cycle**

*Secure system design principles:*

Core security principles: security by design e.g. embedded protection, early risk consideration; least privilege e.g. restricted permissions, controlled access; defence-in-depth e.g. layered security, multiple safeguards

Design safeguards and controls: secure defaults e.g. default deny settings, baseline configurations; fail-safe design e.g. secure failure states, controlled degradation; privacy by design e.g. data protection focus, lawful processing; separation of duties e.g. role separation, responsibility segregation; minimisation of attack surface e.g. reduced interfaces, limited exposure.

*Security models and trust concepts:*

Trust structure elements: security models e.g. access control models, integrity models; trust boundaries e.g. system borders, network boundaries; trusted and untrusted zones e.g. internal environments, external environments

System relationship concepts: system components e.g. services, modules; data flows e.g. information movement, transaction paths; privilege boundaries e.g. access transitions, control boundaries; dependency relationships e.g. service reliance, external dependencies.

*Threat-focused design approaches:*

Threat identification concepts: threat surfaces e.g. exposed entry points, interfaces; threat modelling concepts e.g. structured analysis, design-stage assessment; STRIDE categories e.g. spoofing, tampering; attacker perspectives e.g. internal attackers, external attackers

Risk awareness techniques: misuse cases e.g. unintended system use, abuse scenarios; design-time risk awareness e.g. proactive identification, early mitigation.

*Secure development life cycle (SDL):*

Life cycle structure and stages: SDL e.g. security-integrated life cycle, staged development; secure requirements e.g. security constraints, protection objectives; design review points e.g. formal reviews, peer assessment

Security integration across life cycle: security considerations across life cycle stages e.g. design focus, build focus; integration of security into development processes e.g. secure workflows, development alignment.

LO2 **Develop a structured threat and risk assessment for an identified scenario**

*Assets and threat identification:*

Asset classification: information assets e.g. data records, sensitive information; system assets e.g. hardware resources, software components; users e.g. system users, administrator roles

Threat source identification: data sensitivity e.g. confidentiality levels, classification schemes; threat actors e.g. malicious insiders, external attackers; threat sources e.g. human threats, environmental threats; threat intelligence inputs e.g. advisories, threat reports.

*Vulnerabilities and attack techniques:*

System weaknesses: vulnerabilities e.g. software flaws, design weaknesses; weaknesses e.g. lack of controls, poor configurations; misconfigurations e.g. insecure settings, default credentials

Attack methodology analysis: attack patterns e.g. common exploits, repeated techniques; MITRE ATT&CK framework e.g. tactics, techniques; cyber kill chain concepts e.g. reconnaissance stages, exploitation stages.

*Risk analysis concepts:*

Risk evaluation factors: likelihood e.g. probability estimation, threat frequency; impact e.g. business disruption, data loss; risk levels e.g. low risk, high risk

Risk assessment methods: risk matrices e.g. likelihood-impact grids, scoring models; qualitative risk assessment e.g. descriptive scales, expert judgement; quantitative risk assessment e.g. numerical scoring, statistical measures; prioritisation of risks e.g. ranking risks, treatment order.

*Risk treatment and compliance context:*

Risk response activities: risk treatment options e.g. mitigation actions, control selection; risk registers e.g. documented risks, tracking records; acceptance e.g. risk tolerance, formal approval; mitigation e.g. control implementation, risk reduction; transfer e.g. insurance arrangements, third-party responsibility; avoidance e.g. system redesign, activity removal

Standards and frameworks: compliance alignment e.g. regulatory requirements, internal standards; ISO/IEC 27001 e.g. information security controls, information security management system (ISMS) requirements; NIST Cybersecurity Framework (CSF) e.g. identify function, protect function; knowledge currency, enhancements, professional practice.

LO3 **Design a secure system according to an identified scenario using the principles of security by design**

*Secure system architecture:*

Architecture representation methods: system architecture models e.g. logical architecture, physical architecture; layered architectures e.g. presentation layer, application layer; secure design patterns e.g. Model–View–Controller, broker pattern

Structural protection techniques: component isolation e.g. isolated services, segregated components; trust zone segmentation e.g. trusted internal zones, demilitarised zones, untrusted external zones

Security-driven architectural decisions: system architecture models e.g. high-level system views, security-focused structures; secure design patterns e.g. defence-in-depth pattern, least privilege pattern

Boundary and interaction control: layered architectures e.g. tier separation, responsibility separation; trust zone segmentation e.g. demilitarised zones, trusted boundaries; component isolation e.g. sandboxed components, minimal interaction.

*Cryptography and data protection:*

Cryptographic foundations: cryptography concepts e.g. encryption principles, cryptographic functions; symmetric encryption e.g. shared keys, block ciphers; asymmetric encryption e.g. public keys, private keys

Data protection mechanisms: key management e.g. key storage, key rotation; data confidentiality e.g. access restriction, data secrecy; data integrity e.g. tamper detection, validation mechanisms.

*Access control and identity concepts:*

Identity and authentication methods: authentication e.g. credential verification, identity proofing; identity and access management (IAM) e.g. identity life cycle, access provisioning

Authorisation structures: authorisation e.g. access decisions, permission assignment; role-based access control e.g. role definition, permission grouping; least privilege enforcement e.g. restricted access, minimal permissions.

*Secure communication and system hardening:*

Secure communication methods: secure protocols e.g. communication standards, protected exchanges; encrypted communications e.g. protected data transfer, secure channels

System protection practices: system hardening e.g. removal of weaknesses, reduced functionality; secure configuration baselines e.g. approved settings, standard configurations; reduction of attack surface e.g. limited services, reduced exposure.

LO4 **Implement a range of security controls and technologies to meet the system design requirements of the identified scenario**

*Authentication and access technologies:*

Identity verification mechanisms: authentication mechanisms e.g. credential verification, identity validation; multi-factor authentication (MFA) e.g. knowledge factors, possession factors; identity providers e.g. centralised identity services, federated identity services

Access enforcement technologies: access enforcement technologies e.g. access gateways, policy enforcement points; identity providers e.g. directory services, identity management platforms.

*Authorisation and control enforcement:*

Permission management approaches: authorisation mechanisms e.g. access approval, permission assignment; privilege management e.g. elevated access control, privilege monitoring; access review processes e.g. periodic reviews, permission audits

Policy-based control mechanisms: policy enforcement e.g. access rules, security policies; access review processes e.g. compliance checks, governance reviews; privilege management e.g. role adjustment, access revocation.

*Cryptographic controls and technologies:*

Encryption and hashing mechanisms: encryption standards e.g. AES, Rivest–Shamir–Adleman (RSA); hashing algorithms e.g. SHA-256, message digests; digital certificates e.g. identity certificates, trust validation

Secure communication technologies: Transport Layer Security (TLS) e.g. encrypted sessions, secure channels; digital certificates e.g. certificate authorities, certificate chains; encryption standards e.g. protocol encryption, data protection.

*Monitoring and protective controls:*

Protective control types: security controls e.g. preventive controls, detective controls; technical safeguards e.g. system protections, automated controls; system protection mechanisms e.g. intrusion prevention, boundary protection

Control alignment and assurance: alignment with design requirements e.g. architectural consistency, requirement traceability; security controls e.g. layered protection, control coverage; technical safeguards e.g. enforcement mechanisms, resilience measures.

*Secure system testing:*

Test planning and design: security test planning e.g. test scope definition, test objectives; test case design e.g. positive test cases, negative test cases; traceability testing e.g. requirements mapping, control traceability

Access and cryptography testing: authentication testing e.g. credential validation, factor enforcement; authorisation testing e.g. permission validation, role enforcement; cryptographic testing e.g. encryption verification, key handling

Control and assurance testing: control effectiveness testing e.g. mitigation validation, control coverage; configuration validation e.g. baseline compliance, secure settings; vulnerability testing e.g. known weakness identification, exposure analysis; penetration testing e.g. exploit simulation.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Discuss the principles of secure system design across the secure development life cycle | | |
| **P1** Describe, using a selection of real-world examples, the principles of secure system design including security models and threat focused design approaches.<br><br>**P2** Discuss, using a selection of real-world examples, the principles of secure system design across the secure development life cycle. | **M1** Analyse, using a selection of real-world examples, the application of secure system design principles across the stages of the secure development life cycle in an identified context. | **D1** Evaluate the effectiveness of secure system design principles in reducing security risks across the secure development life cycle. |

| Pass | Merit | Distinction |
|---|---|---|
| **LO2** Develop a structured threat and risk assessment for an identified scenario | | **LO2, LO3 and LO4**<br>**D2** Justify the decisions made in the risk assessment, design and implementation of the secure system design in comprehensively meeting the security requirements of the identified scenario. |
| **P3** Develop a structured threat and risk assessment for an identified scenario, identifying assets, threats, vulnerabilities and risk evaluations.<br>**P4** Review the structured threat and risk assessment for the identified scenario with a range of internal stakeholders to identify strengths and areas for improvement. | **M2** Optimise, using the results of stakeholder feedback, the threat and risk assessment in response to the significance of identified risks and their potential impact. | |
| **LO3** Design a secure system according to an identified scenario using the principles of security by design | | |
| **P5** Design a secure system for an identified scenario using security-by-design principles, including appropriate use of cryptography, access control, and secure protocols.<br>**P6** Review the secure system design with a range of external stakeholders to determine how well the design meets the requirements of the identified scenario. | **M3** Evaluate, using the stakeholder feedback, the effectiveness of the secure system design in relation to the identified threats and security requirements of an identified scenario. | |
| **LO4** Implement a range of security controls and technologies to meet the system design requirements of the identified scenario | | |
| **P7** Implement a range of security controls and technologies to meet the system design requirements of the identified scenario.<br>**P8** Apply a test plan to thoroughly test how effectively the selected security controls and technologies meet the system design requirements of the identified scenario. | **M4** Analyse, using the results of the system testing, the effectiveness of the implemented security controls in mitigating identified threats and vulnerabilities. | |

## Recommended Resources

### Textbooks

Anderson, R. (2023) *Security Engineering: A Guide to Building Dependable Distributed Systems*. 3rd edn. Hoboken, NJ: Wiley.

Howard, M. and Lipner, S. (2024) *The Security Development Lifecycle*. 2nd edn. Hoboken, NJ: Wiley.

Richards, M. and Ford, N. (2021) *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol, CA: O'Reilly Media.

Shostack, A. (2023) *Threat Modeling: Designing for Security*. 2nd edn. Hoboken, NJ: Wiley.

Stallings, W. (2023) *Cryptography and Network Security: Principles and Practice*. 8th edn. Harlow: Pearson.

Tarandach, I. and Coles, M.J. (2021) *Threat Modeling: A Practical Guide for Development Teams*. Sebastopol, CA: O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| owasp.org/www-project-secure-by-design-framework/ | OWASP Foundation<br>Secure by Design Framework<br>(General reference) |
| www.ibm.com/think/topics/secure-software-development-life-cycle | IBM<br>What is the secure software development lifecycle?<br>(General reference) |
| www.geeksforgeeks.org/ethical-hacking/what-is-secure-software-development-life-cycle-ssdlc/ | GeeksforGeeks<br>What is Secure Software Development Life Cycle (SSDLC)<br>(Article) |
| drata.com/learn/risk/stride-threat-model | Drata Trust Academy<br>Using the STRIDE threat model.<br>(Tutorial) |
| hightable.io/iso-27001-risk-assessment-guide/ | High Table<br>The complete guide to ISO 27001 risk assessment<br>(Reference) |

| | |
|---|---|
| www.geeksforgeeks.org/computer-networks/threat-modelling/ | GeeksforGeeks<br>Threat modelling.<br>(Article) |
| www.ssl.com/article/key-management-best-practices-a-practical-guide/ | SSL.com<br>Practical overview of cryptographic key management concepts<br>(Article) |
| cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html | OWASP Cheat Sheet Series<br>Guidance on implementing multi-factor authentication<br>(Article) |
| learn.microsoft.com/en-us/entra/identity/authentication/howto-mfa-getstarted | Microsoft Learn<br>Plan a Microsoft Entra multi-factor authentication deployment<br>(Tutorial) |
| www.ncsc.gov.uk/guidance/penetration-testing | UK National Cyber Security Centre (NCSC)<br>Penetration testing<br>(General reference) |
| www.ibm.com/think/insights/pen-testing-methodology | IBM<br>Penetration testing methodologies and standards<br>(General reference) |

## Journals and articles

Das, P. *et al*. (2024) 'STRIDE-Based Cybersecurity Threat Modeling, Risk Assessment and Treatment of an In-Vehicle Infotainment System', *Vehicles,* 6(3), pp. 1140–1163. Available at: doi.org/10.3390/vehicles6030054.

Dawson, J.K. *et al*. (2023) 'Ensuring privacy and confidentiality of cloud data: A comparative analysis of cryptographic solutions', *PLOS ONE*, 18(9). Available at: doi.org/10.1371/journal.pone.0290831.

Otieno, M., Odera, D. and Ounza, J.E. (2023) 'Theory and practice in secure software development life cycle: A comprehensive survey', *World Journal of Advanced Research and Reviews*, 18(3), pp. 53–78. Available at: doi.org/10.30574/wjarr.2023.18.3.0944.

Woods, D.W. and Seymour, S. (2024) 'Evidence-based cybersecurity policy? A meta-review of security control effectiveness', *Journal of Cyber Policy*, 8(3), pp. 365–383. Available at: doi.org/10.1080/23738871.2024.2335461.

## Links

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 514: Quality of Service*

*Unit 516: Internet of Things*

*Unit 517: Applied Data Forensics.*

# Unit 516: Internet of Things

**Unit code:** **K/652/3302**

**Unit level:** **5**

**Credit value:** **15**

---

## Introduction

The Internet of Things (IoT) plays a central role in modern digital systems, underpinning the operation of connected technologies across sectors such as manufacturing, healthcare, transport, energy and smart infrastructure. IoT systems enable physical devices to sense, communicate and respond to data in real time, supporting automation, monitoring and data-driven decision-making. As organisations increasingly depend on connected systems to improve efficiency, reliability and service delivery, there is a growing need for professionals who understand how IoT solutions are designed, secured and implemented within real operational environments.

The aim of this unit is to develop students' understanding of IoT technologies from both a technical and applied perspective. It is designed to build the knowledge and skills required to analyse IoT devices and networks, consider security requirements, and design and develop IoT solutions that meet defined business needs. The unit places emphasis on practical application, ensuring students engage with concepts and approaches that reflect current industry practice and professional expectations.

The unit covers key concepts that support the effective use of IoT technologies in organisational contexts. Students examine IoT architectures, devices, communication protocols and network topologies to establish a strong technical foundation. Security considerations are explored in relation to common threats, risk mitigation strategies and secure design principles. The unit then progresses to the design and development of IoT networks, requiring students to analyse requirements, select appropriate devices and connectivity options, and implement and test an IoT solution for a given business scenario.

On successful completion of this unit, students will be able to analyse IoT technologies, evaluate security approaches, and design and develop IoT network solutions that meet specified requirements. These capabilities are directly applicable to roles such as IoT Developer, Network or Systems Engineer, IoT Solutions Architect, Cybersecurity Analyst, and Technical Consultant, and also support progression to further study in areas including cloud computing, embedded systems, cybersecurity and smart technologies.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore fundamentals of IoT devices and networks

LO2 Discuss the security considerations associated with IoT platforms

LO3 Design an IoT network to meet a given organisational scenario

LO4 Develop an IoT network to meet a given organisational scenario.

## Essential Content

LO1 **Explore fundamentals of IoT devices and networks**

*IoT architectures and models:*

Architecture layers: layered architecture e.g. architectural separation, system layering; device layer e.g. physical device focus, sensing layer; gateway layer e.g. data relay points, protocol translation; cloud computing e.g. remote services, centralised resources

Architecture structures: centralised architecture e.g. single control point, unified management; distributed architecture e.g. decentralised control, multiple nodes; client–server model e.g. service provision, request handling

Communication models: publish–subscribe model e.g. event distribution, topic messaging; data flow model e.g. data movement paths, processing stages

Data handling: data management e.g. data organisation, storage methods; cloud computing e.g. scalable storage, remote access.

*IoT devices and components:*

Device hardware: sensors e.g. environmental measurement, physical detection; actuators e.g. mechanical response, physical output; microcontrollers e.g. control processing, embedded computation; hardware platforms e.g. development boards, device bases

Device software: firmware e.g. embedded control code, device instructions; device configuration e.g. parameter settings, operational setup; interoperability e.g. device compatibility, system interaction

Operational constraints: power sources e.g. battery supply, mains power; device constraints e.g. processing limits, memory restrictions.

*IoT communication protocols and network topologies:*

Communication protocols: Message Queuing Telemetry Transport (MQTT) e.g. lightweight messaging, event communication; Hypertext Transfer Protocol (HTTP) e.g. request communication, web interaction; message formats e.g. structured data, encoding rules

Service characteristics: Quality of Service e.g. delivery reliability, assurance levels; latency e.g. response delay, transmission time; scalability e.g. network expansion, device growth

Network structures: star topology e.g. central hub model, single relay; mesh topology e.g. multi-path routing, node redundancy.

*Mechanism of operation:*

Data acquisition: sensing cycles e.g. repeated measurement, timed collection; data sampling e.g. capture intervals, sample frequency; signal encoding e.g. data representation, signal formatting

Communication processes: protocol handshaking e.g. connection setup, session initiation; message brokering e.g. message routing, intermediary handling

System response: feedback loops e.g. response cycles, system adjustment; actuation control e.g. physical control actions, automated outputs.

LO2 **Discuss the security considerations associated with IoT Platforms**

*IoT security threats and risks:*

Security threats: device compromise e.g. unauthorised device access, malware injection techniques, default credential exploitation; network attacks e.g. traffic interception, denial-of-service attacks, packet spoofing; data interception e.g. unauthorised data capture, man-in-the-middle attacks, exposed communication channels

Access risks: unauthorised access e.g. weak authentication schemes, shared credentials; firmware vulnerabilities e.g. outdated firmware versions, unsigned firmware images, insecure update mechanisms

Privacy risks: privacy concerns e.g. personal data exposure, excessive data collection, lack of user transparency; physical security risks e.g. device tampering, hardware access ports, unsecured deployment locations

Threat impact factors: device compromise e.g. loss of device control, unreliable system behaviour; data interception e.g. confidentiality breach, data misuse.

*Security mechanisms and controls:*

Data protection controls: encryption techniques e.g. AES-based data encryption, TLS-secured communication channels; key management e.g. secure key storage, key rotation practices, centralised key distribution

Identity controls: authentication methods e.g. password-based authentication, certificate-based authentication, token-based access; authorisation models e.g. role-based access control, permission hierarchies; access control models e.g. policy-based enforcement, rule-driven access restrictions

System protection controls: secure boot e.g. trusted boot chains, firmware integrity verification; secure firmware updates e.g. digitally signed updates, controlled update delivery; context-based selection of security controls e.g. risk-based control selection, environment-specific security measures

Security trade-off considerations: trade-offs between security, performance and usability e.g. processing overhead, response time impact; access control models e.g. restrictive policies, user convenience.

*Secure design principles for IoT platforms:*

Design strategies: security by design e.g. built-in protection, early risk consideration; least privilege e.g. minimal access rights, restricted permissions; defence in depth e.g. layered controls, multiple safeguards

Data and privacy principles: data protection e.g. encrypted data storage, TLS-protected data transmission, access-controlled databases; privacy by design e.g. data minimisation techniques, anonymisation of collected data, user consent mechanisms

Communication security principles: secure communication protocols e.g. protected data exchange, encrypted messaging; defence in depth e.g. multiple communication safeguards, layered protection

Governance requirements: regulatory and compliance considerations e.g. legal obligations, industry standards.

LO3 **Design an IoT network to meet a given organisational scenario**

*Business and technical requirements analysis:*

Business objectives: business objectives e.g. operational efficiency goals, service improvement targets; functional requirements e.g. system capabilities, required device behaviour; non-functional requirements e.g. availability expectations, reliability targets

Requirement constraints: data requirements e.g. data type needs, data volume expectations; performance constraints e.g. latency limits, throughput requirements; regulatory constraints e.g. data protection obligations, compliance standards

Scalability considerations: scalability needs e.g. anticipated device growth, future system expansion; performance constraints e.g. load handling capacity, response degradation risks.

*IoT network architecture design:*

Architecture representation: system architecture diagrams e.g. logical architecture diagrams, physical deployment diagrams; data flow design e.g. data movement pathways, information processing stages

Component placement: device roles e.g. sensing devices, control devices; gateway placement e.g. edge aggregation points, protocol translation nodes

Integration decisions: cloud and edge integration e.g. hybrid architectures, edge-to-cloud processing models; protocol selection e.g. MQTT for telemetry, HTTP for management interfaces; security considerations e.g. secure communication zones, protected data paths.

*Device selection and connectivity planning:*

Device characteristics: device capabilities e.g. processing power, sensor support; sensor and actuator selection e.g. temperature sensors, relay actuators

Connectivity choices: connectivity options e.g. Wi-Fi networks, cellular IoT technologies; bandwidth requirements e.g. data rate estimation, transmission capacity

Deployment conditions: power consumption e.g. battery-powered devices, energy efficiency needs; deployment environment constraints e.g. indoor environments, outdoor exposure considerations.

LO4 **Develop an IoT network to meet a given organisational scenario**

*IoT network implementation and configuration:*

Deployment activities: device provisioning e.g. device identity registration, initial device onboarding; firmware deployment e.g. firmware image installation, version management; network configuration e.g. IP addressing, network parameter setup

System integration: protocol configuration e.g. MQTT topic configuration, HTTP endpoint definition; cloud platform integration e.g. Amazon Web Services (AWS) IoT Core connection, Azure IoT Hub integration; security configuration e.g. access credentials setup, encryption enablement.

*Testing and validation processes:*

Functional assurance: functional testing e.g. device behaviour verification, message transmission checks; connectivity testing e.g. network reachability tests, connection stability checks

Performance and security validation: performance testing e.g. throughput measurement, response time testing; security testing e.g. vulnerability assessments, access control verification; data validation e.g. data accuracy checks, completeness verification

Quality confirmation: validation against requirements e.g. requirements traceability checks, acceptance testing; reliability metrics e.g. system uptime measures, failure rate indicators; performance thresholds e.g. acceptable response limits, system capacity indicators.

*Operational readiness and deployment considerations:*

Operational preparation: monitoring and logging e.g. system logs, performance monitoring dashboards; maintenance planning e.g. scheduled maintenance activities, update management

Sustainability considerations: scalability considerations e.g. adding additional devices, system expansion readiness; reliability and resilience e.g. fault tolerance mechanisms, system recovery capability

Handover activities: documentation e.g. system configuration records, operational manuals; handover to operations e.g. transition to operational teams, support responsibility transfer.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explore fundamentals of IoT devices and networks | | **LO1 and LO2** |
| **P1** Explore, using a selection of real-world examples, the fundamentals of IoT networks, models and components used in different application contexts.<br><br>**P2** Describe, using a selection of real-world examples, how the different IoT protocols and communication mechanisms allow for effective communication within an IoT network. | **M1** Analyse, using a selection of real-world examples, the benefits and limitations of IoT devices, communication protocols and network architectures in different application contexts. | **D1** Critically evaluate, using a selection of real-world examples, the effectiveness of different IoT devices, network architectures and security considerations for supporting scalable, reliable and secure IoT solutions for a range of different application contexts. |
| **LO2** Discuss the security considerations associated with IoT Platforms | | |
| **P3** Discuss, using a selection of real-world examples, the security threats to IoT networks in different application contexts.<br><br>**P4** Explore, using a selection of real-world examples, the different security mechanisms, including security by design strategy, that are used to minimise the threat to IoT networks. | **M2** Assess, using a selection of real-world examples, a range of threats to IoT platforms and systems and the effectiveness of the strategies used to mitigate against these threats. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design an IoT network to meet a given organisational scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design, development and deployment of the IoT network in how effectively it meets the requirements of the given organisational scenario. |
| **P5** Design an IoT network that meets the identified requirements of a given organisational scenario, considering the network architecture and device selection.<br>**P6** Review the design documentation with a range of stakeholders to identify strengths and areas for improvement. | **M3** Optimise, using the results of the stakeholder feedback, the design of the IoT network so that it comprehensively addresses the requirements of the given organisational scenario. | |
| **LO4** Develop an IoT network to meet a given organisational scenario | | |
| **P7** Develop an IoT network to meet a given organisational scenario in line with the proposed network design.<br>**P8** Implement a test plan to thoroughly test the configuration and performance of the deployed IoT network in meeting the requirements of the given organisational scenario. | **M4** Evaluate, using the results of the test plan, the effectiveness of the deployed IoT network in how well it matches the design and meets the requirements of the given organisational scenario. | |

## Recommended Resources

### Textbooks


Al-Turjman, F. (2021) *Security in IoT-Enabled Spaces*. London: Routledge.

Bahga, A. and Madisetti, V. (2023) *Internet of Things: A Hands-On Approach*. 3rd edn. New Delhi: Universities Press.

Buyya, R. and Dastjerdi, A.V. (2024) *Internet of Things: Principles and Paradigms*. 2nd edn. Amsterdam: Morgan Kaufmann.

Dian, F.J. (2022) *Fundamentals of Internet of Things: For Students and Professionals*. Hoboken, NJ: Wiley-IEEE Press.

Hanes, D. *et al*. (2021 update) *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Indianapolis, IN: Cisco Press (Pearson).

Hassan, Q.F. (2023) *Internet of Things A to Z: Technologies and Applications*. 2nd edn. Hoboken, NJ: Wiley.

McEwen, A. and Cassimally, H. (2024) *Designing the Internet of Things*. 2nd edn. Chichester: Wiley.


### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.ibm.com/internet-of-things | IBM<br>IoT solutions<br>(General reference) |
| www.geeksforgeeks.org/computer-networks/architecture-of-internet-of-things-iot/ | GeeksforGeeks<br>Architecture of Internet of Things (IoT)<br>(General reference) |
| www.rfwireless-world.com/tutorials/iot-tutorial-basics-architecture-protocol-stack | RF Wireless World<br>IoT Tutorial: Basics, Architecture, and Protocol Stack<br>(Tutorial) |
| www.frontiersin.org/articles/10.3389/frai.2024.1397480/full | Frontiers in Artificial Intelligence<br>Exploring security threats and solutions Techniques for Internet of Things (IoT)<br>(General reference) |

| | |
|---|---|
| www.iotevolutionworld.com/iot/articles/463430-from-perimeter-identity-part-1-zero-trust-architectures.htm | IoT Evolution World<br>From Perimeter to Identity (Part 1): Zero-Trust Architectures for IoT and OT<br>(Review) |
| www.ruckusnetworks.com/blog/2023/iot-network-design-best-practices-for-connectivity/ | RUCKUS Networks<br>IoT Network Design: Designing for Connectivity<br>(Article) |
| hal.science/hal-04863852v1/file/IOTProjectWorkBook.pdf | HAL Open Science<br>Internet of Things: Systems Design Project Workbook<br>(PDF) |
| testgrid.io/blog/iot-testing-the-complete-guide/ | TestGrid<br>IoT Testing: A Complete Guide to Validating Smart Devices in the Real-world<br>(Blog) |
| yalantis.com/blog/iot-testing-guide/ | Yalantis<br>IoT Testing: A Complete Guide to Validating Smart Devices in the Real-world<br>(Blog) |
| yalantis.com/blog/iot-testing-guide/ | Yalantis<br>The Ultimate Guide to IoT Testing<br>(Blog) |
| www.geeksforgeeks.org/system-design/design-patterns-for-internet-of-thingsiot/ | GeeksforGeeks<br>Design Patterns for Internet of Things(IoT)<br>(Article) |
| www.cedalo.com/blog/complete-mqtt-protocol-guide | Cedalo<br>MQTT Protocol Guide: Everything You Need to Know<br>(Reference) |

**Journals and articles**

Chataut, R., Phoummalayvane, A. and Akl, R. (2023) 'Unleashing the Power of IoT: A Comprehensive Review of IoT Applications and Future Prospects in Healthcare, Agriculture, Smart Homes, Smart Cities, and Industry 4.0', *Sensors*, 23(16), 7194. Available at: doi.org/10.3390/s23167194.

Elgazzar, K. *et al*. (2022) 'Revisiting the internet of things: New trends, opportunities and grand challenges', *Frontiers in the Internet of Things*, 1. Available at: doi.org/10.3389/friot.2022.1073780.

Gubbi, J. *et al*. (2013) 'Internet of Things (IoT): A vision, architectural elements, and future directions', *Future Generation Computer Systems*, 29(7), pp. 1645–1660. Available at: doi.org/10.1016/j.future.2013.01.010.

Sicari, S. *et al*. (2015) 'Security, privacy and trust in Internet of Things: The road ahead', *Computer Networks*, 76, pp. 146–164. Available at: doi.org/10.1016/j.comnet.2014.11.008.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 514: Quality of Service (QoS)*

*Unit 515: Secure System Design.*

# Unit 517: Applied Data Forensics

**Unit code:** L/652/3303

**Unit level:** 5

**Credit value:** 15

---

## Introduction

As organisations continue to depend on digital systems to store, process and exchange information, the need to investigate digital activity accurately and responsibly has become increasingly important. Applied data forensics plays a key role in supporting organisations when responding to incidents such as data breaches, system misuse, fraud and regulatory investigations. The discipline combines technical investigation skills with legal awareness and professional judgement, ensuring that digital evidence can be identified, preserved and interpreted in a reliable and defensible manner.

The aim of this unit is to develop students' understanding of how data forensic principles are applied in real-world investigative contexts. The unit focuses on professional practice rather than tool-specific training, enabling students to make informed decisions about investigative approaches, evidence handling and documentation. Through this applied focus, students will develop the skills and judgement needed to conduct investigations that meet both organisational and legal expectations.

The unit covers the core principles that underpin effective data forensic practice, including recognised forensic process models, evidence integrity and chain of custody. Students examine a range of forensic tools and technologies, considering their accuracy, limitations and suitability for different investigative scenarios. The unit then moves into applied investigation, requiring students to plan and implement a structured data forensic investigation and to produce professional forensic documentation that reflects basic legislative and professional requirements.

On successful completion of the unit, students will be able to carry out structured data forensic investigations, critically evaluate forensic tools and approaches, and produce clear, defensible forensic reports. These skills support progression into roles such as Digital Forensic Analyst, Cybersecurity Analyst, Incident Response Practitioner, Technical Investigator, Compliance Officer and Information Security Analyst, as well as further study in specialist areas of digital forensics and cyber investigation.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the core principles of applied data forensics

LO2 Review a range of forensic tools and technologies for accuracy and appropriateness

LO3 Implement a data forensic investigation on a given scenario

LO4 Develop legally compliant forensic documentation for a given scenario.

## Essential Content

LO1 **Discuss the core principles of applied data forensics**

*Forensic process models:*

Process stages: identification e.g. incident recognition, case start; preservation e.g. evidence safeguarding, data protection; acquisition e.g. data capture, evidence copying; examination e.g. data inspection, file review; analysis e.g. pattern finding, evidence interpretation

Workflow management: presentation e.g. findings summary, report delivery; review e.g. quality checking, process checking; workflow e.g. task sequence, case flow

Process characteristics: life cycle e.g. case phases, investigation cycle; iteration e.g. repeat checks, re-examination; consistency e.g. standard steps, repeatable results.

*Evidence integrity and chain of custody:*

Evidence handling: handling e.g. safe handling, controlled handling; labelling e.g. evidence tags, item identifiers; access e.g. authorised access, restricted access; storage e.g. secure storage, evidence storage

Integrity assurance: integrity e.g. unchanged data, evidence accuracy; authenticity e.g. source confirmation, genuine evidence; continuity e.g. custody continuity, handover record

Traceability controls: traceability e.g. evidence tracking, item history; logging e.g. audit logs, access logs; admissibility e.g. court acceptance, legal use; contamination e.g. evidence damage, data alteration.

*Data acquisition and preservation:*

Acquisition approaches: live acquisition concepts e.g. running systems, powered-on systems; dead acquisition concepts e.g. powered-off systems, isolated devices; logical acquisition e.g. file copying, directory capture; physical acquisition e.g. full imaging, storage duplication

Protection mechanisms: write-blocking e.g. read-only access, device protection; data integrity verification e.g. check values, match checks; hashing principles e.g. hash values, hash matching

Validation constraints: validation of captured data e.g. capture checking, result matching; acquisition constraints e.g. time limits, device limits; volatile data considerations e.g. memory data, running data; non-volatile data considerations e.g. stored data, disk data.

*Forensic data examples:*

Stored content: files e.g. user files, system files; images e.g. photos, screenshots; documents e.g. reports, letters; databases e.g. data tables, entries

Context data: metadata e.g. file details, device details; logs e.g. activity logs, event logs; timestamps e.g. time markers, access times

Investigation traces: emails e.g. message content, attachments; artefacts e.g. browser artefacts, system artefacts; records e.g. account records, transaction records; fragments e.g. deleted fragments, file fragments.

LO2 **Review a range of forensic tools and technologies for accuracy and appropriateness**

*Forensic tool categories and functions:*

Tool purposes: acquisition tools e.g. data capture software, imaging tools; analysis tools e.g. examination software, investigation tools; reporting tools e.g. report generators, output tools

Tool characteristics: validation utilities e.g. checking tools, verification utilities; integrated toolchains e.g. combined toolsets, investigation suites

Tool scope: open-source tools e.g. community tools, free tools; commercial tools e.g. licensed tools, vendor tools; specialist tools e.g. purpose tools, task-specific tools; general-purpose tools e.g. multi-use tools, common tools; platform compatibility e.g. operating systems, device types

Tool examples: acquisition tools e.g. FTK Imager, Guymager; analysis tools e.g. Autopsy, EnCase; reporting tools e.g. EnCase reports, AXIOM reports.

*Accuracy, validation and reliability:*

Testing and validation: tool testing methodologies e.g. functional testing, practical testing; validation frameworks e.g. testing standards, validation models

Result reliability: repeatability of results e.g. consistent outputs, repeat tests; error rates e.g. incorrect results, failure rates; false positives and negatives e.g. incorrect matches, missed results

Control and evidence: version control e.g. software versions, update tracking; configuration management e.g. setup settings, configuration records; vendor documentation e.g. user guides, manuals; independent testing evidence e.g. third-party tests, external reviews.

*Appropriateness and limitations:*

Contextual suitability: contextual suitability e.g. investigation type, case context; data type compatibility e.g. file types, data formats

Performance and scale: scalability e.g. large datasets, small datasets; performance constraints e.g. speed limits, processing limits; automation impacts e.g. automated processes, reduced manual effort

Professional acceptance: evidential transparency e.g. clear results, visible processes; legal defensibility e.g. court use, legal acceptance; professional acceptance e.g. industry use, practitioner trust; reporting accuracy e.g. correct outputs, clear findings.

LO3 **Implement a data forensic investigation on a given scenario**

*Structured investigative methodologies:*

Investigation planning: forensic workflow execution e.g. defined investigation steps, structured processes; scenario scoping e.g. case boundaries, investigation scope; hypothesis development e.g. working assumptions, investigation questions

Evidence management: systematic evidence handling e.g. controlled handling, documented handling; procedural sequencing e.g. ordered steps, logical sequence; documentation checkpoints e.g. recorded stages, progress records

Quality control: quality assurance controls e.g. review checks, verification steps; documentation checkpoints e.g. evidence records, case notes.

*Data analysis techniques:*

File and system analysis: file system analysis e.g. directory review, file structure; metadata examination e.g. file properties, creation details; artefact identification e.g. system artefacts, application artefacts

Event and activity analysis: log analysis e.g. system logs, activity logs; correlation of artefacts e.g. linked evidence, related artefacts; timeline reconstruction e.g. event ordering, time sequencing

Pattern recognition: pattern identification e.g. repeated actions, common behaviour; anomaly detection e.g. unusual activity, unexpected events; data interpretation e.g. file timestamps, access logs.

*Toolchain integration and repeatability:*

Tool co-ordination: tool interoperability e.g. compatible tools, shared formats; workflow automation e.g. automated steps, scripted tasks; consistency of outputs e.g. repeatable results, stable findings

Process reliability: process repeatability e.g. same steps, repeat investigations; documentation of investigative decisions e.g. decision records, rationale notes

Result validation: validation of findings e.g. result checking, evidence confirmation.

LO4 **Develop legally compliant forensic documentation for a given scenario**

*Forensic report structure and content:*

Report purpose and scope: report purpose e.g. investigation summary, case outcome; scope definition e.g. investigation limits, case boundaries

Report structure: cover page e.g. case identifier, report title; table of contents e.g. section listing, page numbers; section headings e.g. clear sections, logical order; appendices e.g. supporting material, supplementary evidence

Method and findings: methodology description e.g. investigation approach, process overview; evidence summary e.g. collected items, evidence overview; findings presentation e.g. result explanation, evidence outcomes

Evaluation and conclusions: interpretation of results e.g. result meaning, evidence explanation; limitations e.g. data limits, investigation constraints; conclusions e.g. final findings, case conclusions; professional tone e.g. formal language, objective wording.

*Evidential weighting and interpretation:*

Evidence assessment: relevance assessment e.g. case relevance, investigation relevance; reliability considerations e.g. trustworthiness, data accuracy

Evidence comparison: corroboration of evidence e.g. supporting evidence, matching artefacts; transparency of reasoning e.g. clear logic, explained decisions

Interpretative techniques: contextual interpretation e.g. usage context, system context; temporal interpretation e.g. time sequencing, event timing; activity reconstruction e.g. user actions, system behaviour

Professional judgement: facts e.g. verified information, objective findings; opinions e.g. analyst interpretation, informed views; assumptions e.g. stated assumptions, inferred premises; confidence levels e.g. evidence strength, certainty degree.

*Legal and professional compliance*

Legislative requirements: basic legislative requirements e.g. Data Protection Act 2018, UK General Data Protection Regulation (GDPR); disclosure obligations e.g. Criminal Procedure and Investigations Act 1996, disclosure rules; current awareness, developments, evolving practice

Professional standards: professional standards e.g. National Police Chiefs' Council (NPCC) guidelines, organisational policies; ethical considerations e.g. responsible conduct, professional behaviour

Data protection and accountability: data protection principles e.g. lawful processing, secure handling; audit readiness e.g. review trails, compliance evidence.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Discuss the core principles of applied data forensics</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of applied data forensic principles, tools and techniques in supporting evidence generation for professional forensic investigations.</td></tr>
<tr><td>P1 Discuss, using a selection of real-world examples, the core principles of applied data forensics models.<br><br>P2 Explain, using a selection of real-world examples, the types and acquisition of forensic data and how the evidence is handled.</td><td>M1 Analyse, using a selection of real-world examples, the strengths and limitations of the principles of data forensics in a digital forensic investigation.</td></tr>
<tr><td colspan="2">LO2 Review a range of forensic tools and technologies for accuracy and appropriateness</td></tr>
<tr><td>P3 Describe, using a selection of real-world examples, a range of forensic tools and technologies in terms of purpose and functionality.<br><br>P4 Review, using a selection of real-world examples, a range of forensic tools and technologies in terms of their accuracy and appropriateness for forensic investigation.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of a range of forensic tools and technologies in identified forensic scenarios.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Implement a data forensic investigation on a given scenario | | **LO3 and LO4**<br>**D2** Critically evaluate the design and implementation of the forensic investigation in producing effective forensic documentation that supports informed conclusions, legal scrutiny and professional standards. |
| **P5** Design a data forensic investigation on a given scenario considering quality control and evidence management.<br>**P6** Implement a data forensic investigation on a given scenario using structured forensic methodologies and identified tools and techniques. | **M3** Analyse the effectiveness of the applied forensic methodology in supporting repeatability, quality and evidential integrity. | |
| **LO4** Develop legally compliant forensic documentation for a given scenario | | |
| **P7** Review the data generated from the forensic investigation to generate informed conclusions about the given scenario.<br>**P8** Develop forensic documentation for an identified forensic scenario that reflects basic legislative and professional requirements.<br>**P9** Review the forensic documentation with a range of identified stakeholders to identify strengths and areas for improvement. | **M4** Analyse, using the results of stakeholder feedback, the clarity and evidential value of forensic documentation in supporting investigative outcomes. | |

## Recommended Resources

### Textbooks


Casey, E. (2023) *Digital Evidence and Computer Crime*. 4th edn. London: Academic Press.

Easttom, C. (2021) *Digital Forensics, Investigation and Response*. 4th edn. Burlington, MA: Jones & Bartlett Learning.

Holt, T.J., Bossler, A.M. and Seigfried-Spellar, K.C. (2022) *Cybercrime and Digital Forensics: An Introduction*. 2nd edn. Abingdon: Routledge.

Kävrestad, J., Birath, M. and Clarke, N. (2024) *Fundamentals of Digital Forensics: A Guide to Theory, Research and Applications*. 3rd edn. Cham: Springer.

Nelson, B., Phillips, A. and Steuart, C. (2024) *Guide to Computer Forensics and Investigations*. 7th edn. Boston: Cengage Learning.

Nuñez, J.T. (2021) *Digital Forensics for Legal Professionals: Understanding Digital Evidence from the Warrant to the Courtroom*. Abingdon: Routledge.

Sammons, J. (2023) *The Basics of Digital Forensics*. 3rd edn. Amsterdam: Elsevier.


### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| hawkeyeforensic.com/timeline-analysis-in-digital-forensics-reconstructing-events/ | Hawk Eye Forensic<br>Timeline Analysis in Digital Forensics: Reconstructing Events<br>(Article) |
| westoahu.hawaii.edu/cyber/forensics-weekly-executive-summmaries/importance-of-digital-forensics-process-models-some-examples/ | University of Hawai'i – West O'ahu<br>Importance of Digital Forensics Process Models: Some Examples<br>(Article) |
| www.cyberforensicacademy.com/blog/how-to-choose-the-right-forensic-tool-for-any-investigation | Cyber Forensics Academy<br>How to Choose the Right Forensic Tool for Any Investigation<br>(Article) |
| nvlpubs.nist.gov/nistpubs/ir/2025/NIST.IR.8589.pdf | National Institute of Standards and Technology (NIST)<br>Validation in Forensic Science<br>(PDF) |

| | |
|---|---|
| hawkeyeforensic.com/digital-forensics-investigation-process/ | Hawk Eye Forensic<br>Understanding the Digital Forensics Investigation Process<br>(Tutorial) |
| www.geeksforgeeks.org/computer-networks/five-phases-of-computer-forensics-investigation-procedure/ | GeeksforGeeks<br>Five Phases of Computer Forensics Investigation Procedure<br>(Article) |
| computerforensicslab.co.uk/legal-considerations-digital-forensics-uk/ | Computer Forensics Lab (UK)<br>Legal Considerations for Digital Forensics: UK Compliance<br>(General reference) |
| www.gdpr-advisor.com/how-gdpr-impacts-digital-forensics-and-incident-response-investigations/ | GDPR Advisor<br>How GDPR Impacts Digital Forensics and Incident Response Investigations<br>(Article) |
| www.numberanalytics.com/blog/validation-in-digital-forensics | Number Analytics<br>Validation in Digital Forensics<br>(Article) |

**Journals and articles**

Beebe, N.L. and Clark, J.G. (2005) 'A hierarchical, objectives-based framework for the digital investigations process', *Digital Investigation*, 2(2), pp. 147–167. Available at: doi.org/10.1016/j.diin.2005.04.002.

Ismail, I. and Zainol Ariffin, K.A. (2025) 'The admissibility of digital evidence from open-source forensic tools: Development of a framework for legal acceptance', *PLOS ONE*, 20(9), e0331683. Available at: doi.org/10.1371/journal.pone.0331683.

Safie, S.I. and Md Bashah, S.R. (2023) 'A Comprehensive Review of the Evolution and Future Directions of Digital Forensic Investigation Model', *International Journal of Emerging Technology and Advanced Engineering*, 13(7). Available at: doi.org/10.46338/IJETAE0723_01.

Stoykova, R. and Franke, K. (2023) 'Reliability validation enabling framework (RVEF) for digital forensics in criminal investigations', *Forensic Science International: Digital Investigation*, 45, 301554. Available at: doi.org/10.1016/j.fsidi.2023.301554.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 515: Secure System Design.*

# Unit 518: Cloud Data Centre Operations

**Unit code:** M/652/3304

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Cloud data centres play a central role in the delivery of modern digital services, supporting everything from enterprise applications to cloud-native platforms and emerging technologies. These environments combine large-scale physical infrastructure with virtualised and software-defined resources to provide flexible, on-demand computing capability. As organisations become increasingly dependent on digital systems, the performance, reliability and scalability of cloud data centres have become critical to maintaining service continuity and meeting user expectations.

The aim of this unit is to develop a clear and applied understanding of how cloud data centres are structured, operated and maintained. Students will explore how physical infrastructure, virtualisation technologies and orchestration platforms work together to deliver reliable and scalable services. The unit also focuses on the operational practices required to manage performance, availability, security and compliance within cloud environments, preparing students to understand how these systems are managed in real-world contexts.

The unit covers a broad range of topics relevant to cloud data centre operations. Key areas include infrastructure design, virtualisation and resource abstraction, and the integration of compute, storage and networking systems. Students will examine monitoring and observability, capacity planning and scaling strategies, as well as approaches to ensuring availability, resilience and operational efficiency. The unit also introduces security controls, risk management processes and regulatory considerations, alongside incident response and recovery planning within cloud-based environments.

On successful completion of this unit, students will be able to analyse cloud data centre architectures, evaluate operational processes and design appropriate responses to service incidents in line with organisational and regulatory requirements. These skills are directly applicable to a range of technical roles within the cloud and infrastructure domain, including Cloud Operations Engineer, Site Reliability Engineer, Infrastructure Engineer, Data Centre Operations Specialist and Platform Engineer, supporting progression into operational and engineering roles within modern IT environments.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core principles of cloud data centre operations

LO2 Review the range of processes involved in cloud data centre operations

LO3 Discuss the security and compliance principles relevant to cloud data centre operations

LO4 Design a regulation-compliant incident response policy for a cloud data centre in response to an identified scenario.

## Essential Content

LO1 **Explain the core principles of cloud data centre operations**

*Physical data centre infrastructure and facilities:*

Infrastructure components: data centre layout e.g. zones, floorplans; rack configuration e.g. rack units, mounting; server hardware e.g. CPUs, memory; storage arrays e.g. disk arrays, redundant array of independent disks (RAID)

Power and cooling systems: power distribution systems e.g. power supply, circuits; power distribution units (PDUs) e.g. rack PDUs, monitored PDUs; cooling strategies e.g. air cooling, cooling layouts; uninterruptible power supply (UPS) e.g. battery backup, standby power

Environmental management: environmental monitoring e.g. sensors, alerts; temperature e.g. ranges, thresholds; airflow e.g. vents, circulation

Resilience design: redundancy models e.g. backup systems, duplication; N+1 e.g. spare capacity, redundancy level; fault-tolerant design e.g. fail-safe systems, resilience.

*Virtualisation and abstraction layers:*

Virtualisation technologies: hypervisor-based virtualisation e.g. virtual machines, hosts; containerisation platforms e.g. containers, Docker; software-defined infrastructure e.g. virtual networks, software control

Resource abstraction: compute resources e.g. CPUs, virtual machines; storage resources e.g. volumes, disks; networking resources e.g. bandwidth, virtual networks

Workload isolation: multi-tenancy e.g. shared environments, tenants; workload isolation e.g. separation, isolation layers; resource contention e.g. competition, conflicts

Orchestration environments: orchestration e.g. automation tools, schedulers; distributed environments e.g. clusters, nodes.

*Network and storage architecture within data centres:*

Network architectures: data centre networking models e.g. architectures, layouts; spine-leaf architectures e.g. spine switches, leaf switches; high-throughput switching e.g. fast switches, bandwidth

Storage systems: storage architectures e.g. storage designs, layouts; storage area network (SAN) e.g. storage networks, fibre channel; network-attached storage (NAS) e.g. file storage, network storage

System integration: compute integration e.g. servers, processing units; storage integration e.g. storage links, systems; network integration e.g. connections, interfaces

Scalability: scalable workloads e.g. growth, expansion.

*Operational roles and responsibilities:*

Role definitions: operations engineers e.g. technical staff, engineers; site reliability engineers e.g. reliability roles, site reliability engineering (SRE); platform engineers
e.g. platform roles, developers

Operational metrics: availability e.g. uptime, access; performance e.g. speed, efficiency; capacity e.g. limits, usage

Team structures: infrastructure teams e.g. system teams, ops teams; platform teams e.g. development teams, platform staff; application teams e.g. software teams, developers

Co-ordination: Co-ordination e.g. communication, collaboration.

*Workload orchestration and service delivery:*

Workload management: workload scheduling e.g. timing, allocation; workload placement e.g. location, distribution; load balancing e.g. traffic distribution, balancing

Elasticity and scaling: elasticity e.g. flexibility, adjustment; scaling mechanisms e.g. scaling tools, autoscaling

Service metrics: service level objectives e.g. targets, goals; service level indicators e.g. metrics, measurements

Resilience mechanisms: fault tolerance e.g. error handling, recovery; self-healing systems e.g. automation, recovery systems; cloud environment e.g. platforms, services.

LO2 **Review the range of processes involved in cloud data centre operations**

*Monitoring and observability:*

Monitoring data: metrics e.g. performance data, usage data; logs e.g. system logs, application logs; traces e.g. request traces, transaction paths

Monitoring systems: infrastructure monitoring e.g. server monitoring, network monitoring; application monitoring e.g. software monitoring, service monitoring; dashboards e.g. visual displays, status panels

Alerting and detection: alerting systems e.g. alerts, notifications; anomaly detection e.g. unusual activity, pattern changes; visibility e.g. system visibility, data visibility.

*Capacity planning and resource optimisation:*

Planning processes: demand forecasting e.g. usage trends, demand patterns; utilisation analysis e.g. resource usage, capacity use; workload profiling e.g. workload types, usage patterns

Scaling approaches: vertical scaling e.g. resource increase, hardware upgrade; horizontal scaling e.g. additional servers, distributed systems; scaling strategies e.g. scaling plans, scaling rules

Performance optimisation: cost–performance trade-offs e.g. cost balance, performance balance; optimisation e.g. efficiency improvements, resource tuning; resource management e.g. compute resources, storage resources.

*Availability, redundancy and recovery processes:*

Availability design: high availability e.g. uptime targets, availability levels; redundancy e.g. backup systems, system duplication; zones e.g. availability zones, system areas

Recovery mechanisms: failover mechanisms e.g. automatic failover, manual failover; backup strategies e.g. data backups, system backups; data replication e.g. data copies, mirrored data

Recovery objectives: recovery time objectives (RTO) e.g. RTO targets, recovery time; recovery point objectives (RPO) e.g. RPO targets, data recovery points; recovery testing e.g. test procedures, test plans.

*Operational efficiency and sustainability:*

Efficiency measures: energy consumption e.g. power usage, electricity use; cooling efficiency e.g. cooling performance, heat management; power usage effectiveness e.g. PUE metrics, efficiency ratios

Environmental impact: environmental impact e.g. emissions, carbon footprint; cost optimisation e.g. cost reduction, budget control; operational efficiency e.g. process efficiency, system efficiency.

*Automation and operational tooling:*

Automation tools: automation frameworks e.g. automation tools, automation platforms; infrastructure as code e.g. configuration scripts, deployment scripts; configuration management e.g. system configuration, version control

Process improvement: deployment pipelines e.g. continuous integration/ continuous deployment (CI/CD) pipelines, release pipelines; consistency e.g. standardisation, uniformity; repeatability e.g. repeat processes, consistent outcomes.

LO3 **Discuss the security and compliance principles relevant to cloud data centre operations**

*Security architecture within data centres:*

Security controls: physical security controls e.g. locks, access cards; access restrictions e.g. permissions, user roles; surveillance e.g. cameras, monitoring systems

Network protection: network security e.g. secure networks, protections; segmentation e.g. network segments, isolation zones; firewalls e.g. filtering systems, firewall rules

Access management: identity and access management e.g. user accounts, roles; encryption e.g. data encryption, secure data; key management practices e.g. encryption keys, key storage.

*Threats, vulnerabilities and risk management:*

Threat types: threats e.g. cyber threats, physical threats; cyberattacks e.g. malware, phishing; insider threats e.g. unauthorised access, misuse

Vulnerability management: misconfiguration e.g. incorrect settings, weak controls; vulnerability identification e.g. scanning tools, assessment tools; physical incidents e.g. equipment damage, system failure

Risk processes: risk assessment methodologies e.g. risk scoring, risk frameworks; mitigation strategies e.g. security controls, safeguards; infrastructure e.g. systems, components.

*Compliance and regulatory frameworks:*

Compliance requirements: industry standards e.g. ISO 27001, ISO 22301; regulatory requirements e.g. General Data Protection Regulation (GDPR), Data Protection Act 2018; data protection obligations e.g. data privacy, data security

Governance processes: governance policies e.g. organisational policies, IT policies; audit processes e.g. audits, reviews; reporting processes e.g. reports, records

Accountability: accountability e.g. responsibility, ownership; documentation requirements e.g. records, documents.

*Operational governance and control:*

Change processes: change management processes e.g. change requests, approval workflows; configuration control e.g. version control, system baselines; audit trails e.g. activity logs, change records

Control structures: policy enforcement e.g. policy rules, compliance controls; organisational governance structures e.g. management structures, reporting lines; alignment e.g. standards alignment, policy alignment.

*Security monitoring and incident detection:*

Monitoring tools: security information and event management (SIEM) e.g. SIEM tools, monitoring platforms; intrusion detection e.g. intrusion detection systems (IDS), alerts; intrusion prevention e.g. intrusion prevention system (IPS), controls

Detection and analysis: log analysis e.g. log review, data analysis; real-time alerting e.g. alerts, notifications; investigation e.g. incident investigation, analysis

Security events: security events e.g. alerts, incidents.

LO4 **Design a regulation-compliant incident response policy for a cloud data centre in response to an identified scenario**

*Incident identification and classification:*

Detection processes: detection mechanisms e.g. monitoring tools, sensors; categorisation e.g. incident types, classifications; incidents e.g. digital incidents, physical incidents

Assessment factors: severity e.g. impact levels, priority levels; impact e.g. service impact, data impact; prioritisation e.g. response order, urgency levels

Response planning: response actions e.g. containment actions, mitigation steps.

*Incident response life cycle:*

Response stages: preparation e.g. readiness plans, resources; detection e.g. identification processes, alerts; containment e.g. isolation actions, control measures

Recovery stages: eradication e.g. removal actions, cleaning processes; recovery e.g. restoration actions, service recovery; post-incident review e.g. reviews, evaluations

Response co-ordination: cyber security incident response team (CSIRT) e.g. response teams, security teams; communication protocols e.g. communication plans, reporting lines; escalation procedures e.g. escalation paths, authority levels.

*Response to digital and physical threats:*

Digital threats: cyber incidents e.g. malware attacks, data breaches; denial-of-service attacks e.g. traffic flooding, service disruption; response actions e.g. mitigation steps, containment actions

Physical threats: equipment failure e.g. hardware faults, system breakdowns; power disruption e.g. outages, power loss; environmental hazards e.g. fire risks, flooding risks

Response environments: infrastructure e.g. systems, components; facilities e.g. buildings, data centres.

*Disaster recovery and business continuity:*

Continuity planning: continuity planning e.g. business plans, service plans; strategies e.g. response strategies, recovery strategies; implementation e.g. process setup, deployment

Recovery processes: failover e.g. automatic failover, manual failover; redundancy e.g. backup systems, duplication; restoration processes e.g. data recovery, system restoration

Resilience planning: resilience planning e.g. preparedness plans, risk planning; critical services e.g. key systems, essential services.

*Compliance and reporting requirements:*

Compliance needs: regulatory obligations e.g. legal requirements, standards; incident handling e.g. handling procedures, response actions; evidence collection e.g. data collection, log collection

Reporting processes: evidence preservation e.g. data storage, record keeping; reporting e.g. incident reports, formal reports; stakeholders e.g. users, organisations

Authority engagement: authorities e.g. regulators, agencies; documentation e.g. records, reports.

*Continuous improvement and resilience:*

Post-incident analysis: post-incident analysis e.g. review processes, evaluations; weaknesses e.g. gaps, issues; lessons learned e.g. outcomes, findings

Improvement actions: control improvements e.g. system controls, safeguards; process improvements e.g. workflow changes, updates; response strategies e.g. improved strategies, revised plans

Operational resilience: operational practices e.g. standard practices, procedures.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the core principles of cloud data centre operations</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Critically evaluate how operational principles, processes and security compliance contribute to performance, resilience and efficiency in cloud data centre operations.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the physical infrastructure and virtualisation and storage architectures used in cloud data centre operations.<br><br>P2 Examine, using a selection of real-world examples, the roles, responsibilities and workload management practices within cloud data centre environments.</td><td>M1 Assess, using a selection of real-world examples, how the principles of cloud data centre operations enable scalable and resilient service delivery.</td></tr>
<tr><td colspan="2">LO2 Review the range of processes involved in cloud data centre operations</td></tr>
<tr><td>P3 Explore, using a selection of real-world examples, the monitoring, capacity planning and redundancy processes used in cloud data centre environments.<br><br>P4 Review, using a selection of real-world examples, how automation can contribute to the efficiency of cloud data centre operations.</td><td>M2 Analyse, using a selection of real-world examples, the effectiveness of operational processes in contributing to efficient cloud data centre service delivery.</td></tr>
<tr><td colspan="2">LO3 Discuss the security and compliance principles relevant to cloud data centre operations</td></tr>
<tr><td>P5 Discuss, using a selection of real-world examples, security controls used to protect cloud data centre environments from identified threats and vulnerabilities.<br><br>P6 Review, using a selection of real-world examples, compliance and governance requirements relevant to cloud data centre operations.</td><td>M3 Analyse, using a selection of real-world examples, how security and governance practices mitigate operational and regulatory risk.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Design a regulation-compliant incident response policy for a cloud data centre in response to an identified scenario | | |
| **P7** Design a regulatory compliant incident response policy for a cloud data centre in response to an identified scenario.<br><br>**P8** Review the design of the regulatory compliant incident response policy for a cloud data centre with a range of identified stakeholders to identify strengths and areas for improvement. | **M4** Evaluate the effectiveness of the regulatory compliant incident response policy in meeting the requirements of the identified scenario. | **D2** Justify the decisions made in the design of the regulatory compliant incident response policy that enables it to comprehensively meet the requirements of an identified scenario. |

## Recommended Resources

### Textbooks

Antonopoulos, N. and Gillam, L. (2023) *Cloud Computing: Principles, Systems and Applications*. Cham: Springer.

Davis, D. (2023) *Data Center Virtualization Fundamentals*. Indianapolis, IN: Cisco Press.

Erl, T., Mahmood, Z. and Puttini, R. (2021) *Cloud Computing: Concepts, Technology and Architecture*. 2nd edn. Boston, MA: Pearson.

Geng, H. (2024) *Data Center Handbook*. 3rd edn. Hoboken, NJ: Wiley.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| learn.microsoft.com/en-us/azure/architecture/example-scenario/apps/scalable-apps-performance-modeling-site-reliability | Microsoft Learn<br>Scalable cloud applications and site reliability engineering architecture principles, including monitoring, performance modelling and reliability design<br>(General reference) |
| www.geeksforgeeks.org/software-engineering/site-reliability-engineering/ | GeeksforGeeks<br>Site reliability engineering concepts, monitoring, observability and incident response in cloud systems<br>(Tutorials) |
| docs.cloud.google.com/architecture/framework/reliability | Google Cloud<br>Cloud reliability framework covering resilience, redundancy and fault-tolerant system design principles<br>(General reference) |
| www.datacamp.com/blog/virtualization-in-cloud-computing | DataCamp<br>Virtualisation concepts, resource abstraction and the role of virtual machines and containers in cloud environments<br>(Article) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.ovhcloud.com/en/learn/virtualization-and-containers/ | OVHcloud<br>Virtualisation, containers and orchestration technologies including Kubernetes and container management<br>(General reference) |
| www.meegle.com/en_us/topics/cloud-monitoring/cloud-monitoring-capacity-planning | Meegle<br>Cloud monitoring and capacity planning strategies, including forecasting resource demand and performance optimisation<br>(Article) |
| network-king.net/the-complete-guide-to-data-center-capacity-planning-step-by-step-implementation/ | Network King<br>Data centre capacity planning processes, including resource allocation, power, cooling and infrastructure scaling<br>(Tutorials) |
| docs.oracle.com/en-us/iaas/Content/cloud-adoption-framework/monitoring-and-observability.htm | Oracle Cloud<br>Monitoring and observability approaches using metrics, logs and events to maintain system performance and visibility<br>(General reference) |
| www.eon.io/blog/cloud-security-compliance | Eon<br>Cloud security compliance requirements covering GDPR, ISO standards and shared responsibility in cloud environments<br>(Article) |
| www.eccouncil.org/cybersecurity-exchange/incident-handling/cloud-incident-response-best-practices/ | EC-Council Cybersecurity Exchange<br>Cloud incident response life cycle, including detection, containment, recovery and post-incident review practices<br>(Tutorials) |

**Journals and articles**

Long, S. *et al*. (2022) 'A review of energy efficiency evaluation technologies in cloud data centers', *Energy and Buildings*. Available at: doi.org/10.1016/j.enbuild.2022.111848.

Mytton, D. *et al*. (2023) 'Stretched grid? Managing data center energy demand and grid capacity', *Oxford Open Energy*, 2, oiad014. Available at: doi.org/10.1093/ooenergy/oiad014.

Rajesh, Y.S., Kiran Kumar, V.G. and Poojary, A. (2024) 'A Unified Approach Toward Security Audit and Compliance in Cloud Computing', *Journal of the Institution of Engineers (India): Series B*, 105(3), pp. 733–750. Available at: doi.org/10.1007/s40031-024-01034-x.

Xiahou, X. *et al*. (2022) 'Research on Safety Resilience Evaluation Model of Data Center Physical Infrastructure: An ANP-Based Approach', *Buildings*, 12(11), 1911. Available at: doi.org/10.3390/buildings12111911.

**Links**

This unit links to the following related units:

*Unit 519: Cloud System Architecture.*

# Unit 519: Cloud System Architecture

**Unit code:** R/652/3305

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Modern organisations rely heavily on cloud-based systems to deliver services that are scalable, flexible and accessible across a wide range of sectors. As cloud adoption continues to increase, the focus has moved beyond simply using cloud services towards designing well-structured systems that can meet performance, security and cost requirements. Understanding how cloud systems are put together is therefore an essential skill for those working in computing and IT infrastructure roles.

The aim of this unit is to develop students' knowledge and understanding of cloud systems architecture and the principles that underpin effective design. Students will explore how different components and services work together within cloud environments, and how architectural decisions are influenced by requirements such as scalability, resilience and performance. The unit also introduces the importance of selecting appropriate design approaches to meet specific organisational needs.

This unit covers a broad range of topics that support the design and evaluation of cloud architectures. Students will study core architectural principles, distributed systems concepts and virtualisation technologies. They will examine how key infrastructure components such as compute, storage and networking interact, and how scalability and resilience can be achieved through effective design. The unit also explores architectural patterns, requirement analysis, optimisation strategies and approaches to presenting design solutions to different audiences.

On successful completion, students will be able to analyse cloud architecture principles, design scalable and reliable solutions, and communicate their designs clearly to stakeholders. These skills are directly applicable to a range of roles within the cloud and infrastructure domain, including Cloud Architect, Solutions Architect, DevOps Engineer, Cloud Engineer and Infrastructure Engineer.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core principles of cloud systems architecture

LO2 Explore the interactions of key cloud infrastructure components in supporting scalable and resilient systems

LO3 Design a scalable and reliable cloud architecture to meet the needs of an identified scenario

LO4 Present the design of the scalable and reliable cloud architecture to a range of industry stakeholders.

## Essential Content

LO1 **Explain the core principles of cloud systems architecture**

*Cloud architecture principles:*

Design principles: service abstraction e.g. Application Programming Interface (API) gateway abstraction, service mesh layers; separation of concerns e.g. application tier segregation, domain-driven service boundaries; loose coupling e.g. stateless microservices, RESTful API decoupling

Scalability and resilience: elasticity e.g. autoscaling groups, Kubernetes horizontal pod autoscaling; fault tolerance e.g. retry patterns, circuit breaker patterns; redundancy e.g. multi-AZ deployment, replicated services

Availability and deployment: high availability e.g. service level agreement (SLA) targets, multi-region failover; distributed environments e.g. geo-distributed clusters, edge deployments; recovery mechanisms e.g. automated failover, disaster recovery orchestration.

*Distributed systems concepts:*

System structure: distributed architectures e.g. microservices architecture, service-oriented architecture; multiple nodes e.g. EC2 instances, Kubernetes nodes; distributed processing e.g. parallel task execution

Consistency and data behaviour: data consistency models e.g. strong consistency, eventual consistency; replication strategies e.g. leader-follower, multi-master; partition tolerance e.g. consistency, availability and partition tolerance (CAP) considerations

Performance factors: latency, e.g. round-trip latency, service latency; throughput, e.g. request rates, data transfer rates; trade-offs, e.g. latency vs throughput, cost vs performance

Failure conditions: unreliable networks e.g. packet loss, partitioning; timing issues e.g. timeouts, clock drift; partial failure, e.g. node failure, service outage; failure isolation, e.g. fault domains, availability zones; cascading failure, e.g. dependency failure, service chain impact

Distributed co-ordination: co-ordination e.g. service synchronisation; consensus e.g. leader election.

*Virtualisation and resource abstraction:*

Virtualisation models: hypervisor-based e.g. Kernel-based Virtual Machine (KVM), VMware ESXi; container-based e.g. Docker; shared infrastructure e.g. multi-tenant compute clusters

Resource abstraction: compute e.g. virtual machines, containers; storage e.g. block, object; networking e.g. virtual private clouds (VPCs), subnet segmentation

Resource behaviour: resource pooling e.g. shared compute pools; dynamic allocation e.g. autoscaling provisioning; workload isolation e.g. container isolation.

*Cloud deployment and service models:*

Service model implications: control boundaries e.g. provider vs customer; abstraction levels e.g. infrastructure, platform; service constraints e.g. platform limits.

*Multi-tenancy and optimisation:*

Shared models: multi-tenancy e.g. Software as a Service (SaaS) platforms; tenant isolation e.g. identity and access management (IAM), network isolation

Resource usage: allocation e.g. quotas; utilisation e.g. capacity metrics

Optimisation: performance optimisation e.g. caching strategies; cost optimisation e.g. reserved or spot usage.

LO2 **Explore the interactions of key cloud infrastructure components in supporting scalable and resilient systems**

*System interaction patterns:*

Service communication: REST API calls, e.g. Hypertext Transfer Protocol (HTTP) endpoints; gRPC service calls, e.g. protobuf-based services; synchronous communication, e.g. request-response patterns, blocking HTTP calls; asynchronous communication, e.g. message queues, event streaming

Interaction models: event-driven architectures, e.g. event sourcing, publish–subscribe models; messaging systems, e.g. Kafka brokers, RabbitMQ queues; workflow orchestration, e.g. orchestration engines, DAG workflows

Interaction resilience patterns: circuit breaker, e.g. failure isolation thresholds; bulkhead pattern, e.g. resource partitioning; queue-based load levelling, e.g. buffering queues

System relationships: component interaction, e.g. microservice dependencies; service co-ordination, e.g. service registries.

*Scalability and performance behaviour:*

Scaling behaviour: horizontal scaling, e.g. autoscaling groups; vertical scaling, e.g. instance resizing; elasticity, e.g. dynamic provisioning

Load and distribution: load balancing, e.g. application load balancers; workload distribution, e.g. task schedulers; demand patterns, e.g. peak traffic

Emergent behaviour: bottlenecks, e.g. CPU bottlenecks; scaling limits, e.g. capacity ceilings; unpredictable demand, e.g. traffic spike.

*Resilience and fault handling:*

Recovery mechanisms: failover, e.g. Domain Name System (DNS) failover; retry logic, e.g. exponential backoff; fallback strategies, e.g. cached responses

Resilience patterns: redundancy, e.g. replicated instances; self-healing systems, e.g. container rescheduling; monitoring signals, e.g. health checks.

*Orchestration and workload management:*

Orchestration tools: container orchestration platforms, e.g. Kubernetes clusters; configuration management, e.g. Terraform scripts; service discovery mechanisms, e.g. DNS-based discovery

Workload control: scheduling, e.g. pod scheduling; placement strategies, e.g. affinity rules; distributed workloads, e.g. cluster workloads

Automation: monitoring, e.g. monitoring dashboards; recovery processes, e.g. auto-restart.

*Trade-offs and architectural decisions:*

Design trade-offs: performance, e.g. latency vs throughput; resilience, e.g. availability vs complexity

Scalability decisions: scaling limits, e.g. resource limits; cost–performance trade-offs, e.g. efficiency vs cost

System optimisation: performance optimisation, e.g. tuning parameters; efficiency, e.g. utilisation efficiency.

LO3 **Design a scalable and reliable cloud architecture to meet the needs of an identified scenario**

*Architecture design approaches:*

Architecture patterns: layered architecture, e.g. N–tier architectures, hexagonal architecture; microservices architecture, e.g. API gateway patterns, service mesh patterns; serverless architecture, e.g. function-as-a-service, managed backend services; event-driven architecture, e.g. publish–subscribe systems, event streaming platforms

Architecture structures: layered architectures, e.g. multi-tier applications, application layers; distributed architectures, e.g. service clusters, distributed nodes; hybrid architectures, e.g. hybrid cloud systems, multi-environment deployments

Design alignment: requirement traceability, e.g. business alignment; architecture selection, e.g. pattern selection, architecture decisions.

*Requirements and constraints:*

Requirement types: functional requirements, e.g. service endpoints, application features; non-functional requirements, e.g. latency targets, availability SLAs; business requirements, e.g. organisational KPIs, service objectives

Constraint factors: constraints, e.g. budget limits, resource quotas; compliance requirements, e.g. regulatory requirements, standards; system limitations, e.g. platform limits, service quotas

Priority and trade-offs: requirement prioritisation, e.g. MoSCoW prioritisation, backlog prioritisation; trade-offs, e.g. cost vs performance.

*Performance and reliability design:*

Performance factors: latency considerations, e.g. response time targets, network latency; throughput considerations, e.g. request rates, data throughput; caching strategies, e.g. in-memory caching, content delivery network (CDN) caching

Reliability mechanisms: fault tolerance, e.g. failure recovery; high availability, e.g. multi-AZ deployment, uptime service level agreements (SLAs); resilience strategies, e.g. failover mechanisms

Scalability decisions: scaling strategies, e.g. horizontal scaling, vertical scaling; load management, e.g. load balancing, traffic distribution.

*Cost and resource optimisation:*

Cost models: pricing models, e.g. pay-as-you-go, reserved instances; cost structures, e.g. compute costs, storage costs

Optimisation strategies: cost optimisation, e.g. rightsizing, reserved capacity; performance optimisation, e.g. autoscaling tuning

Decision impact: cost-performance trade-offs, e.g. performance vs cost; scalability–cost balance, e.g. scaling cost, resource cost.

*Industry best practice and governance:*

Access and identity control: identity and access management, e.g. identity and access management (IAM) roles, policy-based access; authentication, e.g. multi-factor authentication, identity providers; authorisation, e.g. role-based access control

Protection measures: data protection, e.g. encryption; segmentation, e.g. network segmentation, security groups

Governance considerations: compliance, e.g. regulatory compliance, audit standards; shared responsibility, e.g. provider model, customer model.

*Design justification and evaluation:*

Design reasoning: design decisions, e.g. architecture choices, service selection; rationale, e.g. design justification, trade-off reasoning; assumptions, e.g. workload assumptions

Evaluation factors: trade-offs, e.g. performance, cost; strengths and limitations, e.g. scalability benefits, platform constraints

Alignment verification: requirements alignment, e.g. requirement traceability; scenario suitability, e.g. use-case alignment, architecture suitability.

LO4 **Present the design of the scalable and reliable cloud architecture to a range of industry stakeholders**

*Professional communication approaches:*

Audience and messaging: technical language, e.g. architecture terminology, service definitions; structured presentation, e.g. logical architecture flows, layered structure; information structure, e.g. hierarchical content, system decomposition.

*Presentation artefacts:*

Visual artefacts: architecture diagrams, e.g. component diagrams, deployment diagrams; system models, e.g. service models, infrastructure models; component interaction, e.g. data flow diagrams, sequence diagrams

Documentation outputs: written documentation, e.g. technical specifications, architecture design documents; design specifications, e.g. API specifications, service contracts; reports, e.g. architecture reports, system summaries.

*Stakeholder-focused communication:*

Audience types: technical audiences, e.g. developers, cloud engineers; non-technical audiences, e.g. stakeholders, business managers; communication adaptation, e.g. abstraction levels, detail granularity

Information clarity: abstraction, e.g. high-level architecture views, conceptual diagrams; complex concepts, e.g. distributed systems behaviour, scaling models; clarity, e.g. structured explanations, layered descriptions

Business relevance: benefits, e.g. performance improvements, cost optimisation; risks, e.g. security risks, availability risks; trade-offs, e.g. cost vs performance, scalability vs complexity.

*Justification and evidencing of design decisions:*

Design rationale: design rationale, e.g. architecture decision records, design documentation; assumptions, e.g. workload assumptions, usage patterns; constraints, e.g. resource limits, service quotas

Evaluation factors: architectural trade-offs, e.g. latency vs throughput, availability vs consistency; design alignment, e.g. requirement alignment, business alignment

Evidence and validation: requirements, e.g. functional requirements, non-functional requirements; best practice, e.g. Well-Architected Framework, cloud design principles.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the core principles of cloud systems architecture</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a selection of real-world examples, how architecture principles and component interactions contribute to scalable and resilient cloud systems.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the core cloud architecture principles in relation to distributed systems.<br><br>P2 Describe, using a selection of real-world examples, the application of virtualisation, deployment and multi-tenancy models within cloud architectures.</td><td>M1 Analyse, using a selection of real-world examples, how cloud architecture principles shape design decisions in distributed environments.</td></tr>
<tr><td colspan="2">LO2 Explore the interactions of key cloud infrastructure components in supporting scalable and resilient systems</td></tr>
<tr><td>P3 Explore, using a selection of real-world examples, the interaction patterns and workload management that support scalable and resilient cloud systems.<br><br>P4 Examine, using a selection of real-world examples, the architectural decisions and trade-off considerations when designing scalable and resilient systems.</td><td>M2 Assess, using a selection of real-world examples, the effectiveness of component interactions in supporting scalability and system resilience.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a scalable and reliable cloud architecture to meet the needs of an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design of the scalable and reliable cloud architecture that enables it to comprehensively meet the needs of an identified scenario. |
| **P5** Design a cloud architecture that meets the requirements of an identified scenario.<br>**P6** Review the design of the cloud architecture to identify strengths and limitations in meeting the needs of an identified scenario. | **M3** Evaluate how effectively the architecture design addresses the needs of an identified scenario. | |
| **LO4** Present the design of the scalable and reliable cloud architecture to a range of industry stakeholders | | |
| **P7** Present the cloud architecture design using appropriate formats for industry stakeholders to obtain feedback.<br>**P8** Review the feedback from industry stakeholders to identify further areas to improve the effectiveness of the scalability or reliability of the cloud architecture design. | **M4** Optimise, using the results of the internal and stakeholder review, the design of the scalable and reliable cloud architecture so that it comprehensively meets the needs of an identified scenario. | |

## Recommended Resources

### Textbooks

Erl, T. and Barceló Monroy, E. (2023) *Cloud Computing: Concepts, Technology, Security, and Architecture.* 2nd edn. Boston: Pearson.

Kleppmann, M. and Riccomini, C. (2026) *Designing Data-Intensive Applications*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Morris, K. (2025) *Infrastructure as Code: Designing and Delivering Dynamic Systems for the Cloud Age*. 3rd edn. Sebastopol, CA: O'Reilly Media.

Richards, M. and Ford, N. (2025) *Fundamentals of Software Architecture*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Ruparelia, N.B. (2023) *Cloud Computing*. Revised edn. Cambridge, MA: MIT Press.

Surianarayanan, C. and Chelliah, P.R. (2023) *Essentials of Cloud Computing: A Holistic, Cloud-Native Perspective*. Cham: Springer.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/ | Microsoft Azure<br>Cloud architecture styles including layered, microservices and event-driven approaches, with clear explanations of design principles<br>(General reference) |
| www.ibm.com/think/topics/distributed-cloud | IBM Cloud<br>Overview of distributed systems concepts including scalability, communication and fault tolerance in cloud environments<br>(General reference) |
| cloud.google.com/architecture/framework | Google Cloud<br>Cloud architecture framework covering design principles, reliability, performance and operational excellence<br>(General reference) |

| | |
|---|---|
| www.redhat.com/en/topics/cloud-computing/what-is-cloud-architecture | Red Hat<br>Explanation of cloud architecture components, infrastructure layers and architectural design approaches<br>(General reference) |
| www.geeksforgeeks.org/distributed-systems/distributed-systems-tutorial/ | GeeksforGeeks<br>Structured tutorial covering distributed systems concepts including communication, scalability and co-ordination between components<br>(Tutorial) |
| uptimerobot.com/knowledge-hub/devops/cloud-infrastructure-guide/ | UptimeRobot<br>Comprehensive guide to cloud infrastructure components and how they support performance, scalability and reliability<br>(Article) |
| martinfowler.com/articles/microservices.html | Martin Fowler<br>Detailed explanation of microservices architecture and how services interact within scalable distributed systems<br>(Article) |
| www.techtarget.com/searchcloudcomputing/definition/cloud-infrastructure | TechTarget<br>Explanation of cloud infrastructure components including compute, storage and networking and how they interact<br>(Article) |

### Journals and articles

Gundla, N.K. (2024) 'Building resilient and scalable cloud infrastructure architectures', *International Journal of Computer Trends and Technology*, 72(9), pp. 77–92. Available at: doi.org/10.14445/22312803/IJCTT-V72I9P113.

Perumal, A.P. (2025) 'Designing and Implementing a Scalable and Secure Cloud Infrastructure for Real-Time Data Processing in High-Performance Analytics', in *Lecture Notes in Networks and Systems*, 1430, pp. 451–465. Available at: doi.org/10.1007/978-981-96-6883-0_31.

Kingsley, M.S. (2023) 'Well-Architected Framework', in *Cloud Technologies and Services*. Cham: Springer. Available at: doi.org/10.1007/978-3-031-33669-0_5.

Nouhas, H., Belangour, A. and Nassar, M. (2023) 'Cloud and Edge Computing Architectures: A Survey', in *2023 IEEE 11th Conference on Systems*, Process & Control (ICSPC), Malacca, Malaysia, pp. 210–215. Available at: doi.org/10.1109/ICSPC59664.2023.10420123.

### Links

This unit links to the following related units:

*Unit 518: Cloud Data Centre Operations.*

# Unit 520: Applied AI

**Unit code:** **T/652/3306**

**Unit level:** **5**

**Credit value:** **15**

---

## Introduction

Artificial intelligence (AI) now underpins many modern digital systems, influencing how organisations analyse data, support decision-making, and streamline operations. AI technologies are increasingly embedded across sectors such as business, healthcare, finance, manufacturing and public services, making applied AI knowledge a valuable part of contemporary computing practice. As organisations continue to adopt AI-enabled tools, there is a growing need for practitioners who can understand how these systems work in realistic settings and who can contribute effectively to their development and use.

The aim of this unit is to provide students with a solid grounding in applied artificial intelligence. It focuses on how AI models are selected, implemented and interpreted within real-world contexts rather than on theoretical or abstract methods. The unit encourages students to think critically about the suitability of different AI approaches, the data required to support them, and the practical considerations that influence successful implementation.

Throughout the unit, students will explore core principles of applied AI, including learning paradigms, data preparation, feature engineering and workflow design. They will examine a range of AI model types commonly used in industry, analyse their strengths and limitations, and apply selected techniques to an identified scenario. The unit also develops students' ability to communicate AI results clearly, using performance metrics, data visualisation and storytelling approaches appropriate to different audiences.

On completion of this unit, students will have obtained the knowledge and practical skills needed to contribute to AI-enabled projects in a professional environment. They will be able to apply core AI techniques, evaluate model performance and present findings with confidence. These skills support progression into roles such as AI Technician, Data Analyst, Junior AI Engineer, Automation Specialist, Insights Analyst or Digital Transformation Practitioner, as well as into further study within computing and data-driven disciplines.

 © Pearson Education Limited 2026

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the foundational principles of applied AI in a range of organisational contexts

LO2 Explore the different models used in applied AI

LO3 Implement a range of applied AI techniques for an identified scenario

LO4 Present the results of the AI model to an identified audience.

## Essential Content

LO1 **Explain the foundational principles of applied AI in a range of organisational contexts**

*Principles of applied AI:*

Supervised learning: labels e.g. classification outputs, regression targets; training data e.g. feature matrices, historical records; target values e.g. output categories, prediction variables

Unsupervised learning: clusters e.g. segment groups, similarity groups; patterns e.g. hidden structures, behavioural signatures; unlabelled data e.g. raw observations, exploratory datasets

Data types: labelled data e.g. annotated samples, verified records; unlabelled data e.g. raw inputs, unsupervised collections; input features e.g. numeric attributes, derived signals

Model generalisation: generalisation e.g. predictive stability, cross-domain robustness; overfitting e.g. noise memorisation, variance inflation; underfitting e.g. low complexity, representation gaps.

*Data preparation fundamentals:*

Feature engineering: feature extraction e.g. domain attributes, vector embeddings; feature selection e.g. relevance rankings, importance scores; transformations e.g. scaling methods, encoding schemes

Data preprocessing: cleaning e.g. anomaly removal, consistency checks; normalisation e.g. range scaling, feature standardisation; encoding e.g. categorical conversion, binary representation

Data quality and governance: completeness e.g. non-missing fields, dataset coverage; lineage e.g. data provenance, versioned sources; reproducibility e.g. random seeds, pipeline snapshots

Workflow stages: training workflows e.g. iteration cycles, optimisation routines; validation workflows e.g. model comparison, error checking; testing workflows e.g. final evaluation, performance summaries.

*Principles of model selection:*

Model suitability: model fit e.g. task relevance, scenario alignment; outcome relevance e.g. decision requirements, insight needs; interpretability needs e.g. rule visibility, feature contributions

Operational constraints: latency budgets e.g. response targets, real-time thresholds; availability targets e.g. uptime objectives, failover plans; resource limits e.g. hardware quotas, memory ceilings

Deployment context: environment types e.g. edge platforms, cloud services; integration points e.g. data pipelines, messaging buses; monitoring hooks e.g. drift indicators, audit logs

Risk and compliance: regulatory conditions e.g. sector standards, reporting duties; data protection e.g. retention rules, access controls; accountability measures e.g. review trails, approval gates.

*Examples of real-world AI tools:*

A range of real-world AI tools e.g.

Applied AI systems: Amazon Fraud Detector e.g. fraud detection, risk scoring; Tesla Autopilot Vision e.g. image recognition, autonomous driving; Google Photos e.g. image classification, object detection

Feature engineering and data tools: Monzo Risk Models e.g. transaction analysis, fraud prediction; Airbnb Pricing Model e.g. price prediction, demand estimation; Netflix Personalisation e.g. recommendation, user profiling

Model selection and application tools: Meta Prophet e.g. time-series forecasting, trend prediction; Criteo Models e.g. ad targeting, click prediction; IBM Watson natural language processing (NLP) e.g. text processing, language analysis

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Explore the different models used in applied AI**

*AI model families:*

Decision trees: tree structures e.g. nodes, branches; splitting criteria e.g. Gini impurity, information gain, entropy

Neural networks: network architectures e.g. multi-layer perceptrons, convolutional networks; activation functions e.g. rectified linear units (ReLU), sigmoid, hyperbolic tangent

Clustering methods: clustering algorithms e.g. k-means, hierarchical clustering; similarity measures e.g. Euclidean distance, cosine similarity, Manhattan distance

Rule-based systems: rule collections e.g. decision rules, production rules; inference engines e.g. forward chaining, backward chaining, hybrid inference.

*Model behaviour and application:*

Strengths and limitations: capability factors e.g. interpretability, scalability; constraint factors e.g. memory cost, compute demand, training time

Interpretability: explanation outputs e.g. feature importance, decision paths; documentation resources e.g. model cards, audit summaries, decision logs

Scalability: scaling behaviours e.g. data-volume tolerance, dimensionality handling; acceleration options e.g. GPU pipelines, vector engines, parallel frameworks

Typical application domains: business domains e.g. customer analytics, demand forecasting; operational domains e.g. anomaly detection, predictive maintenance, resource optimisation

Suitability for structured and unstructured data: structured data types e.g. tabular datasets, time-series signals; unstructured data types e.g. text corpora, images, audio streams.

*Model comparison factors:*

Model complexity: complexity measures e.g. parameter counts, structural depth; computational load e.g. memory footprint, compute budget, runtime demands

Performance trade-offs: performance factors e.g. accuracy expectations, stability requirements; resource constraints e.g. hardware limits, allocation budgets, inference cost

Transparency: transparency elements e.g. audit logs, model cards; traceability assets e.g. version records, lineage traces, change histories

Deployment considerations: deployment contexts e.g. cloud environments, edge platforms; integration points e.g. data pipelines, messaging buses, orchestration layers

Ethical and operational implications: ethical safeguards e.g. bias controls, fairness checks; operational safeguards e.g. drift detectors, fail-safe policies, monitoring rules.

*Examples of real-world AI tools:*

A range of real-world AI tools e.g.

Decision tree models: Azure Decision Forest e.g. classification, prediction; SageMaker XGBoost e.g. tree-based modelling, forecasting; scikit-learn DecisionTree e.g. classification, decision paths

Explainability tools: SHAP e.g. feature importance, prediction explanation; LIME e.g. local model explanations; IBM Watson OpenScale e.g. model monitoring, explainability

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Implement a range of applied AI techniques for an identified scenario**

*Applied AI implementation context:*

Operational scenario requirements: scenario elements e.g. service constraints, workflow dependencies; deployment conditions e.g. runtime limits, availability requirements, audit needs

Data pipeline characteristics: pipeline stages e.g. ingestion flows, data contracts, version checkpoints; data conditions e.g. drift indicators, imbalance patterns, noise levels

Outcome specifications: output expectations e.g. actionable insights, operational triggers, risk flags; validation assets e.g. acceptance criteria, performance thresholds.

*Implementation tools and environments:*

Industry-recognised AI tools: platform tools e.g. machine learning operations (MLOps) services, pipeline orchestrators, experiment trackers; integration tools e.g. API endpoints, model registries, automation frameworks

Programming environments: development environments e.g. notebook interfaces, integrated development environments; runtime environments e.g. Python stacks, container images, virtual environments

Libraries and frameworks: core libraries e.g. NumPy, pandas; machine-learning frameworks e.g. scikit-learn, TensorFlow, PyTorch.

*Model execution and outputs:*

Model training execution: training processes e.g. iteration cycles, optimisation routines; configuration elements e.g. hyperparameter sets, training schedules, validation splits

Inference generation: inference outputs e.g. predicted classes, probability scores; runtime behaviours e.g. response times, batch behaviours, stability indicators

Reproducibility records: reproducibility assets e.g. random seeds, versioned datasets, environment snapshots; execution records e.g. configuration logs, model artefacts, pipeline histories

Performance outputs: evaluation artefacts e.g. error summaries, validation metrics; diagnostic elements e.g. residual patterns, confusion structures, distribution shifts.

*Effectiveness considerations:*

Model suitability: suitability factors e.g. domain alignment, data compatibility, risk tolerance; operational constraints e.g. latency budgets, compute limits, deployment targets

Output relevance: relevance indicators e.g. decision value, insight clarity; alignment measures e.g. stakeholder expectations, scenario fit, requirement matches

Performance interpretation: interpretation artefacts e.g. metric summaries, stability profiles; explanation tools e.g. feature contributions, decision paths, influence scores

Scenario alignment: alignment elements e.g. requirement matches, constraint matches; validation checks e.g. quality thresholds, acceptance criteria, robustness checks

*Examples of real-world AI tools:*

A range of real-world AI tools e.g.

AI development platforms: Google Vertex AI e.g. model deployment, pipeline tools; Amazon SageMaker e.g. model training, workflows; Azure Machine Learning e.g. AI development, life cycle management

Model training tools: MLflow e.g. experiment tracking, pipelines; TensorFlow e.g. model training, deep learning; PyTorch e.g. model development, neural networks

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Present the results of the AI model to an identified audience**

*Performance interpretation and insight generation:*

Performance metrics: metric forms e.g. accuracy scores, precision values; evaluation artefacts e.g. recall measures, F1 indicators, error summaries

Error measures: error types e.g. misclassification counts, false-positive rates, false-negative rates; diagnostic patterns e.g. confusion structures, residual trends, distribution shifts

Insight generation: insight elements e.g. model limitations, decision boundaries; interpretation assets e.g. stability profiles, confidence distributions, uncertainty estimates.

*Data storytelling and visual communication:*

Data visualisation techniques: visual forms e.g. line charts, bar charts; analytical visuals e.g. heatmaps, confusion matrices, scatter plots

Narrative structure: narrative elements e.g. key findings, supporting evidence; communication assets e.g. storyline flow, insight anchors, summary statements

Clarity of messaging: clarity factors e.g. concise wording, visual simplicity; comprehension aids e.g. labels, legends, colour cues.

*Audience-focused presentation considerations:*

Audience needs: audience groups e.g. technical stakeholders, operational teams; requirement factors e.g. domain expectations, decision priorities, insight demands

Technical vs non-technical communication: communication modes e.g. plain-language summaries, detailed analytical outputs; audience adaptations e.g. abstraction levels, context framing, terminology selection

Professional presentation standards: presentation artefacts e.g. structured layouts, formatted visuals; delivery standards e.g. consistent styling, accessibility features, documentation quality.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the foundational principles of applied AI in a range of organisational contexts</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate the effectiveness of the applied AI principles and models used for an identified organisational context.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the foundational principles used when AI is applied to a range of organisational contexts.<br><br>P2 Explain, using a selection of real-world examples, the concepts of data preparation and model selection used when AI is applied to a range of organisational contexts.</td><td>M1 Assess, using a selection of real-world examples, the benefits and limitations of the use of applied AI within a range of organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Explore the different models used in applied AI</td></tr>
<tr><td>P3 Explore, using a selection of real-world examples, different model families used in applied AI, considering their applications and behaviour in a range of organisational use cases.<br><br>P4 Review, using a selection of real-world examples, the factors that inform the identification and selection of AI models in different business contexts.</td><td>M2 Analyse, using a selection of real-world examples, the suitability of the applied AI models for a range of identified business contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Implement a range of applied AI techniques for an identified scenario | | |
| **P5** Implement applied AI techniques for an identified data scenario using industry-recognised tools and programming methods.<br><br>**P6** Review the implementation of the applied AI techniques with a range of relevant stakeholders to identify and inform improvements. | **M3** Optimise, using the results of the stakeholder feedback, the implementation of the applied AI techniques to meet the requirements of the identified scenario. | **D2** Critically evaluate the effectiveness of implemented applied AI techniques in meeting the requirements of the identified scenario. |
| **LO4** Present the results of the AI model to an intended audience | | |
| **P7** Review the results of the AI model using a range of performance metrics to generate insights that are appropriate for an identified audience.<br><br>**P8** Present the insights obtained from the AI model to an intended audience, obtaining a range of relevant feedback. | **M4** Analyse the results of the audience feedback on the presentation to identify a range of strengths and areas for improvement. | **D3** Evaluate the effectiveness of the presentation in communicating the insights generated from the AI model to the intended audience. |

## Recommended Resources

### Textbooks

Davenport, T.H. and Mittal, N. (2023) *All-in On AI: How Smart Companies Win Big with Artificial Intelligence*. Boston, MA: Harvard Business Review Press.

Géron, A. (2022) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd edn. Sebastopol, CA: O'Reilly Media.

Marr, B. (2024) *Artificial Intelligence in Practice: How 50 Successful Companies Used AI to Solve Problems*. 2nd edn. Chichester: Wiley.

Yao, M. and Koidan, K. (2024) *Applied Artificial Intelligence: A Handbook for Business Leaders*. 2nd edn. Sebastopol: O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| arxiv.org/abs/2312.01164 | arXiv<br>Research literature on data storytelling and visual communication<br>(Research) |
| azure.microsoft.com/en-us/products/ai-foundry/tools/ | Microsoft Azure<br>Overview of industry-recognised AI tools and platforms<br>(General reference) |
| blog.unitlab.ai/dive-into-confusion-matrix-and-f1-score/ | Unitlab<br>Understanding confusion matrices and F1 score in applied contexts<br>(Article) |
| www.databricks.com/blog/top-ai-use-cases-transforming-industries-2025 | Databricks<br>Industry-focused applied AI use cases across sectors<br>(Article) |
| www.geeksforgeeks.org/artificial-intelligence/introduction-to-applied-ai | GeeksforGeeks<br>Introduction to applied artificial intelligence concepts and workflows<br>(General reference) |

 © Pearson Education Limited 2026

| | |
|---|---|
| mljourney.com/common-metrics-for-evaluating-classification-models/ | ML Journey<br>Explanation and interpretation of AI model performance metrics<br>(Article) |
| www.kaggle.com/learn/intro-to-machine-learning | Kaggle Learn<br>Intro to Machine Learning course including data exploration, basic modelling, validation and overfitting<br>(Tutorials) |
| github.com/microsoft/AI-For-Beginners | GitHub – Microsoft AI for Beginners<br>Open educational repository containing lessons, datasets, examples and multi-language support for AI fundamentals<br>(Development tool) |
| inst.eecs.berkeley.edu/~cs188/textbook/ | UC Berkeley CS188<br>Freely accessible AI textbook covering search, problem solving, reasoning and introductory machine-learning concepts<br>(E-books) |

**Journals and articles**

Miller, C. *et al*. (2024) 'A review of model evaluation metrics for machine learning', *Frontiers in Bioinformatics*, 4. Available at: doi.org/10.3389/fbinf.2024.1457619.

Naidu, G., Zuva, T. and Sibanda, E.M. (2023) 'A Review of Evaluation Metrics in Machine Learning Algorithms', *Lecture Notes in Networks and Systems*, 724, pp. 15–25. Available at: doi.org/10.1007/978-3-031-35314-7_2.

Schröder, K. *et al*. (2023) 'Telling stories with data – A systematic review', *arXiv*. Available at: arxiv.org/abs/2312.01164.

Verdonck, T. *et al*. (2024) 'Special issue on feature engineering', *Machine Learning*, 113, pp. 3917–3928. Available at: doi.org/10.1007/s10994-021-06042-2.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science (Pearson-set)*

*Unit 420: Core AI*

*Unit 421: AI Automation and Digital Productivity*

*Unit 423: Agentic AI.*

 © Pearson Education Limited 2026

# Unit 521: Generative and Conversational AI

**Unit code:** Y/652/3307

**Unit level:** 5

**Credit value:** 15

---

## Introduction

The rapid growth of artificial intelligence has brought generative and conversational models into everyday digital systems, from productivity tools to customer-facing applications. These technologies are reshaping how organisations design services, automate interactions and handle information. As these models become more widely integrated across sectors, it is increasingly important for students to understand how they work and the considerations involved in developing and using them responsibly.

The aim of this unit is to provide students with a clear and structured understanding of generative and conversational AI. The unit focuses on the core principles that underpin these models, the methods used to train and optimise them, and the practical steps required to apply them in real scenarios. Students will gain awareness of both the capabilities and the limitations of current AI systems, preparing them to make informed decisions about their use.

The unit covers the key topics needed to develop a working knowledge of generative and conversational AI. Students will investigate foundational concepts, including data representation and model architectures, before exploring training, fine-tuning and optimisation approaches. They will then design and develop an AI application for a specified scenario and evaluate its performance. The unit emphasises practical understanding supported by structured theoretical knowledge.

On successful completion of this unit, students will be able to explain how generative and conversational models operate, implement core development methods and assess the effectiveness of an AI solution in context. These skills are relevant to a range of emerging roles, including AI Technician, Machine Learning Assistant, Data Engineering Support Officer, Junior AI Developer, AI Product Assistant and Digital Solutions Analyst.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Examine the core principles of generative and conversational AI models

LO2 Review the methods for training, fine-tuning and optimising generative and conversational AI models

LO3 Develop a generative or conversational AI application that meets the requirements of an identified scenario

LO4 Review the effectiveness of the developed AI application for the identified scenario.

 © Pearson Education Limited 2026

## Essential Content

LO1 **Examine the core principles of generative and conversational AI models**

*Foundations of generative and conversational AI:*

AI types: generative AI e.g. content generation, pattern learning, text production; conversational AI e.g. dialogue systems, virtual assistants, chatbot interfaces; examples: Google Assistant, Microsoft Copilot, Amazon Alexa

Learning paradigms: supervised learning e.g. labelled datasets, training targets, classification tasks; unsupervised learning e.g. clustering methods, pattern grouping, representation discovery

Data concepts: data-driven learning e.g. dataset analysis, pattern extraction, model training.

*Model architectures and mechanisms:*

Neural architectures: transformers e.g. attention layers, sequence handling, parallel processing; encoder–decoder structures e.g. input encoding, output generation, sequence mapping; examples: GPT-4 architecture, Google Bidirectional Encoder Representations from Transformers (BERT), T5 Transformer model

Core mechanisms: attention mechanisms e.g. relevance scoring, context weighting, alignment processes; tokenisation e.g. text splitting, token mapping, vocabulary processing; embeddings e.g. semantic vectors, feature mapping, representation encoding.

*Language modelling theory:*

Modelling approaches: statistical modelling e.g. probability estimation, frequency analysis, distribution modelling; sequence prediction e.g. continuation forecasting, structured output estimation

Representation concepts: contextual representation e.g. semantic meaning, relational cues, contextual links

Prediction elements: next-token prediction e.g. probabilistic selection, output estimation, sequence continuation; examples: ChatGPT response generation, GitHub Copilot code completion, Google Smart Compose.

*Capabilities and limitations:*

Performance factors: generalisation e.g. unseen input handling, pattern extension, knowledge transfer; scalability e.g. resource expansion, performance scaling; computational requirements e.g. memory needs, processing load

Risk-related factors: hallucination e.g. incorrect responses, fabricated details, unreliable outputs; bias e.g. dataset imbalance, representation skew; examples: ChatGPT hallucination cases, facial recognition bias studies, automated CV screening tools

Model behaviour: explainability e.g. transparency, reasoning insights, feature attribution; model limitations e.g. capability gaps, performance boundaries, practical restrictions.

*Examples of real-world AI applications:*

A range of real-world AI tools e.g.

AI systems and applications: Google Assistant e.g. voice interaction, task automation; Copilot e.g. content generation, productivity support; Alexa e.g. smart home control, voice queries

Model architectures: GPT-4 e.g. text generation, language modelling; BERT e.g. text understanding, search relevance; T5 e.g. text transformation, sequence tasks

Language modelling tools: ChatGPT e.g. response generation, conversation; GitHub Copilot e.g. code completion, suggestions; Google Smart Compose e.g. text prediction, email writing

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Review the methods for training, fine-tuning and optimising generative and conversational AI models**

*Training approaches:*

Training processes: pre-training e.g. large-scale learning, pattern extraction, representation building; data preparation e.g. data cleaning, formatting, preprocessing; data quality e.g. accuracy checks, consistency review, completeness assessment

Dataset factors: large-scale datasets e.g. diverse inputs, broad examples, varied topics; dataset bias e.g. unbalanced classes, skewed representation, missing groups; infrastructure requirements e.g. compute resources, storage needs, processing capacity.

*Fine-tuning methods:*

Adaptation techniques: supervised fine-tuning e.g. labelled examples, guided improvements, targeted corrections; transfer learning e.g. reusing learned features, adapting knowledge, extending capability; domain adaptation e.g. scenario matching, terminology alignment, context tuning

Alignment methods: instruction tuning e.g. task-specific prompts, behavioural guidance, improved response format; alignment techniques e.g. safety tuning, output refinement, behaviour adjustment.

*Optimisation techniques:*

Model optimisation: hyperparameters e.g. learning rate, batch size, training steps; loss functions e.g. error measurement, performance comparison, output evaluation; optimisation algorithms e.g. gradient descent methods, update rules, weight adjustments

Efficiency methods: regularisation e.g. overfitting reduction, stability improvement, performance smoothing; model compression e.g. size reduction, lightweight deployment, faster inference; inference optimisation e.g. speed improvement, resource efficiency, latency reduction.

*Performance evaluation considerations:*

Evaluation metrics: accuracy measures e.g. correctness, match rate, output precision; relevance measures e.g. meaningful responses, topic alignment, contextual fit; coherence measures e.g. logical flow, stable structure, clear meaning; examples: bilingual evaluation understudy (BLEU) score evaluation, recall-oriented understudy for gisting evaluation (ROUGE) metrics, human evaluation of ChatGPT responses

Assessment factors: robustness e.g. error resistance, adaptability, reliable outputs; efficiency trade-offs e.g. speed vs. accuracy, cost vs. performance, resource balancing; ethical considerations e.g. fairness checks, harm reduction, responsible use; safety considerations e.g. harmful output prevention, safe boundaries, risk mitigation

*Real-world examples:*

A range of real-world AI tools e.g.

Training tools: OpenAI GPT pipelines e.g. large-scale training, pattern learning; Google BERT training e.g. language modelling, data training; Meta Llama tools e.g. model training, dataset processing

Fine-tuning tools: Azure OpenAI Studio e.g. model tuning, prompt adjustment; Hugging Face Transformers e.g. model fine-tuning, dataset training; OpenAI fine-tuning API e.g. model adaptation, task improvement

Optimisation tools: TensorFlow e.g. model optimisation, tuning; PyTorch Lightning e.g. training workflows, performance tuning; SageMaker tuning jobs e.g. hyperparameter optimisation, model improvement

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Develop a generative or conversational AI application that meets the requirements of an identified scenario**

*Application design considerations:*

Design requirements: problem definition e.g. scenario needs, task scope, target outcomes; scenario identification e.g. context setting, user environment, goal identification; user requirements e.g. user expectations, input needs, interaction goals

Solution parameters: functional scope e.g. key features, core actions, essential behaviours; constraints e.g. time limits, resource limits, delivery restrictions; assumptions e.g. expected inputs, system conditions, typical use patterns.

*Model selection and configuration:*

Model choice: model criteria e.g. suitability, performance needs, resource match; architecture suitability e.g. language use, task alignment, complexity fit; model selection e.g. capability match, output type, scenario appropriateness

Configuration factors: parameter configuration e.g. model settings, output control, processing limits; integration requirements e.g. system connection, platform fit, compatibility needs; resource considerations e.g. hardware needs, compute cost, memory use.

*Prototyping and implementation:*

Development steps: application workflows e.g. input flow, processing steps, output delivery; system integration e.g. linking components, application programming interface (API) connection, environment setup; Interaction design e.g. user prompts, response handling, interface behaviour

Deployment process: deployment environments e.g. cloud setup, local runtime, platform choice; testing approaches e.g. sample testing, interaction trials, error checks; implementation steps e.g. coding tasks, configuration setup, system assembly.

*Performance characteristics:*

Output quality: output relevance e.g. meaningful responses, topic alignment, appropriate detail; response consistency e.g. stable structure, predictable behaviour, reliable tone

System behaviour: error handling e.g. fallback responses, invalid input management, safe outputs; latency considerations e.g. response delay, processing time, loading impact; system responsiveness e.g. smooth interaction, immediate feedback, reliable updates

Bias handling and mitigation: bias identification e.g. skewed outputs, unequal treatment, inconsistent outcomes; bias sources e.g. sampling bias, data gaps, representational imbalance; fairness checks e.g. output comparison, pattern review, disparity detection; mitigation actions e.g. data adjustment, rule refinement, human review; ongoing monitoring e.g. bias tracking, feedback loops, periodic review.

*Examples of real-world applications:*

A range of real-world AI tools e.g.

AI model platforms: GPT-4 e.g. text generation, chatbot models; Google Gemini e.g. multimodal tasks, conversational AI; Meta Llama e.g. local models, language processing

Prototyping and implementation tools: OpenAI API chatbot prototypes e.g. conversational workflows, response handling; Azure Bot Service implementations e.g. chatbot deployment, system integration; Slack AI integrations e.g. messaging automation, workflow support

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Review the effectiveness of the developed AI application for the identified scenario**

*Effectiveness evaluation criteria:*

Application fit: purpose alignment e.g. scenario fit, user relevance, task appropriateness; reliability e.g. stable outputs, repeatable behaviour, dependable performance; consistency e.g. predictable responses, uniform structure, steady behaviour

Output quality: output quality e.g. clear content, meaningful information, appropriate detail; scenario fit e.g. correct context, relevant response, appropriate behaviour; performance suitability e.g. responsiveness, accuracy, task match.

*Testing and validation:*

Testing types: functional testing e.g. correct behaviour, expected outputs, feature accuracy; performance testing e.g. speed checks, reliability checks, workload handling; reliability considerations e.g. stable responses, consistent actions, repeatable outcomes

Validation activities: limitation identification e.g. weak outputs, missing cases, performance gaps; refinement processes e.g. adjustments, improvements, issue fixes; validation checks e.g. scenario fit, requirement confirmation, quality assurance.

*Evaluation metrics and methods:*

Measurement approaches: qualitative metrics e.g. user impressions, clarity assessment, perceived usefulness; quantitative metrics e.g. accuracy scores, frequency measures, performance counts; benchmarking e.g. comparison models, reference tasks, standard checks

Assessment methods: comparative analysis e.g. model comparison, scenario review, difference identification; evaluation tools e.g. testing frameworks, scoring guides, assessment templates; assessment methods e.g. structured reviews, procedural testing, criteria-based judgments.

*Strengths and limitations:*

Strength identification: strengths e.g. effective outputs, helpful behaviour, accurate responses; performance limitations e.g. slow responses, low accuracy, restricted cases

Challenge areas: ethical issues e.g. fairness concerns, sensitive content risks, safe usage; usability challenges e.g. unclear interaction, confusing outputs, difficult prompts; weaknesses e.g. inconsistent handling, missing features, limited range; improvement needs e.g. refinement tasks, added features, better guidance.

*Improvement considerations:*

Enhancement options: optimisation opportunities e.g. speed improvements, accuracy gains, efficiency tuning; refinement strategies e.g. response adjustments, prompt improvements, behaviour tuning

Future development: scalability considerations e.g. handling more users, increased load, wider deployment; enhancement potential e.g. new capabilities, improved usability, broader coverage; future improvements e.g. iterative upgrades, expanded functions, ongoing refinement; development pathways e.g. update plans, staged improvements, long-term enhancements.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Examine the core principles of generative and conversational AI models</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate the effectiveness of generative and conversational AI theories and concepts in the development of an optimised AI model for a given organisational context.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the foundational principles and architectures used by generative and conversational AI models, including their capabilities and limitations.<br><br>P2 Review, using a selection of real-world examples, the language modelling concepts used by generative and conversational AI models.</td><td>M1 Analyse, using a selection of real-world examples, the core principles of generative and conversational AI models in relation to their use in organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Review the methods for training, fine-tuning and optimising generative and conversational AI models</td></tr>
<tr><td>P3 Review, using a selection of real-world examples, methods used for training and fine-tuning generative and conversational AI models.<br><br>P4 Discuss, using a selection of real-world examples, methods used for optimising and evaluating the performance of generative and conversational AI models.</td><td>M2 Analyse, using a selection of real-world examples, the effectiveness of training, fine-tuning, and optimisation methods used in generative and conversational AI models.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Develop a generative or conversational AI application that meets the requirements of an identified scenario | | **LO3 and LO4**<br>**D2** Justify how design and development decisions contribute to the effectiveness of the generative or conversational AI application in meeting the identified scenario requirements. |
| **P5** Design a generative or conversational AI application that meets the requirements of an identified scenario.<br>**P6** Develop a generative or conversational AI application that meets the requirements of an identified scenario. | **M3** Assess how effectively the selected model and its configuration meet the requirements of the identified generative or conversational AI application scenario. | |
| **LO4** Review the effectiveness of the developed AI application for the identified scenario | | |
| **P7** Implement, using a range of established metrics, a test plan to determine the performance of the developed generative or conversational AI application against key requirements.<br>**P8** Review, with a range of organisation stakeholders, the effectiveness of the developed generative or conversational AI application against the identified scenario requirements. | **M4** Optimise, using the results of the testing and review, the performance of the AI program, identifying strengths and limitations. | |

## Recommended Resources

### Textbooks

Alammar, J. and Grootendorst, M. (2024) *Hands-On Large Language Models: Language Understanding and Generation*. Sebastopol, CA: O'Reilly Media.

Dhamani, N. and Engler, M. (2024) *Introduction to Generative AI*. New York: Simon & Schuster.

Foster, D. (2023) *Generative Deep Learning*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Gupta, R., Tiwari, S. and Chaudhary, P. (2025) *Generative AI: Techniques, Models and Applications*. Cham: Springer Nature.

McTear, M. (2022) *Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots*. Cham: Springer Nature.

Phoenix, J. and Taylor, M. (2024) *Prompt Engineering for Generative AI: Future-Proof Inputs for Reliable AI Outputs*. Sebastopol, CA: O'Reilly Media.

Sanseviero, O. *et al*. (2024) *Hands-On Generative AI with Transformers and Diffusion Models*. Sebastopol, CA: O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| huggingface.co | Hugging Face<br>Open-source platform providing transformer models, datasets and documentation for generative and conversational AI development<br>(Resource) |
| cdn.openai.com/papers/Training_language_models_to_follow_instructions_with_human_feedback.pdf | OpenAI Research<br>Training language models to follow instructions with human feedback<br>(PDF) |
| www.frontiersin.org/journals/artificial-intelligence | Frontiers in Artificial Intelligence<br>Peer-reviewed journal covering conversational AI, language models and human–AI interaction<br>(Research Hub) |

| | |
|---|---|
| arxiv.org/abs/2401.14423 | arXiv<br>Prompt Design and Engineering: Introduction and Advanced Methods<br>(Research Article) |
| arxiv.org/pdf/1706.03762v4 | arXiv<br>Attention Is All You Need<br>(PDF) |
| www.ibm.com/academic/ | IBM<br>Academic Computing & AI Learning Portal<br>(Program Page) |
| www.mdpi.com/journal/futureinternet | MDPI – Future Internet<br>Open-access journal publishing research on generative AI, large language models and conversational systems<br>(Journal) |

## Journals and articles

Betzalel, E., Penso, C. and Fetaya, E. (2024) 'Evaluation Metrics for Generative Models: An Empirical Study', *Machine Learning and Knowledge Extraction*, 6(3), pp. 1531–1544. Available at: doi.org/10.3390/make6030073.

Kishore, Y. (2024) 'Optimizing Enterprise Conversational AI: Accelerating Response Accuracy with Custom Dataset Fine-Tuning', *Intelligent Information Management*, 16, pp. 65–76. Available at: doi.org/10.4236/iim.2024.162005.

Roumeliotis, K.I. and Tselikas, N.D. (2023) 'ChatGPT and Open-AI Models: A Preliminary Review', *Future Internet*, 15(6), Article 192. Available at: doi.org/10.3390/fi15060192.

Villa, L. *et al*. (2023) 'Conversational Agent Development Through Large Language Models: Approach with GPT', in *Proceedings of the 15th International Conference on Ubiquitous Computing and Ambient Intelligence* (UCAmI 2023). Cham: Springer. Available at: doi.org/10.1007/978-3-031-48306-6_29.

## Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science (Pearson-set)*

*Unit 420: Core AI*

*Unit 421:  AI Automation and Digital Productivity*

*Unit 423: Agentic AI*

*Unit 520: Applied AI.*

# Unit 522: Machine Learning Concepts

**Unit code:** **A/652/3308**

**Unit level:** **5**

**Credit value:** **15**

---

## Introduction

Machine learning now sits at the heart of many modern technologies, from everyday digital services to advanced systems used across industry. Organisations increasingly rely on data-driven tools to support decision-making, automate routine tasks and identify patterns that would be difficult to detect manually. As these technologies continue to develop, there is a growing need for professionals who understand how machine learning models work and how they can be applied responsibly in real-world contexts.

The aim of this unit is to give students a clear and structured understanding of the principles that underpin machine learning. It focuses on the key ideas that explain how models learn from data, how mathematical foundations influence their behaviour and performance, and why ethical and responsible practice is essential when these systems are deployed. Students will develop the theoretical grounding needed to interpret, explain and work effectively with machine learning approaches.

The unit introduces students to the core concepts that shape machine learning, including learning paradigms, data characteristics, and model generalisation. It explores the mathematical ideas that support model development, such as probability, statistics and optimisation, before examining classical machine learning algorithms and how they are selected for particular problem types. Ethical, legal and responsible AI considerations are integrated throughout, helping students recognise the risks, constraints and safeguards required when building or applying machine learning solutions.

On completion of this unit, students will be able to explain key machine learning concepts, interpret the mathematical principles behind model behaviour and review a range of classical algorithms. These skills support progression into a variety of computing roles that make use of data-driven technologies, including Data Analyst, Machine Learning Technician, Junior Machine Learning Engineer, AI Support Specialist, Systems Developer and Technology Consultant.

 © Pearson Education Limited 2026

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core machine learning concepts and principles

LO2 Explain the key mathematical principles of machine learning models

LO3 Review a range of classical machine learning algorithms

LO4 Deliver a workshop on the ethical, legal, and responsible practices of implementing machine learning models for an identified scenario.

## Essential Content

### LO1 **Explain the core machine learning concepts and principles**

*Machine learning paradigms:*

Learning paradigms: supervised learning e.g. labelled-pattern structures, prediction targets; unsupervised learning e.g. latent-pattern groups, distribution clusters, structural relationships; labelled data e.g. structured annotations, curated labels; unlabelled data e.g. raw observations, unstructured streams, domain records

Datasets: training datasets e.g. optimisation samples, representation corpora; validation datasets e.g. model-tuning subsets, comparative partitions, metric-checking sets; testing datasets e.g. robustness partitions, unseen evaluation sets.

*Model learning and generalisation:*

Learning processes: model training e.g. parameter trajectories, gradient refinements, optimisation steps

Model behaviour: overfitting e.g. variance-driven complexity, noise-linked patterns, overspecialised structures; underfitting e.g. low-capacity forms, simplified structures; bias–variance trade-off e.g. error-source balance, capacity tension, structural equilibrium

Generalisation concepts: generalisation error e.g. unseen-data deviation, population-level discrepancy; cross-validation e.g. rotation partitions, sampling loops, stability checks.

*Data properties and features:*

Feature properties: features e.g. signal descriptors, semantic indicators; feature spaces e.g. embedding manifolds, structured domains, high-dimensional sets; dimensionality e.g. feature counts, interaction scales

Data quality: noise e.g. stochastic variation, measurement distortion; missing data e.g. absent entries, structural gaps, incomplete fields; data quality e.g. validity indicators, consistency constraints

Data distribution issues: feature relevance e.g. signal strength, predictive indicators; class imbalance e.g. distribution skew, minority-class scarcity, majority dominance.

LO2 **Explain the key mathematical principles of machine learning models**

*Probability and statistics foundations:*

Probability concepts: probability distributions e.g. parameterised curves, density structures; random variables e.g. outcome domains, measurable quantities, stochastic variables; conditional probability e.g. dependency structures, joint-event relationships

Statistical properties: expectation e.g. central tendency measures, long-run averages; variance e.g. spread indicators, dispersion levels, variability measures; correlation e.g. dependency strength, association magnitude.

*Optimisation principles:*

Optimisation objectives: optimisation objectives e.g. performance targets, constraint conditions; cost functions e.g. error quantities, penalty structures, objective scores; loss functions e.g. discrepancy measures, prediction errors

Gradient-based methods: gradient-based optimisation e.g. gradient vectors, update directions, descent surfaces

Optimisation conditions: local minima e.g. suboptimal basins, narrow valleys; global minima e.g. optimum points, target solutions; learning rate sensitivity e.g. step-size scales, adjustment magnitudes.

*Loss functions and model evaluation mathematics:*

Error metrics: error metrics e.g. evaluation scores, deviation measures; regression loss functions e.g. squared-error terms, absolute-error terms, residual magnitudes; classification loss functions e.g. margin penalties, misclassification costs

Model reliability: complexity-error trade-offs e.g. structure penalties, performance balances; mathematical contribution to model robustness e.g. stability factors, reliability determinants.

LO3 **Review a range of classical machine learning algorithms**

*Regression approaches:*

Regression models: linear regression e.g. coefficient structures, residual patterns; logistic regression e.g. decision boundaries, class-probability surfaces, parameter weights

Output behaviour: continuous outputs e.g. prediction ranges, numerical targets; categorical outputs e.g. class mappings, threshold categories

Interpretability factors: model interpretability e.g. coefficient clarity, parameter meaning, structural transparency.

*Classification methods:*

Tree-based methods: decision trees e.g. split rules, node structures, leaf categories; boundary formation e.g. partition lines, region separations

Neighbourhood methods: nearest neighbour methods e.g. proximity metrics, neighbourhood sets, similarity measures; class separation e.g. cluster divisions, margin structures.

*Clustering techniques:*

Centroid-based methods: K-means clustering e.g. centroid assignments, distance groupings

Cluster structure: limitations e.g. shape constraints, noise sensitivity, scale distortions

Grouping behaviour: centroid-based grouping e.g. iterative refinements, partition cycles.

*Algorithm performance considerations:*

Performance measures: model accuracy e.g. error rates, prediction scores; precision and recall e.g. minority-class indicators, relevance measures

Computational properties: scalability e.g. dataset expansion effects, memory demands

Robustness factors: robustness to noise e.g. perturbation tolerance, stability margins; suitability to problem contexts e.g. task requirements, domain structures.

LO4 **Deliver a workshop on the ethical, legal, and responsible practices of implementing machine learning models for an identified scenario**

*Ethical considerations in machine learning:*

Ethical risks: bias e.g. representational gaps, group disparities; fairness e.g. equitable outcomes, protected-attribute considerations, allocation balance; discrimination e.g. exclusion risks, impact inequalities

Societal impact: societal impact e.g. community effects, social externalities; unintended consequences e.g. misuse risks, emergent issues, downstream harms; responsible innovation e.g. governance principles, design safeguards.

*Legal and regulatory considerations:*

Data protection: data protection e.g. privacy rights, personal-data controls; consent e.g. user permissions, access authorisations, opt-in requirements; accountability e.g. decision traceability, responsibility structures

Regulatory frameworks: compliance expectations e.g. legal standards, policy requirements; governance frameworks e.g. organisational controls, oversight structures, auditing mechanisms.

*Responsible AI principles:*

Governance and oversight: governance processes e.g. review committees, oversight roles; accountability structures e.g. responsibility records, decision logs, audit trails

Risk management: risk assessments e.g. harm indicators, impact scores; mitigation plans e.g. safeguard strategies, response frameworks; monitoring systems e.g. drift trackers, compliance checks

Operational standards: ethical guidelines e.g. fairness principles, transparency criteria; deployment policies e.g. usage boundaries, operational constraints, safety protocols.

*Workshop planning and delivery:*

Content structuring: session outlines e.g. concept sequences, topic stages; learning materials e.g. slide sets, demonstration assets, reference sheets

Engagement methods: engagement techniques e.g. interactive prompts, group activities; discussion elements e.g. case-study questions, scenario debates, reflection points

Delivery resources: delivery materials e.g. presentation tools, visual aids; facilitation resources e.g. workshop notes, activity instructions, timing guides.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the core machine learning concepts and principles</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of different machine learning concepts, algorithms and mathematical principles in developing reliable and effective machine learning models for different problem contexts.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the core machine learning concepts and principles, including model learning and generalisation.<br><br>P2 Discuss, using a selection of real-world examples, how data properties and features affect the performance of machine learning models.</td><td>M1 Analyse, using a selection of real-world examples, the core machine learning concepts and principles to show how learning paradigms and data characteristics affect model behaviour.</td></tr>
<tr><td colspan="2">LO2 Explain the key mathematical principles of machine learning models</td></tr>
<tr><td>P3 Explain, using a selection of real-world examples, the core probability and statistics concepts that underpin machine learning models.<br><br>P4 Explore, using a selection of real-world examples, the role of optimisation objectives and loss functions in machine learning model training.</td><td>M2 Evaluate, using a selection of real-world examples, how mathematical principles can impact the performance and behaviour of machine learning models.</td></tr>
<tr><td colspan="2">LO3 Review a range of classical machine learning algorithms</td></tr>
<tr><td>P5 Describe a selection of classical machine learning algorithms, including regression, classification, and clustering approaches.<br><br>P6 Discuss the performance factors that must be considered when selecting a machine learning algorithm.</td><td>M3 Assess the strengths and limitations of classical machine learning algorithms in relation to model performance and decision boundaries.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Deliver a workshop on the ethical, legal, and responsible practices of implementing machine learning models for an identified scenario | | |
| **P7** Plan the delivery of a workshop on the ethical, legal, and responsible practices of implementing machine learning models for an identified scenario to an identified audience.<br><br>**P8** Deliver a workshop on the ethical, legal, and responsible practices of implementing machine learning models for an identified scenario and obtain audience feedback. | **M4** Analyse the results of the audience feedback on the workshop delivery to identify a range of strengths and areas for improvement. | **D2** Critically evaluate the effectiveness of the workshop in communicating the ethical, legal, and responsible practices of implementing machine learning models to the intended audience. |

## Recommended Resources

### Textbooks

Aggarwal, C.C. (2023) *Neural Networks and Deep Learning: A Textbook*. 2nd edn. Cham: Springer.

Burkov, A. (2023) *Machine Learning Engineering*. Sebastopol, CA: O'Reilly Media.

James, G. *et al*. (2021) *An Introduction to Statistical Learning*. 2nd edn. Cham: Springer.

Jung, A. (2022) *Machine Learning: The Basics*. Cham: Springer.

Murphy, K. (2023) *Probabilistic Machine Learning: Advanced Topics*. Cambridge, MA: MIT Press.

Qamar, U. and Raza, M.S. (2023) *Data Science Concepts and Techniques with Applications*. Springer.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/machine-learning/machine-learning-with-python/ | GeeksforGeeks<br>Machine Learning with Python tutorial series covering core concepts, workflows and common libraries for implementing ML in Python.<br>(Tutorials) |
| www.geeksforgeeks.org/machine-learning/machine-learning-with-java/ | GeeksforGeeks<br>End-to-end guide to Machine Learning with Java, including setup, data handling and libraries for supervised/unsupervised learning.<br>(Tutorials) |
| realpython.com/tutorials/advanced/ | Real Python<br>Advanced Python tutorials (OOP, testing, performance, concurrency, internals) to deepen the algorithmic and implementation skills needed for ML work<br>(Tutorials) |

 © Pearson Education Limited 2026

| | |
|---|---|
| aicompetence.org/mastering-machine-learning-in-c-ultimate-guide/ | AI Competence<br>Theory-led overview of Machine Learning in C: motivations, linear algebra foundations, and implementing algorithms at low level for efficiency<br>(Guide) |
| www.geeksforgeeks.org/machine-learning/machine-learning/ | GeeksforGeeks<br>Broad Machine Learning theory hub: learning paradigms, pipelines, metrics and algorithm families with links into maths and evaluation.<br>(Reference) |
| www.ibm.com/think/machine-learning | IBM – Think<br>Conceptual Machine Learning guide: learning types, model training, feature engineering, evaluation and responsible-AI considerations.<br>(Learning Hub) |
| cs.cmu.edu/~avrim/Talks/mlt.pdf | Carnegie Mellon University<br>Machine Learning Theory overview – generalisation, complexity and formal models of learning<br>(PDF) |

## Journals and articles

Du, K.-L. *et al*. (2025) 'Understanding Machine Learning Principles: Learning, Inference, Generalization and Computational Learning Theory', *Mathematics*, 13(3), 451. Available at: doi.org/10.3390/math13030451.

Ezugwu, A.E. *et al*. (2024) 'Classical Machine Learning: Seventy Years of Algorithmic Learning Evolution', *Data Intelligence*. Available at: doi.org/10.3724/2096-7004.di.2024.0051.

Radanliev, P. *et al*. (2024) 'Ethics and responsible AI deployment', *Frontiers in Artificial Intelligence*, 7. Available at: doi.org/10.3389/frai.2024.1377011.

Yan, Y. (2023) 'Machine Learning Fundamentals', *Encyclopedia of Digital Agricultural Technologies*. Available at: doi.org/10.1007/978-3-031-24861-0_69.

## Links

This unit links to the following related units:

*Unit 520: Applied AI*

*Unit 521: Generative and Conversational AI.*

# Unit 523: Natural Language Processing

**Unit code:** D/652/3309

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Natural Language Processing (NLP) plays a central role in modern artificial intelligence, enabling digital systems to work directly with human language in both written and spoken form. As organisations continue to increase their use of data-driven technologies, NLP has become essential to fields such as software development, data analytics, cybersecurity, marketing, and customer-facing digital services. Its methods underpin everyday applications ranging from search engines and chatbots to document analysis and automated reporting, making NLP a key area of capability across the wider computing and AI sector.

The aim of this unit is to help students understand how machines process language and to build the skills needed to analyse and apply NLP methods in practical situations. Students will explore the ideas that underpin language processing, how linguistic structures can be modelled, and the techniques used to extract meaning from text. The unit also encourages students to think carefully about the strengths, limitations and appropriate use of NLP tools in real-world contexts.

The unit content follows a clear progression from core principles to applied practice. Students begin by examining the characteristics of natural language and the typical steps used to prepare and process text. They then investigate techniques for representing and analysing the structure of language, compare different approaches for understanding meaning, and apply NLP methods to generate insights for a chosen scenario. This structured approach supports both technical understanding and confident decision-making.

On successful completion of this unit, students will be able to explain how NLP works, compare the methods used to analyse text, and implement appropriate techniques to produce meaningful outputs. These capabilities support progression into a range of roles where language data plays a central part, including AI Engineer, Data Analyst, Machine Learning Specialist, Software Developer, Business Intelligence Analyst and Information Systems Technician.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the principles and concepts of Natural Language Processing (NLP)

LO2 Explore different NLP techniques that allow a machine to understand the structure of text

LO3 Discuss different NLP techniques that allow a machine to understand the meaning of text

LO4 Implement different NLP methods to generate insights from text for an identified scenario.

 © Pearson Education Limited 2026

## Essential Content

LO1 **Explain the principles and concepts of Natural Language Processing (NLP)**

*Foundations of Natural Language Processing:*

Natural language foundations: natural language data: e.g. text corpora, domain datasets; unstructured text, e.g. customer feedback, social media posts; linguistic ambiguity, e.g. word ambiguity, syntactic ambiguity, semantic ambiguity; human language characteristics, e.g. grammar rules, vocabulary variation

Approach types: rule-based approaches: e.g. grammar rules, pattern rules; statistical approaches: e.g. probability models, frequency distributions; machine learning approaches: e.g. classifiers, labelled datasets.

*Linguistic concepts in NLP:*

Linguistic levels: syntax: e.g. sentence structure, dependency relations; morphology: e.g. prefixes, suffixes; semantics: e.g. word meaning, semantic relations; pragmatics: e.g. situational context, conversational cues; discourse: e.g. topic structure, coherence

Lexical elements: lexical analysis: e.g. token patterns, word forms; sentence structure: e.g. clauses, phrases; word meaning: e.g. definitions, contextual variations; context dependence: e.g. discourse cues, co-text dependencies.

*Core NLP processing pipeline:*

Text processing stages: text acquisition: e.g. dataset sourcing, document collection; text normalisation: e.g. lowercasing, punctuation removal; tokenisation: e.g. word tokens, sentence tokens; sentence segmentation: e.g. boundary detection, segmentation models

Lexical processing: stemming: e.g. root forms, truncation rules; lemmatisation: e.g. canonical forms, dictionary lookup; stop-word handling: e.g. removal lists, filtering rules

Representation methods: feature representation: e.g. vectorisation, dense features, sparse features.

*Applications, benefits and limitations of NLP:*

Application areas: information retrieval, e.g. document ranking, search engines; text classification, e.g. topic grouping, spam detection; sentiment analysis, e.g. polarity scoring, opinion categories

Benefits: efficiency gains, e.g. automation, faster processing; improved insight, e.g. data patterns, topic trends; enhanced decision support, e.g. summaries, customer insight

Limitations: challenges of ambiguity, e.g. vague expressions, unclear context; bias in language data, e.g. skewed datasets, harmful associations; multilingual considerations, e.g. language variation, limited resources.

LO2 **Explore different NLP techniques that allow a machine to understand the structure of text**

*Text representation techniques:*

Text representation models: bag-of-words models: e.g. term counts, vocabulary frequencies; term frequency-inverse document frequency (TF-IDF): e.g. weighted terms, importance scores; n-grams: e.g. bigrams, trigrams

Vector representations: vector space models: e.g. document vectors, similarity spaces.

*Syntactic analysis techniques:*

Tagging methods: part-of-speech tagging: e.g. noun tags, verb tags; grammatical relationships: e.g. subject roles, object roles

Parsing techniques: constituency parsing: e.g. phrase structure trees, sentence branches; dependency parsing: e.g. dependency links, head relations; sentence tree structures: e.g. hierarchical forms, syntactic branches.

*Sequence modelling approaches:*

Probabilistic models: hidden Markov models: e.g. state transitions, emission probabilities; conditional random fields: e.g. sequence labelling, feature functions.

*Structural challenges in text:*

Structural variability: word order variation: e.g. flexible ordering, reordering patterns; sentence length: e.g. long inputs, complex structures

Text quality issues: grammatical inconsistency: e.g. incorrect forms, irregular patterns; domain-specific language: e.g. technical terms, specialised vocabulary; noisy text data: e.g. informal spelling, user-generated errors.

LO3 **Discuss the different NLP techniques that allow a machine to understand the meaning of text**

*Semantic representation methods:*

Vector representations: word embeddings: e.g. word vectors, semantic spaces; distributional semantics: e.g. co-occurrence patterns, semantic relations

Similarity measures: semantic similarity measures: e.g. cosine similarity, distance metrics.

*Semantic modelling techniques:*

Statistical semantics: latent semantic analysis: e.g. term matrices, concept mapping; topic modelling: e.g. topic distributions, word clusters.

*Meaning extraction approaches:*

Entity and role extraction: named entity recognition: e.g. entity labels, category tags; semantic role labelling: e.g. argument roles, predicate structures

Interpretation techniques: sentiment polarity detection: e.g. polarity scores, sentiment categories; intent recognition: e.g. intent labels, user goals; contextual meaning resolution: e.g. disambiguation cues, contextual signals.

*Comparative considerations in semantic analysis:*

Analytical factors: context sensitivity: e.g. meaning variation, word dependencies; scalability: e.g. dataset size, model capacity; computational complexity: e.g. memory use, processing time; training data requirements: e.g. labelled datasets, annotated corpora

Cross-domain factors: generalisation across domains: e.g. domain adaptation, cross-domain transfer; interpretability of semantic outputs: e.g. feature insights, model explanations; bias propagation: e.g. dataset imbalance, representation bias; domain transferability: e.g. adaptation effort, performance variation.

LO4 **Implement different NLP methods to generate insights from text for an identified scenario**

*Data preparation for NLP implementation:*

Data collection: text data collection: e.g. dataset sourcing, document gathering; data cleaning: e.g. noise removal, format correction

Preprocessing stages: preprocessing pipelines: e.g. token preparation, normalisation routines; feature extraction: e.g. vector creation, key term identification; dataset partitioning: e.g. training sets, testing sets.

*NLP model implementation:*

Model configuration: NLP techniques: e.g. classification models, sequence models; model configuration: e.g. architecture settings, feature mappings

Training processes: training processes: e.g. model training, weight updates; application to datasets: e.g. prediction generation, text analysis.

*Insight generation from text:*

Insight types: pattern identification: e.g. recurring terms, common themes; trend detection: e.g. upward trends, temporal changes; classification outputs: e.g. category labels, topic groups

Analytical outputs: sentiment trends: e.g. polarity movement, emotional patterns; thematic insights: e.g. theme clusters, topic patterns; summarisation outputs: e.g. key points, condensed summaries.

*Evaluation and interpretation of results:*

Evaluation metrics: model performance metrics: e.g. accuracy scores, precision values; output validation: e.g. result checking, data alignment; error analysis: e.g. misclassification review, anomaly detection

Interpretation factors: fitness for purpose: e.g. scenario alignment, requirement suitability; stakeholder relevance: e.g. audience needs, decision support; reliability of insights: e.g. consistency checks, stability analysis; ethical considerations: e.g. bias checks, fairness assessment.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the principles and concepts of Natural Language Processing (NLP)</td><td rowspan="2"></td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the core principles and concepts that underpin NLP.<br><br>P2 Discuss, using a selection of real-world examples, the benefits and limitations of NLP when applied across different business contexts.</td><td>M1 Analyse, using a selection of real-world examples, how key NLP principles and concepts enable machines to process and interpret human language.</td><td>D1 Critically evaluate, using real-world examples, the effectiveness of core NLP principles and concepts in supporting real-world language processing applications.</td></tr>
<tr><td colspan="2">LO2 Explore different NLP techniques that allow a machine to understand the structure of text</td><td rowspan="4">LO2 and LO3<br><br>D2 Critically evaluate, using real-world examples, the effectiveness of different NLP techniques for understanding the structure and meaning of text in a range of application contexts.</td></tr>
<tr><td>P3 Explore, using a selection of real-world examples, the NLP techniques used to analyse and represent the structural components of text.<br><br>P4 Investigate, using a selection of real-world examples, the structural challenges in processing text in NLP systems.</td><td>M2 Assess, using a selection of real-world examples, the strengths and limitations of NLP techniques used for understanding text structure.</td></tr>
<tr><td colspan="2">LO3 Discuss different NLP techniques that allow a machine to understand the meaning of text</td></tr>
<tr><td>P5 Discuss, using a selection of real-world examples, the NLP techniques used to understand the meaning of text.<br><br>P6 Investigate, using a selection of real-world examples, the comparative factors used in the semantic analysis of text.</td><td>M3 Analyse, using a selection of real-world examples, the differences in how NLP techniques derive meaning from textual data.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Implement different NLP methods to generate insights from text for an identified scenario | | |
| **P7** Implement NLP methods to generate insights from text for an identified scenario.<br><br>**P8** Review the insights generated by the NLP methods with a range of identified stakeholders to inform decisions. | **M4** Evaluate the quality of the insights generated by implemented NLP methods in relation to the identified scenario. | **D3** Justify how the chosen NLP methods contribute to the quality and value of the insights generated for the identified scenario. |

 © Pearson Education Limited 2026

## Recommended Resources

### Textbooks

Chaudhary, A. (2025) *Natural Language Processing with Transformers*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Eisenstein, J. (2023) *Introduction to Natural Language Processing*. Cambridge, MA: MIT Press.

Jurafsky, D. and Martin, J.H. (2024) *Speech and Language Processing*. 3rd edn. Hoboken: Pearson.

Lane, H., Howard, C. and Hapke, H. (2023) *Natural Language Processing in Action*. 2nd edn. Shelter Island: Manning.

Liu, B. (2024) *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. 2nd edn. Cambridge: Cambridge University Press.

Vajjala, S. *et al*. (2025) *Practical Natural Language Processing*. O'Reilly.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/nlp/natural-language-processing-nlp-101-from-beginner-to-expert/ | Geeks For Geeks<br>Natural Language Processing (NLP) 101: From Beginner to Expert<br>(Article Hub) |
| www.mindee.com/blog/natural-language-processing-basics | Mindee<br>The Basics of Natural Language Processing<br>(Blog) |
| huggingface.co | Hugging Face<br>NLP models, datasets, and implementation resources for applied language processing<br>(Development tool) |
| www.ibm.com/topics/natural-language-processing | IBM Technology<br>Overview of NLP concepts, applications, and enterprise use cases<br>(General reference) |

| | |
|---|---|
| www.mindwalkai.com/blog/from-words-to-meaning-exploring-semantic-analysis-in-nlp | MindWalk<br>From Words to Meaning: Exploring Semantic Analysis in NLP<br>(Article) |
| nlp.stanford.edu | Stanford NLP Group<br>NLP theory, tools, and language model research<br>(Research Hub) |
| scikit-learn.org | scikit-learn<br>Text processing and machine learning techniques for natural language processing<br>(Tutorials) |
| spacy.io | spaCy<br>Industrial-strength natural language processing library and documentation<br>(Development tool) |
| towardsdatascience.com/hugging-face-transformers-in-action-learning-how-to-leverage-ai-for-nlp/ | Towards Data Science<br>Hugging Face Transformers in Action: Learning How To Leverage AI for NLP<br>(Article) |
| spotintelligence.com/2023/10/28/syntactic-analysis-nlp/ | Spot Intelligence<br>Syntactic Analysis: A Power Tool In NLP Made Easy With Examples, Illustrations & Tutorials<br>(Resource) |

**Journals and articles**

Devlin, J. *et al*. (2019) 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding', *Proceedings of NAACL-HLT*, pp. 4171–4186. Available at: doi.org/10.18653/v1/N19-1423.

Radanliev, P. *et al*. (2024) 'Ethics and responsible AI deployment', *Frontiers in Artificial Intelligence*, 7. Available at: doi.org/10.3389/frai.2024.1377011.

Rogers, A., Kovaleva, O. and Rumshisky, A. (2020) 'A Primer in BERTology: What We Know About How BERT Works', *Transactions of the Association for Computational Linguistics*, 8, pp. 842–866. Available at: doi.org/10.1162/tacl_a_00349.

Stengel-Eskin, E. and Van Durme, B. (2023) 'Calibrated Interpretation: Confidence Estimation in Semantic Parsing', *Transactions of the Association for Computational Linguistics*, 11, pp. 1213–1231. Available at: doi.org/10.1162/tacl_a_00598.

**Links**

This unit links to the following related units:

*Unit 520: Applied AI*

*Unit 521: Generative and Conversational AI.*

# Unit 524: Advanced AI Concepts

**Unit code:** J/652/3310

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Artificial intelligence is now embedded in many aspects of modern digital systems, from everyday consumer applications to complex platforms used in business, engineering and scientific environments. As the capability of AI technologies continues to grow, organisations increasingly rely on more advanced methods that extend beyond basic automation and data processing. Understanding these higher-level concepts is essential for students who want to contribute confidently to the design, evaluation and responsible use of intelligent systems.

The aim of this unit is to give students a clear understanding of advanced AI concepts, focusing on how different approaches, structures and optimisation methods influence the behaviour and effectiveness of AI systems. Students will develop insight into how organisations select suitable AI models, how these systems are trained and refined, and how technical choices affect performance, scalability and reliability. The unit is designed to build on prior knowledge while supporting progression towards more specialised or technical pathways.

The unit covers several key areas that shape advanced AI practice. Students will explore a range of programming paradigms, including symbolic, data-driven and hybrid approaches, and examine how these models operate in different contexts. They will also study optimisation strategies used to scale AI systems effectively, alongside the training and tuning techniques that support robust model development. The unit concludes with a practical focus on communicating technical AI topics to a professional audience, ensuring students can present complex ideas clearly and with purpose.

On successful completion of the unit, students will be able to evaluate advanced AI approaches and explain how these methods are applied within real-world settings. They will also develop the confidence to communicate technical concepts in a professional environment, supporting progression into roles such as AI Technician, Junior AI Engineer, Machine Learning Assistant, Data Analyst, Software Developer or Technical Consultant working with intelligent systems. These skills are relevant to both employment and further academic study in computing and AI-related disciplines.

 © Pearson Education Limited 2026

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore a range of advanced AI programming paradigms

LO2 Discuss the key optimisation strategies for scalable AI systems

LO3 Review a range of advanced model training and tuning methodologies

LO4 Deliver a workshop to a technical audience about an advanced AI programming topic that shows professionalism.

## Essential Content

### LO1 **Explore a range of advanced AI programming paradigms**

*Advanced AI programming paradigms:*

Symbolic approaches: symbolic AI e.g. logic rules, knowledge bases; rule-based systems e.g. expert rules, decision tables

Data-driven approaches: neural network models e.g. convolutional networks, recurrent networks, multi-layer perceptrons; deep learning approaches e.g. deep neural networks, representation learning

Hybrid approaches: hybrid AI models e.g. neuro-symbolic systems, ensemble methods; data-driven approaches e.g. statistical modelling, pattern extraction, model-based inference.

*Paradigm characteristics and purposes:*

System properties: knowledge representation e.g. semantic structures, feature encoding; learning approaches e.g. supervised learning, unsupervised learning, reinforcement learning

Operational methods: inference methods e.g. rule-chaining, probabilistic inference; adaptability e.g. model updating, task generalisation, transferability

Application qualities: explainability e.g. rule transparency, feature importance, interpretable outputs; application context e.g. domain suitability, system requirements.

*Paradigm comparison and evaluation factors:*

Performance factors: flexibility e.g. task adaptability, model variation; scalability e.g. system expansion, increasing complexity, resource growth; performance considerations e.g. accuracy needs, latency constraints

Interpretability factors: interpretability e.g. clarity of outputs, transparency of reasoning; suitability for problem types e.g. structured tasks, unstructured data, real-time processing

Selection factors: evaluation criteria e.g. model fit, system constraints; paradigm suitability e.g. domain needs, operational fit, complexity alignment.

*Model Context Protocol (MCP) Standard:*

Structural elements: structure e.g. workflow stages, component organisation; prompts e.g. task instructions, contextual cues, system directives; tools e.g. data interfaces, evaluation utilities

Operational elements: agent roles e.g. reasoning agent, data agent; safety rules e.g. compliance checks, usage boundaries, risk controls; workflows e.g. task pipelines, operational sequence

Symbolic approaches e.g. Cyc, Wolfram Mathematica rule-based engine, Siemens BRIX expert-system framework

Data–driven approaches e.g. Google Photos CNN, OpenAI Whisper, Meta DeepFace

Paradigm properties e.g. Google Knowledge Graph, Wikidata, DeepMind AlphaZero

Performance factors e.g. NVIDIA TensorRT, Amazon SageMaker distributed training, TikTok Recommendation Engine

Model standards e.g. LangChain, Microsoft PromptFlow, OpenAI Assistants API.

LO2 **Discuss the key optimisation strategies for scalable AI systems**

*AI system scalability concepts:*

Scalability principles: scalability e.g. system expansion, capacity growth; horizontal scaling e.g. node replication, workload distribution; vertical scaling e.g. resource upgrades, hardware enhancement

Distributed deployment: distributed deployment e.g. cluster deployment, multi-region deployment, distributed services

System performance: performance e.g. execution speed, stability; throughput e.g. task volume, data flow; latency e.g. response duration, request delay

Resource management: resource management e.g. memory planning, hardware usage; load balancing e.g. traffic routing, demand distribution, workload sharing.

*Optimisation strategies:*

Model optimisation: model optimisation e.g. pruning, quantisation; algorithmic improvements e.g. efficiency refinement, complexity reduction, optimisation routines

Data optimisation: data optimisation e.g. data filtering, dataset restructuring

Specialised hardware: specialised hardware e.g. GPUs, TPUs, AI accelerators

Parallel processing: parallel processing e.g. task parallelism, distributed execution, multithreading.

*Performance, efficiency, and effectiveness considerations:*

Resource use: resource use e.g. memory consumption, processing load;

Reliability: reliability e.g. operational stability, fault tolerance, consistent outputs

Sustainability: considerations e.g. energy efficiency, carbon impact, resource optimisation; environmental impact e.g. power consumption, data centre efficiency, hardware life cycle; efficiency strategies e.g. model compression, low-power inference, workload optimisation

Cost considerations: cost considerations e.g. infrastructure cost, resource budgeting

Constraint balancing: performance constraints e.g. latency limits, hardware limits, processing limits; system constraints e.g. infrastructure constraints, deployment constraints

*Examples of real-world AI system applications:*

A range of AI system examples e.g.

Deployment systems e.g. Google Kubernetes Engine, Amazon EKS, Microsoft Azure AKS

Model optimisations e.g. TensorRT model optimisation, ONNX Runtime optimised inference, Meta Llama-2 quantised model

Resource usage e.g. NVIDIA GPU memory-optimised scheduling, AWS Inferentia low-resource inference, Google TPU v5e resource-efficient compute

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Review a range of advanced model training and tuning methodologies**

*Model training methodologies:*

Supervised training workflows: supervised workflows e.g. training datasets, labelled outputs; evaluation sets e.g. validation data, performance checks, accuracy metrics

Unsupervised approaches: unsupervised approaches e.g. clustering patterns, feature grouping; representation learning e.g. structure extraction, pattern formation, embedding spaces

Transfer learning: transfer learning e.g. pretrained models, feature reuse; domain adaptation e.g. task alignment, dataset adjustment, context adaptation.

*Model tuning techniques:*

Hyperparameter adjustment: hyperparameter adjustment e.g. learning rate, batch size; tuning schedules e.g. optimisation routines, step changes, adjustment cycles

Regularisation methods: regularisation methods e.g. dropout, layer weight decay; constraint techniques e.g. capacity limits, smoothing methods, model penalties

Validation strategies: validation strategies e.g. cross-validation, holdout sets; performance checks e.g. accuracy measures, error rates, evaluation criteria.

*Training, tuning, and evaluation outcomes:*

Model accuracy: accuracy e.g. correct predictions, score measures; performance metrics e.g. precision measures, recall measures, evaluation ratios

Reliability: reliability e.g. robustness indicators, stability checks; consistency e.g. reproducible outputs, dependable behaviour, predictable patterns

Limitations of training approaches: limitations e.g. data bias, overfitting; constraints e.g. resource limitations, dataset restrictions, generalisation limits

Criteria for determining model effectiveness: effectiveness criteria e.g. task suitability, domain alignment; model comparison e.g. benchmark scores, evaluation ranking, performance balance

*Examples of real-world AI system applications:*

A range of AI system examples e.g.

Training workflows e.g. ImageNet training pipelines, OpenAI GPT-finetuning workflows, Google Speech Commands supervised models

Hyperparameter adjustment e.g. Ray Tune hyperparameter search, Optuna optimisation runs, Google Vizier hyperparameter tuning

Accuracy checking e.g. Bidirectional Encoder Representations from Transformers (BERT) benchmark accuracy, ResNet ImageNet accuracy, YOLO detection precision

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Deliver a workshop to a technical audience about an advanced AI programming topic that shows professionalism**

*Workshop purpose and structure:*

Technical audience requirements: audience requirements e.g. prior knowledge, technical expectations; audience needs e.g. learning aims, topic relevance, skill alignment

Scope definition: scope definition e.g. session boundaries, core topics; content depth e.g. essential details, technical clarity, appropriate complexity

Logical sequencing of content: content sequencing e.g. structured flow, progressive steps; session outlines e.g. segment order, time allocation, topic grouping.

*Technical communication:*

Technical language: technical language e.g. accurate terminology, domain-specific vocabulary; clarity of explanation e.g. concise descriptions, precise wording, clear definitions

Audience engagement: audience engagement e.g. interactive prompts, question opportunities; delivery structure e.g. logical transitions, consistent formatting, clear progression

Learning materials: learning materials e.g. slide resources, reference notes; delivery resources e.g. demonstration tools, visual aids, supporting documents.

*Professional delivery standards:*

Presentation organisation: presentation organisation e.g. structured layout, ordered sections; coherence e.g. consistent narrative, logical emphasis, aligned messaging

Technical tone: technical tone e.g. appropriate register, professional language; audience-appropriate language e.g. accessible wording, level-matched terminology, clear phrasing

Time management: time management e.g. session pacing, allocated segments; structured responses e.g. orderly answers, relevant clarification, focused explanations.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explore a range of advanced AI programming paradigms | | **LO1, LO2 and LO3**<br>**D1** Evaluate how advanced AI programming paradigms, optimisation strategies, and model training/tuning methods work together to solve complex AI problems, using real-world examples to support your judgement. |
| **P1** Explore, using a selection of real-world examples, a range of advanced AI programming paradigms and the Model Context Protocol standard.<br>**P2** Review, using a selection of real-world examples, the characteristics of the identified paradigms and the factors that are used to compare and evaluate them. | **M1** Compare, using a selection of real-world examples, the effectiveness of the chosen programming paradigms in a range of application contexts. | |
| **LO2** Discuss the key optimisation strategies for scalable AI systems | | |
| **P3** Discuss, using a selection of real-world examples, the key optimisation strategies used to support different AI systems.<br>**P4** Review, using a selection of real-world examples, the concepts of scalability, performance, efficiency and effectiveness in a range of application contexts. | **M2** Analyse, using a selection of real-world examples, how optimisation strategies contribute to the scalability and performance of AI systems. | |
| **LO3** Review a range of advanced model training and tuning methodologies | | |
| **P5** Review, using a selection of real-world examples, a range of advanced model training and tuning methodologies used in AI development.<br>**P6** Discuss, using a selection of real-world examples, the training, tuning and evaluation outcomes that are used in the development of AI systems. | **M3** Assess, using a selection of real-world examples, the impact of training and tuning methodologies on AI model performance and reliability. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Deliver a workshop to a technical audience about an advanced AI programming topic that shows professionalism | | |
| **P7** Plan the delivery of a workshop on an advanced AI programming topic to an identified audience.<br><br>**P8** Deliver a workshop on an advanced AI programming topic to a technical audience that shows professionalism and obtain audience feedback. | **M4** Analyse the results of the audience feedback on the professionalism of the workshop delivery to identify a range of strengths and areas for improvement. | **D2** Evaluate the overall effectiveness of the workshop in communicating an advanced AI programming topic to an intended technical audience. |

## Recommended Resources

### Textbooks


Aggarwal, C.C. (2023) *Neural Networks and Deep Learning: A Textbook*. 2nd edn. Cham: Springer.

Alammar, J. and Grootendorst, M. (2024) *Hands-On Large Language Models: Language Understanding and Generation*. Sebastopol, CA: O'Reilly Media.

Das, D., Sadiq, A.S. and Mirjalili, S. (2025) *Optimization Algorithms in Machine Learning: A Meta-Heuristics Perspective*. Cham: Springer.

Foster, D. (2023) *Generative Deep Learning*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*. Cambridge, MA: MIT Press.

Murphy, K.P. (2023) *Probabilistic Machine Learning: Advanced Topics*. Cambridge, MA: MIT Press.

Russell, S. and Norvig, P. (2021) *Artificial Intelligence: A Modern Approach*. 4th edn. Harlow: Pearson.


### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| algocademy.com/uses/modern-programming-paradigms-with-ai/ | AlgoCademy,<br>AI-supported programming paradigms and step-by-step demonstrations.<br>(Training) |
| cdn.aaai.org/ojs/19007/19007-13-22824-1-10-20211006.pdf | AAAI<br>Case studies on AI programming paradigms translated into Python for practical learning.<br>(Research) |
| docs.cloud.google.com/architecture/framework/perspectives/ai-ml/performance-optimization | Google Cloud<br>Well-Architected Framework, AI/ML performance optimisation principles.<br>(Report) |
| www.ibm.com/think/topics/optimize-ai-workloads | IBM Think<br>Infrastructure optimisation strategies for AI workloads.<br>(Article) |

| | |
|---|---|
| www.geeksforgeeks.org/machine-learning/hyperparameter-tuning/ | GeeksforGeeks<br>Techniques for hyperparameter search and optimisation.<br>(Reference) |
| www.technical-communication.org/ai-in-technical-communication | Tekom<br>AI in professional technical communication practices.<br>(Blog) |
| www.wevolver.com/article/ai-infrastructure | Wevolver<br>Enterprise-scale AI infrastructure architectures and optimisation patterns. (Reference) |

**Journals and articles**

Alenezi, M. and Akour, M. (2025) 'AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions', *Applied Sciences*, 15(3), 1344. Available at: doi.org/10.3390/app15031344.

Radanliev, P. (2024) 'Artificial intelligence: reflecting on the past and looking towards the next paradigm shift', *Journal of Experimental & Theoretical Artificial Intelligence*. Available at: doi.org/10.1080/0952813X.2024.2323042.

Shah, R. and Gillen, A.L. (2024). 'A systematic literature review of university-industry partnerships in engineering education', *European Journal of Engineering Education*, 49(3), pp. 577–603. Available at: doi.org/10.1080/03043797.2023.2253741.

Wang, L. *et al*. (2025) 'Parameter-efficient fine-tuning in large language models: A survey of methodologies', *Artificial Intelligence Review*, 58, Article 227. Available at: doi.org/10.1007/s10462-025-11236-4.

**Links**

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science (Pearson-set)*

*Unit 420: Core AI*

*Unit 421:  AI Automation and Digital Productivity*

*Unit 423: Agentic AI*

*Unit 520: Applied AI.*

# Unit 525: Computer Vision

**Unit code:** K/652/3311

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Computer vision is a key area of modern computing that focuses on enabling machines to interpret and make sense of visual information such as images and video. It underpins many everyday and industrial technologies, from facial recognition and automated inspection to medical imaging and intelligent transport systems. As organisations increasingly rely on visual data to support decision-making and automation, there is a growing need for computing professionals who understand how visual information is captured, processed, and analysed within digital systems.

The aim of this unit is to develop students' understanding of the principles and techniques that underpin computer–vision, while building the practical skills required to apply these techniques in real-world computing contexts. The unit is designed to strengthen students' analytical and technical capabilities by linking theoretical concepts with hands-on implementation, enabling them to make informed decisions when selecting and evaluating computer vision approaches.

This unit covers the core foundations of computer vision, including image formation, pixel representation, colour models, filtering operations, and feature extraction. Students will then explore classical computer–vision techniques used for image processing and recognition, before examining how machine learning and deep learning approaches are applied in modern vision systems. The unit concludes with the design and implementation of a simple computer–vision prototype, allowing students to follow a structured development process from problem definition through to evaluation.

On successful completion of this unit, students will be able to analyse visual data, evaluate different computer–vision techniques, and contribute to the development of image-based computing solutions. These skills support progression into roles such as Computer Vision Developer, Software Developer, Data Analyst, Machine Learning Engineer, AI Technician, or Systems Developer, as well as further study in areas related to artificial intelligence, data science, and advanced computing technologies.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core principles of computer vision

LO2 Explore classical computer–vision techniques for image processing and recognition

LO3 Review the machine learning approaches used in modern computer vision

LO4 Implement a simple prototype computer–vision solution for an identified scenario.

## Essential Content

### LO1 **Explain the core principles of computer vision**

*Image formation and representation:*

Image structure: pixel grids e.g. pixel layout, pixel density; resolutions e.g. image resolution, sensor resolution, output resolution; quantisation levels e.g. bit depth, intensity levels; examples: smartphone image sensors, DSLR pixel-grid sensors, CCTV camera resolution systems

Image types: grayscale images e.g. luminance values, grayscale channels; colour images e.g. red-green-blue (RGB) images, hue, saturation and value (HSV) images, multichannel images

Image components: pixels e.g. pixel values, pixel positions, pixel intensity; channels e.g. red channel, green channel, blue channel.

*Colour models and transformations:*

Colour spaces: RGB e.g. red values, green values, blue values; HSV e.g. hue settings, saturation levels, value levels; YUV e.g. luminance values, chrominance-U values, chrominance-V values

Transformations: colour conversions e.g. RGB-to-HSV, RGB-to-grayscale; grayscale transformations e.g. luminance extraction, weighted grayscale methods; colour mappings e.g. histogram mapping, palette mapping

Image properties: colour distributions e.g. colour histograms, pixel colour spreads, palette ranges; brightness levels e.g. luminance values, light intensity levels, exposure levels; chrominance values e.g. U-channel values, V-channel values, colour-difference values.

*Filtering and feature extraction principles:*

Filtering operations: smoothing filters e.g. mean filter, median filter; sharpening filters e.g. unsharp mask, Laplacian filter, high-boost filter; convolution kernels e.g. edge kernel, blur kernel

Feature responses: image gradients e.g. horizontal gradients, vertical gradients, magnitude gradients; intensity changes e.g. brightness differences, contrast variations; edges e.g. detected edges, boundary lines, edge transitions

Noise handling: noise patterns e.g. sensor noise, grain noise; noise reduction methods e.g. smoothing techniques, denoising filters; artefact removal e.g. blur correction, distortion removal.

*Examples of real-world computer-vision applications:*

A range of real-world computer vision tools e.g.

Colour and image tools: OpenCV e.g. colour conversion, image processing; YouTube video encoding e.g. YUV colour processing, video transformation; Adobe Photoshop e.g. colour adjustment, image editing

Filtering and feature detection tools: OpenCV blur tools e.g. noise reduction, smoothing; OpenCV Laplacian e.g. edge enhancement, sharpening; Sobel operator e.g. edge detection, gradient analysis

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Explore classical computer–vision techniques for image processing and recognition**

*Edge detection and segmentation techniques:*

Edge features: gradients e.g. gradient magnitudes, gradient directions, edge responses; thresholds e.g. threshold levels, intensity thresholds; boundaries e.g. object boundaries, region boundaries

Segmentation methods: regions e.g. region groups, region areas, region masks; contours e.g. contour shapes, contour outlines; masks e.g. segmentation masks, binary masks

Contrast and transitions: contrast e.g. local contrast, global contrast; transitions e.g. intensity transitions, boundary transitions.

*Keypoint detection and feature matching:*

Keypoint types: corners e.g. corner features, corner points; blobs e.g. blob regions, blob shapes, blob points; keypoints e.g. keypoint positions, keypoint features, keypoint strengths

Feature comparison: matches e.g. feature matches, match pairs, match sets; distances e.g. Euclidean distances, descriptor distances; similarities e.g. similarity scores, feature similarities

Local neighbourhoods: neighbourhoods e.g. local patches, local windows; scores e.g. response scores, detection scores.

*Pattern recognition methods:*

Pattern structures: patterns e.g. pattern templates, pattern features, structured patterns; shapes e.g. simple shapes, geometric shapes; classes e.g. object classes, shape classes

Feature groupings: categories e.g. object categories, feature categories; groupings e.g. grouped features, grouped patterns; features e.g. visual features, extracted features, spatial features

Similarity measures: similarities e.g. similarity scores, similarity metrics; metrics e.g. distance metrics, comparison metrics; distances e.g. feature distances, shape distances

*Examples of real-world computer-vision applications:*

A range of real-world computer vision tools e.g.

Edge detection tools: Canny edge detector e.g. edge detection, boundary extraction; Sobel operator e.g. gradient detection, edge analysis; OpenCV Laplacian e.g. edge enhancement, image sharpening

Feature detection tools: Harris corner detector e.g. corner detection, feature points; scale-invariant feature transform (SIFT) e.g. keypoint detection, feature matching;
speeded-up robust features (SURF) e.g. feature extraction, image matching

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Review the machine learning approaches used in modern computer vision**

*Machine learning foundations in computer vision:*

Learning types: supervised learning e.g. labelled data, training labels, validation labels; training data e.g. input samples, dataset images; validation data e.g. validation sets, validation samples, validation labels

Dataset structure: labelled datasets e.g. image classes, annotation tags; data splits e.g. train sets, validation sets, test sets; distributions e.g. class distributions, sampling distributions

Generalisation factors: annotations e.g. bounding boxes, class labels; overfitting e.g. training errors, memorised patterns; generalisation e.g. unseen samples, test performance, prediction stability.

*Deep learning architectures for vision:*

Core components: convolutional networks e.g. convolutional stages, learned filters, extracted features; layers e.g. convolutional layers, pooling layers, activation layers; filters e.g. edge filters, texture filters, shape filters

Representations: feature maps e.g. activation maps, spatial feature maps; activations e.g. rectified linear units (ReLU) activations, output activations; representations e.g. learned features, hierarchical features

Model development: pretrained models e.g. pretrained backbones, pretrained weights; transfer learning e.g. feature reuse, fine-tuning methods, adapted models; architectures e.g. convolutional neural network (CNN) architectures, lightweight networks, deep networks.

*Model evaluation and performance measures:*

Dataset splits: training sets e.g. training images, training batches; test sets e.g. evaluation images, unseen samples; comparisons e.g. model comparisons, metric comparisons, score comparisons

Performance metrics: metrics e.g. accuracy metrics, classification metrics; accuracy e.g. correct predictions, accuracy scores; precision e.g. positive precision, class precision, prediction precision

Evaluation outputs: recall e.g. recall scores, recall rates; results e.g. evaluation results, output scores, summary metrics; performance scores e.g. model scores, performance results, scoring outputs.

*Examples of real-world computer-vision applications:*

A range of real-world computer vision tools e.g.

Deep learning models: ResNet e.g. image classification, feature extraction; Visual Geometry Group (VGG) e.g. image recognition, convolution layers; MobileNet e.g. lightweight models, mobile vision

Model evaluation tools: ImageNet benchmarks e.g. accuracy testing, model comparison; Common Objects in Context (COCO) mean average precision (mAP) e.g. object detection, precision scoring; YOLOv8 metrics e.g. detection accuracy, performance evaluation

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Implement a simple prototype computer–vision solution for an identified scenario**

*Computer–vision pipeline design:*

Problem framing: problem definition e.g. task objectives, system goals, use cases; input data e.g. image data, video data, dataset inputs; output requirements e.g. classification results, detection outputs, processed images

Processing pipeline: preprocessing stages e.g. resizing, normalisation, filtering; feature extraction e.g. key features, visual patterns, descriptors; processing steps e.g. sequential operations, pipeline stages, transformation steps

System configuration: algorithm selection e.g. detection algorithms, classification models, processing methods; configuration settings e.g. parameters, thresholds, tuning values; data flow e.g. data movement, pipeline flow, processing paths

Design documentation: design requirements e.g. input requirements, user needs; design diagrams e.g. flow diagrams, block diagrams, pipeline diagrams; system specifications e.g. hardware specifications, software specifications, configuration specifications; workflow descriptions e.g. step descriptions, process summaries, task outlines.

*Solution development and integration:*

Prototype building: prototype implementation e.g. pipeline build, workflow setup; algorithms e.g. filtering algorithms, detection algorithms, segmentation algorithms; components e.g. processing modules, input modules, output modules

System integration: system setup e.g. file handling, input setup; data handling e.g. loading data, storing outputs, managing inputs; integration points e.g. module connections, data transfers, step linking

Outputs: outputs e.g. processed images, detection outputs, segmentation results.

*Optimisation and performance considerations:*

Performance management: efficiency e.g. processing speed, runtime reduction; accuracy e.g. detection accuracy, classification accuracy; resources e.g. memory use, compute use

Evaluation: performance measures e.g. performance scores, evaluation scores, result summaries; evaluation results e.g. output quality, measured accuracy, performance differences; limitations e.g. data limits, processing limits, method limits

Improvements: improvements e.g. parameter adjustments, preprocessing adjustments, method adjustments.

*Examples of real-world computer-vision applications:*

A range of real-world computer vision tools e.g.

Computer–vision optimisation tools: TensorRT e.g. model acceleration, optimised inference; ONNX Runtime e.g. performance optimisation, cross-platform inference; OpenCV optimisation e.g. CPU/GPU processing, runtime efficiency

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explain the core principles of computer vision | | **LO1, LO2 and LO3**<br>**D1** Critically evaluate how machine learning approaches affect the application and effectiveness of computer vision in image processing and recognition. |
| **P1** Explain, using a selection of real-world examples, the core principles of computer vision in relation to image formation, pixel representation and colour models.<br>**P2** Describe, using a selection of real-world examples, the principles of image filtering and feature extraction. | **M1** Examine, using a selection of real-world examples, how core computer-vision principles allow effective image representation and feature extraction in computer-vision systems. | |
| **LO2** Explore classical computer–vision techniques for image processing and recognition | | |
| **P3** Explore, using a selection of real-world examples, classical computer-vision techniques used for image processing including edge detection, segmentation and keypoint detection.<br>**P4** Discuss, using a selection of real-world examples, the methods and techniques of pattern recognition within images. | **M2** Analyse, using a selection of real-world examples, the effectiveness of different classical computer-vision techniques for image processing and pattern recognition in defined contexts. | |
| **LO3** Review the machine learning approaches used in modern computer vision | | |
| **P5** Review, using a selection of real-world examples, the application of machine learning approaches and deep learning architectures used in modern computer-vision systems and applications.<br>**P6** Explore, using a selection of real-world examples, the measures used to evaluate model performance in modern computer-vision systems and applications. | **M3** Assess, using a selection of real-world examples, how machine learning approaches can impact performance in modern computer-vision systems. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Implement a simple prototype computer-vision solution for an identified scenario | | |
| **P7** Design a simple prototype computer-vision solution for an identified scenario using an appropriate computer vision pipeline.<br><br>**P8** Implement a simple prototype computer–vision solution for an identified scenario using an appropriate computer-vision pipeline.<br><br>**P9** Review the implemented prototype computer-vision solution for an identified scenario with a range of key stakeholders. | **M4** Optimise, using the results of the stakeholder review, the implemented computer-vision solution to meet the requirements of the identified scenario more effectively. | **D2** Critically evaluate the performance and limitations of the optimised computer-vision solution, identifying areas for future development. |

## Recommended Resources

### Textbooks

Distante, A. and Distante, C. (2021) *Handbook of Image Processing and Computer Vision*. Cham: Springer.

Lakshmanan, V., Görner, M. and Gillard, R. (2021) *Practical Machine Learning for Computer Vision*. Sebastopol, CA: O'Reilly Media.

Prince, S.J.D. (2024) *Computer Vision: Models, Learning, and Inference*. 2nd edn. Cambridge: Cambridge University Press.

Szeliski, R. (2022) *Computer Vision: Algorithms and Applications*. 2nd edn. Cham: Springer.

Zhao, X. *et al*. (2024) *Deep Learning for Computer Vision*. Cham: Springer.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/computer-vision/computer-vision/ | GeeksforGeeks<br>Computer Vision Tutorial<br>(General reference) |
| link.springer.com/journal/11263 | International Journal of Computer Vision<br>List of searchable research papers on computer vision.<br>(Research Hub) |
| visionbook.mit.edu | MIT Press – Foundations of Computer Vision<br>Free online textbook on computer vision<br>(General reference) |
| www.toptal.com/developers/computer-vision/computer-vision-pipeline | Toptal<br>Computer vision pipeline architecture and implementation<br>(Tutorial) |
| www.geeksforgeeks.org/computer-vision/comprehensive-guide-to-edge-detection-algorithms/ | GeeksforGeeks<br>Edge detection algorithms in computer vision<br>(Reference) |

| | |
|---|---|
| towardsdatascience.com/image-segmentation-with-classical-computer-vision-based-approaches-80c75d6d995f/ | Towards Data Science<br>Classical Image Segmentation Techniques<br>(Article) |
| www.geeksforgeeks.org/computer-vision/evaluation-of-computer-vision-model/ | GeeksforGeeks<br>Evaluation metrics for computer vision models<br>(Article) |

**Journals and articles**

Bhuvaneshwaran, D.S. *et al*. (2025) 'Edge detection in image segmentation', *AIP Conference Proceedings*, 3175(1). Available at: doi.org/10.1063/5.0259758.

Jing, J. *et al*. (2023) 'Recent advances on image edge detection: A comprehensive review', *Information Fusion*, 90, pp. 1–21. Available at: doi.org/10.1016/j.neucom.2022.06.083.

Nissana, A. *et al*. (2025) 'Efficient computer vision pipeline for automated anesthetic injection documentation', *Computer Assisted Surgery*, 30(1). Available at: doi.org/10.1080/24699322.2025.2582020.

Zhao, X. *et al*. (2024) 'A review of convolutional neural networks in computer vision', *Artificial Intelligence Review*, 57(99). Available at: doi.org/10.1007/s10462-024-10721-6.

**Links**

This unit links to the following related units:

*Unit 520: Applied AI*

*Unit 521: Generative and Conversational AI.*

 © Pearson Education Limited 2026

# Unit 526: Deep Learning

**Unit code:** L/652/3312

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Deep learning has become a central part of modern computing, driving major advances across sectors that rely on large-scale data and pattern recognition. From automated customer service systems to image analysis, fraud detection and autonomous technologies, deep learning models now support a wide range of real-world applications. As demand grows for professionals who understand how these models work and how they can be applied responsibly, deep learning has become a key area of study within artificial intelligence and data-driven computing.

The aim of this unit is to give students a clear foundation in the theoretical principles that underpin deep learning. The unit introduces the structure and behaviour of neural networks, the role of activation functions, and the mechanisms that support model learning, including gradient-based optimisation. The unit emphasises conceptual understanding rather than complex coding, ensuring that students can interpret how models operate and explain the decisions behind architectural choices.

Students will explore a range of widely used deep learning architectures, including convolutional networks, recurrent networks and transformer-based models, and will examine how these approaches are suited to different types of data and problem domains. The unit also guides students through the process of adapting pretrained models, focusing on the workflow stages involved in data preparation, model configuration, training and evaluation.

On successful completion of this unit, students will be able to explain fundamental deep learning concepts, analyse different model architectures, develop basic training workflows and evaluate model performance using appropriate measures. These skills support progression into technical and applied roles such as AI Engineer, Machine Learning Engineer, Data Scientist, Computer Vision Technician, Applied AI Specialist or Software Developer working with intelligent systems.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the theoretical principles of deep learning

LO2 Explore the different deep learning architectures and their application

LO3 Develop a simple training workflow that adapts a pretrained deep learning model to solve a defined problem in a specific scenario

LO4 Evaluate the performance and limitations of the training workflow developed for the identified scenario.

## Essential Content

LO1 **Explain the theoretical principles of deep learning**

*Foundations of deep learning:*

Learning concepts: artificial neural networks e.g. convolutional networks, transformer models; feature learning e.g. pattern extraction, representation learning, abstraction processes

Generalisation principles: supervised learning principles e.g. labelled data, target outputs; model generalisation e.g. unseen data behaviour, validation performance, distribution adaptation

Model behaviour: overfitting e.g. noise memorisation, overspecialisation, poor generalisation; underfitting e.g. high bias, insufficient learning.

*Perceptrons and neural network structures:*

Perceptron model: perceptron model e.g. threshold logic, binary decision boundaries; weighted inputs e.g. feature weights, weighted summation

Network structures: multi-layer perceptrons e.g. nonlinear mapping, multilayer structure, representation learning; hidden layers e.g. intermediate abstractions, feature transformations; feedforward networks e.g. directed acyclic flow, input-to-output propagation.

*Activation functions:*

Function purpose: activation function purpose e.g. signal transformation, modelling flexibility; non-linearity e.g. nonlinear mapping, complex boundary formation, expressive capacity

Function types: rectified linear units (ReLU) e.g. sparse activations, fast computation; sigmoid e.g. probability mapping, smooth transitions; tanh e.g. centred output, stronger gradients.

*Optimisation and learning mechanisms:*

Learning mechanics: loss functions e.g. error measurement, objective calculation; gradient descent principles e.g. optimisation direction, iterative improvement; weight updates e.g. parameter adjustment, refinement

Training dynamics: learning rate e.g. convergence tuning, stability control, optimisation speed; training stability e.g. gradient behaviour, consistency control; common training challenges e.g. vanishing gradients, exploding gradients

*Examples of real-world AI tools:*

A range of real-world AI tools e.g.

Deep learning systems: OpenAI GPT models e.g. text generation, language modelling; Google Vision AI e.g. image analysis, feature extraction; Meta convolutional models e.g. image recognition, feature learning

Neural network tools: TensorFlow MLPClassifier e.g. model training, classification; PyTorch networks e.g. model development, forward propagation; Keras perceptron models e.g. neural network building, training workflows

Optimisation tools: PyTorch Adam optimiser e.g. parameter updates, optimisation; TensorFlow SGD e.g. gradient descent, model training; Keras loss functions e.g. error calculation, model evaluation

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Explore the different deep learning architectures and their applications**

*Convolutional neural network architectures:*

Core components: convolutional layers e.g. feature extraction, spatial filtering; filters and kernels e.g. edge detection, texture capture, pattern extraction; pooling operations e.g. max pooling, average pooling, spatial reduction

Application focus: feature maps e.g. object representation, activation patterns; spatial hierarchy e.g. low-level features, high-level features, hierarchical understanding.

*Recurrent neural network architectures:*

Temporal modelling: sequential data e.g. sentence sequences, sensor readings, time-ordered inputs; temporal patterns e.g. rhythm detection, trends, time-dependencies

Recurrent units: hidden state e.g. memory retention, context propagation; gated units (long short-term memory (LSTM) basics) e.g. input gate control, forget gate behaviour, output gate filtering.

*Transformer-based architectures:*

Attention mechanism: attention mechanism e.g. relevance weighting, context scoring; self-attention basics e.g. token interactions, contextual relationships, parallel attention

Model structure: encoder roles e.g. feature encoding, context building; decoder roles e.g. output generation, sequence prediction.

*Architecture selection and application context:*

Task alignment: data characteristics e.g. image data, text data, sequential data; input dimensionality e.g. vector inputs, matrix inputs; computational needs e.g. processing speed, resource constraints, latency requirements

Practical suitability: scalability needs e.g. large-scale data handling, model expansion; suitability for defined tasks e.g. classification tasks, sequence modelling, language understanding.

*Examples of real-world AI tools:*

A range of real-world AI tools e.g.

Convolutional network models: ResNet e.g. image classification, feature extraction; Visual Geometry Group (VGG) e.g. image recognition, pattern detection; MobileNet e.g. lightweight models, mobile vision

Recurrent network models: LSTM models e.g. sequence prediction, time-series analysis; gated recurrent unit (GRU) models e.g. sequential data processing, pattern learning; DeepSpeech e.g. speech recognition, audio processing

Transformer-based models: Bidirectional Encoder Representations from Transformers (BERT) e.g. text understanding, language modelling; GPT e.g. text generation, sequence prediction; Vision Transformer e.g. image analysis, attention-based processing

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Develop a simple training workflow that adapts a pretrained deep learning model to solve a defined problem in a specific scenario**

*Pretrained models and transfer learning concepts:*

Transfer learning principles: pretrained model definition e.g. model reuse, knowledge transfer; transfer learning purpose e.g. reduced training time, improved performance; feature reuse e.g. shared representations, generic feature extraction, pretrained embeddings

Adaptation approaches: basic fine-tuning e.g. parameter updating, task-specific adjustment; frozen layers e.g. fixed early layers, stable low-level features; trainable layers e.g. adaptable upper layers, task-aligned representations, customised outputs.

*Training workflow components:*

Data preparation: data ingestion e.g. dataset loading, data pipeline creation; dataset preparation e.g. preprocessing, normalisation, augmentation; training/validation split e.g. 80/20 division, stratified sampling; examples: ImageNet preprocessing pipelines, Common Objects in Context (COCO) dataset augmentation routines, Hugging Face image normalisation workflows

Workflow configuration: model configuration e.g. architecture selection, pretrained checkpoint loading; loss selection e.g. classification loss, regression loss, task-specific objective; optimisation setup e.g. optimiser choice, learning rate settings, update rules.

*Workflow scope and complexity expectations:*

Workflow design: limited hyperparameter adjustments e.g. simple tuning, learning rate changes; controlled dataset size e.g. manageable batches, reduced subsets; simple model settings e.g. minimal layers changed, default configurations, stable defaults

Process expectations: reproducible process e.g. fixed seeds, documented steps; clear training objectives e.g. accuracy improvement, loss reduction; defined workflow steps e.g. preprocessing, training, validation.

*Model training and debugging considerations:*

Training checks: training stability e.g. smooth loss curve, consistent gradients; monitoring loss values e.g. batch loss, epoch loss; spotting overfitting e.g. validation gap, unstable learning, early peaking

Debugging actions: addressing data issues e.g. data cleaning, balancing; adjusting learning rate e.g. reducing instability, improving convergence; simple debugging steps e.g. checking batch shapes, verifying labels, confirming preprocessing.

*Computational and tooling considerations:*

Tools and frameworks: deep learning frameworks e.g. TensorFlow, PyTorch; hardware acceleration basics e.g. graphics processing unit (GPU) use, parallel processing; training efficiency e.g. batch optimisation, compute utilisation, reduced overhead

Resource management: resource limits e.g. memory constraints, compute availability; simple tool workflows e.g. notebook execution, scripted runs; hardware considerations e.g. GPU selection, VRAM usage.

*Examples of real-world AI applications:*

A range of real-world AI tools e.g.

Convolutional network models: residual neural network (ResNet) e.g. image classification, feature extraction; VGG e.g. image recognition, pattern detection; MobileNet e.g. lightweight models, mobile vision

Recurrent network models: LSTM models e.g. sequence prediction, time-series analysis; GRU models e.g. sequential data processing, pattern learning; DeepSpeech e.g. speech recognition, audio processing

Transformer-based models: BERT e.g. text understanding, language modelling; generative pre-trained transformer (GPT) e.g. text generation, sequence prediction; Vision Transformer e.g. image analysis, attention-based processing

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO4 **Evaluate the performance and limitations of the training workflow developed for the identified scenario**

*Performance measurement and evaluation metrics:*

Classification metrics: accuracy e.g. correct predictions, overall performance; precision e.g. relevance of predicted positives, quality of positive predictions; recall e.g. sensitivity, ability to capture true positives

Generalisation metrics: validation results e.g. validation accuracy, validation loss, consistency across epochs; generalisation performance e.g. unseen data behaviour, distribution alignment.

*Limitations and risk considerations:*

Data risks: data bias e.g. skewed representation, unfair outcomes, demographic imbalance; class imbalance e.g. dominant classes, minority class scarcity; robustness issues e.g. noise sensitivity, vulnerability to variations

Model risks: overfitting risks e.g. memorisation, low generalisation; limited interpretability e.g. opaque reasoning, complex internal representations.

*Workflow effectiveness and constraints:*

Workflow constraints: training efficiency e.g. time per epoch, resource use; computational cost e.g. GPU time, memory load; reproducibility considerations e.g. fixed seeds, repeatable steps

Model suitability: suitability of pretrained model e.g. domain match, task relevance; scalability limits e.g. large dataset handling, model size constraints.

*Improvement and optimisation considerations:*

Model improvements: refining training settings e.g. learning rate adjustments, batch size changes; improving data quality e.g. enhanced preprocessing, better labelling; adjusting model configuration e.g. unfreezing layers, modifying architecture depth

Performance enhancements: addressing identified limitations e.g. balancing classes, improving generalisation; balancing performance trade-off e.g. accuracy vs. training time, complexity vs. interpretability.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the theoretical principles of deep learning</td><td rowspan="5">LO1 and LO2<br><br>D1 Justify, using a selection of real-world examples, how the theoretical principles of deep learning and the characteristics of different deep learning architectures can support effective model learning, stability and performance.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the core theoretical foundations of deep learning.<br><br>P2 Describe, using a selection of real-world examples, the key structural and functional components of deep learning models, including perceptron-based network structures, common activation function types and optimisation mechanisms.</td><td>M1 Analyse, using a selection of real-world examples, how core theoretical principles influence the behaviour and learning process of deep learning models.</td></tr>
<tr><td colspan="2">LO2 Explore the different deep learning architectures and their applications</td></tr>
<tr><td>P3 Explore, using a selection of relevant real-world examples, the different deep learning architectures used in the development of modern AI systems.<br><br>P4 Explain, using a selection of relevant real-world examples, how deep learning architectures are chosen for use in a selection of defined application contexts.</td><td>M2 Evaluate, using a selection of relevant real-world examples, the effectiveness of different deep learning architectures in addressing real-world application requirements.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Develop a simple training workflow that adapts a pretrained deep learning model to solve a defined problem in a specific scenario | | **LO3 and LO4**<br>**D2** Justify how design and development decisions contribute to the effectiveness of the training workflow in meeting the identified scenario requirements. |
| **P5** Design a simple training workflow that uses transfer learning concepts, workflow components and a pretrained deep learning model to solve a defined problem in a specific scenario.<br>**P6** Develop a simple training workflow that adapts a pretrained deep learning model to solve a defined problem in a specific scenario. | **M3** Analyse the effectiveness of the developed training workflow in adapting a pretrained model to the identified scenario. | |
| **LO4** Review the performance and limitations of the training workflow developed for the identified scenario | | |
| **P7** Review the performance and limitations of the training workflow developed for the identified scenario using appropriate metrics.<br>**P8** Demonstrate the training workflow developed for the identified scenario to a range of key stakeholders to obtain feedback and identify strengths and areas for improvement. | **M4** Evaluate, using the stakeholder feedback and the results of the performance review, the effectiveness of the training workflow in meeting the requirements of the identified scenario. | |

## Recommended Resources

### Textbooks

Aggarwal, C.C. (2023) *Neural Networks and Deep Learning: A Textbook*. 2nd edn. Cham: Springer.

Bishop, C.M. and Bishop, H. (2024) *Deep Learning: Foundations and Concepts*. Cham: Springer.

Géron, A. (2022) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd edn. Sebastopol, CA: O'Reilly Media.

Goodfellow, I., Bengio, Y. and Courville, A. (2024 reprint) *Deep Learning*. Cambridge, MA: MIT Press.

Murphy, K.P. (2022) *Probabilistic Machine Learning: An Introduction*. Cambridge, MA: MIT Press.

Zhang, A., Lipton, Z.C., Li, M. and Smola, A. (2023) *Dive into Deep Learning*. Cambridge: Cambridge University Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| developer.nvidia.com/deep-learning | NVIDIA Developer<br>Deep learning concepts, GPU acceleration, optimisation and tooling guidance<br>(Resource Hub) |
| keras.io | Keras Documentation<br>High-level deep learning API documentation and conceptual explanations<br>(Tutorials) |
| pytorch.org/docs/stable/index.html | PyTorch Documentation<br>Framework reference for model training, transfer learning and evaluation<br>(Development tool) |
| scikit-learn.org/stable | scikit-learn<br>Machine Learning in Python<br>(General reference) |

| | |
|---|---|
| tensorflow.org/learn | TensorFlow Learn<br>Deep learning workflows, model training and optimisation guidance<br>(Tutorials) |
| cs231n.github.io/convolutional-networks/ | CS231n:<br>Convolutional Neural Networks – layers, pooling, design patterns and classic CNN case studies<br>(Reference) |
| huggingface.co/learn/computer-vision-course/en/unit3/vision-transformers/vision-transformers-for-image-classification | HuggingFace<br>HF Computer Vision Course: Transfer Learning & Fine-tuning ViT – step-by-step workflow to adapt pre-trained models<br>(Tutorial) |
| jalammar.github.io/illustrated-transformer/ | The Illustrated Transformer (Jay Alammar)<br>Visual walkthrough of encoder/decoder and self-attention<br>(Article/Tutorial) |
| www.kaggle.com/learn/computer-vision | Kaggle Learn:<br>Build convolutional neural networks with TensorFlow and Keras, including transfer learning pipelines<br>(Tutorial) |

## Journals and articles

LeCun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', *Nature*, 521(7553), pp. 436–444. Available at: doi.org/10.1038/nature14539.

Pan, S.J. and Yang, Q. (2010) 'A Survey on Transfer Learning', *IEEE Transactions on Knowledge and Data Engineering*, 22(10), pp. 1345–1359. Available at: doi.org/10.1109/TKDE.2009.191.

Sculley, D. *et al*. (2015) 'Hidden Technical Debt in Machine Learning Systems', in *Advances in Neural Information Processing Systems*, 28. Available at: proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf.

Vaswani, A. *et al*. (2017) 'Attention Is All You Need', in *Advances in Neural Information Processing Systems,* 30. Available at: doi.org/10.48550/arXiv.1706.03762.

## Links

This unit links to the following related units:

*Unit 520: Applied AI*

*Unit 521: Generative and Conversational AI.*

# Unit 527: System Analysis and Design

**Unit code:** M/652/3313

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Modern organisations depend on well-designed information systems to support business operations, decision-making and long-term strategic planning. As digital systems continue to increase in scale and complexity, there is a growing need for professionals who can analyse organisational requirements and translate them into effective system designs. This unit sits within the computing and digital technologies sector and reflects the practical demands of roles that focus on bridging business needs with technical solutions.

The aim of this unit is to develop students' capability to apply structured system analysis and design approaches in a professional context. Students will explore how problems are defined, how requirements are identified and documented, and how appropriate system solutions are designed using recognised methodologies and models. The unit focuses on industry-relevant practices, enabling students to work with approaches and techniques that are aligned to current organisational expectations.

The unit covers a range of interconnected topics that support the full system analysis and design process. Students will examine systems analysis concepts, development methodologies and SDLC models, alongside requirement elicitation and documentation techniques. They will produce and interpret logical and physical system models, and develop the ability to communicate system designs clearly to a range of stakeholders. The content builds progressively, allowing students to connect analysis, modelling and design into a coherent approach to developing system solutions.

On successful completion of this unit, students will be able to analyse requirements, develop structured documentation, design system models and present solutions in a professional context. These skills support progression into roles such as Systems Analyst, Business Analyst, Software Designer, IT Consultant and Digital Project Co-ordinator, as well as further study in areas related to software engineering and information systems.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Define the key systems analysis concepts and methodologies

LO2 Develop system requirement documentation using industry-standard techniques for an identified scenario

LO3 Design logical and physical models to represent system structure and behaviour for an identified scenario

LO4 Present the system design for an identified scenario to a range of key stakeholders.

## Essential Content

### LO1 Define the key systems analysis concepts and methodologies

*Systems analysis concepts:*

Problem framing: systems thinking, e.g. systems theory models, socio-technical system models; organisational context, e.g. enterprise architecture layers, operating models; problem definition, e.g. problem statements, scope boundaries

Business alignment: business objectives, e.g. key performance indicators (KPI targets), strategic objectives; stakeholder identification, e.g. Responsible, Accountable, Consulted, and Informed (RACI) matrices, stakeholder maps; system boundaries, e.g. context diagrams, boundary definitions

Feasibility and constraints: operational constraints, e.g. resource limitations, capacity constraints; feasibility analysis, e.g. cost–benefit analysis, technical feasibility studies; risk assessment, e.g. risk matrices, impact–likelihood models.

*Requirements elicitation approaches:*

Elicitation techniques: stakeholder interviews, e.g. structured interviews, semi-structured interviews; facilitated workshops, e.g. joint application development (JAD) sessions, sprint workshops; direct observation, e.g. workflow observation, task analysis

Analytical techniques: document analysis, e.g. legacy system documentation, process documentation; prototyping techniques, e.g. wireframes, interactive prototypes; scenario analysis, e.g. use-case scenarios, user journey mapping

Validation and prioritisation: requirements validation, e.g. validation workshops, stakeholder sign-off; elicitation planning, e.g. elicitation frameworks, requirements plans; requirements prioritisation, e.g. MoSCoW prioritisation, backlog ranking.

*Systems development methodologies:*

Methodological approaches: structured analysis, e.g. data flow modelling, functional decomposition; object-oriented analysis, e.g. Unified Modelling Language (UML) class models, object associations; agile methodologies, e.g. Scrum frameworks, sprint iterations

Life cycle models: waterfall model, e.g. sequential life cycle phases, stage-gate models; iterative development, e.g. incremental builds, iteration cycles; spiral model, e.g. risk-driven iterations, evaluation loops

Modern practices: rapid application development, e.g. rapid prototyping, time-boxed delivery; lean development, e.g. value stream mapping, waste reduction; DevOps integration, e.g. continuous integration/continuous deployment (CI/CD) pipelines, automated deployment.

*Systems development life cycle (SDLC) models:*

Life cycle structure: SDLC phases, e.g. requirements phase, design phase; life cycle activities, e.g. requirements analysis, system design; deliverables management, e.g. specification documents, design artefacts

Governance and control: documentation standards, e.g. IEEE documentation standards, structured templates; governance checkpoints, e.g. stage reviews, approval gates; quality assurance, e.g. verification reviews, validation testing

Validation and change control: validation processes, e.g. user acceptance testing, system validation; verification activities, e.g. design verification, test verification; change control, e.g. change requests, version control.

LO2 **Develop system requirement documentation using industry-standard techniques for an identified scenario**

*Functional requirements:*

Behaviour and process: system behaviour, e.g. state transitions, service responses; process requirements, e.g. process steps, process rules; workflow definitions, e.g. workflow states, workflow routing

Inputs and outputs: input specifications, e.g. input validation rules, input schemas; output specifications, e.g. report layouts, response payloads; exception handling, e.g. error conditions, fallback handling

Requirement artefacts: business rules, e.g. decision tables, rule constraints; use cases, e.g. use-case scenarios, actor goals; user stories, e.g. backlog items, acceptance criteria.

*Non-functional requirements:*

Performance and capacity: performance metrics, e.g. throughput metrics, latency metrics; response time, e.g. p95 response time, timeout thresholds; scalability requirements, e.g. horizontal scaling, vertical scaling

Security and compliance: security controls, e.g. access control, encryption control; compliance requirements, e.g. regulatory constraints, audit requirements; accessibility standards, e.g. Web Content Accessibility Guidelines (WCAG) criteria, assistive compatibility

Quality and operability: availability targets, e.g. uptime targets, failover objectives; reliability measures, e.g. error rates, fault tolerance; interoperability, e.g. Application Programming Interface (API) interoperability, data interchange.

*Data requirements:*

Data modelling: data entities, e.g. entity types, domain entities; data attributes, e.g. field constraints, attribute definitions; entity relationships, e.g. cardinality rules, relationship constraints

Integrity and quality: data integrity rules, e.g. referential integrity, constraint rules; data validation, e.g. validation rules, data checks; data quality, e.g. completeness metrics, accuracy metrics

Storage and governance: data formats, e.g. JavaScript Object Notation (JSON) formats, comma-separated values (CSV) formats; data storage structures, e.g. relational tables, document stores; metadata definitions, e.g. data dictionaries, schema metadata.

*Requirements documentation standards:*

Control and traceability: requirements traceability, e.g. trace matrices, trace links; version control, e.g. baseline versions, revision history; change management, e.g. change requests, change logs

Documentation artefacts: documentation structure, e.g. software requirements specification (SRS) sections, document hierarchy; specification templates, e.g. SRS templates, use-case templates; standardised notation, e.g. UML notation, Business Process Model and Notation (BPMN) notation

Review and governance: stakeholder approval, e.g. sign-off, acceptance confirmation; review processes, e.g. peer reviews, walkthroughs; quality assurance, e.g. quality assurance (QA) checklists, quality reviews.

LO3 **Design logical and physical models to represent system structure and behaviour for an identified scenario**

*Logical system models:*

Structural modelling: data flow diagrams, e.g. process flow diagrams, data movement diagrams; entity relationship diagrams, e.g. entity–relationship (ER) models, relational schemas; class diagrams, e.g. class structures, inheritance models

Behavioural modelling: use-case diagrams, e.g. actor interactions, system functions; sequence diagrams, e.g. message sequences, object interactions; activity diagrams, e.g. workflow paths, process logic; state machine diagrams, e.g. state transitions, event triggers.

*Physical system models:*

Architecture design: system architecture, e.g. layered architectures, microservices architectures; deployment models, e.g. deployment topologies, node mappings; infrastructure design, e.g. cloud infrastructure, on-premise systems

System configuration: hardware components, e.g. servers, client devices; software environments, e.g. runtime environments, application platforms; network configurations, e.g. network topologies, communication protocols.

*Process and behaviour modelling:*

Process representation: process flows, e.g. process maps, workflow diagrams; interaction flows, e.g. sequence diagrams, message flows

Event and state modelling: state machine diagrams, e.g. state transitions, system states; event-driven modelling, e.g. event triggers, event flows; business process modelling, e.g. BPMN diagrams, process models.

*Model validation and consistency:*

Alignment and coverage: requirements alignment, e.g. requirement traceability, specification mapping; model completeness, e.g. coverage analysis, gap identification; model accuracy, e.g. validation checks, consistency verification

Quality assurance: stakeholder validation, e.g. validation sessions, stakeholder reviews; consistency checking, e.g. model consistency rules, naming standards; traceability mapping, e.g. trace links, dependency mapping.

LO4 **Present the system design for an identified scenario to a range of key stakeholders**

*System design communication:*

Audience and messaging: audience identification, e.g. stakeholder matrices, audience segmentation models; stakeholder profiling, e.g. stakeholder influence grids, RACI models; communication objectives, e.g. measurable communication KPIs, reporting deliverables

Presentation methods: technical presentation, e.g. system architecture diagrams, UML presentation artefacts; non-technical explanation, e.g. business process summaries, executive briefings; data visualisation, e.g. KPI dashboards, system performance charts

Communication formats: reporting formats, e.g. technical specifications (SRS), design reports; presentation structure, e.g. structured slide decks, architectural reviews; documentation support, e.g. design artefacts, UML documentation.

*Design justification:*

Alignment and rationale: requirements alignment, e.g. traceability matrices, requirement mapping; constraint analysis, e.g. capacity constraints, architectural constraints; design rationale, e.g. Architecture Decision Records (ADR), design justifications

Trade-offs and evaluation: trade-off analysis, e.g. cost–performance trade-offs, scalability–complexity trade-offs; optimisation considerations, e.g. performance tuning strategies, cost optimisation models; feasibility justification, e.g. feasibility studies, business case analysis.

*Stakeholder feedback:*

Feedback processes: feedback collection, e.g. structured feedback tools, survey platforms; validation feedback, e.g. acceptance criteria validation, requirement verification; issue identification, e.g. defect logs, requirement gaps

Analysis and response: feedback analysis, e.g. statistical analysis, trend analysis; response management, e.g. issue tracking systems, workflow tracking; iterative refinement, e.g. agile iteration cycles, backlog refinement.

*Presentation standards:*

Documentation quality: professional formatting, e.g. IEEE document standards, structured templates; visual consistency, e.g. UML standards, diagram conventions; terminology accuracy, e.g. domain vocabularies, technical terminology

Delivery standards: presentation delivery, e.g. stakeholder presentations, design walkthroughs; time management, e.g. milestone scheduling, delivery timelines; digital presentation tools, e.g. presentation software, collaboration platforms.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Define the key systems analysis concepts and methodologies</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of systems analysis concepts and methodologies in supporting successful systems development.</td></tr>
<tr><td>P1 Define, using a selection of real-world examples, the key systems analysis concepts and methodologies used in system analysis and design.<br><br>P2 Explain, using a selection of real-world examples, the requirements elicitation processes used in system analysis and design.</td><td>M1 Analyse, using a selection of real-world examples, the benefits and limitations of system analysis concepts and methodologies for different organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Develop system requirement documentation using industry-standard techniques for an identified scenario</td></tr>
<tr><td>P3 Develop system requirement documentation for an identified scenario using industry-standard techniques.<br><br>P4 Review the developed system requirement documentation with a range of identified stakeholders to identify strengths and areas to improve.</td><td>M2 Optimise, using the results of the stakeholder feedback, the system requirements documentation so that it comprehensively meets the requirements of an identified scenario.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design logical and physical models to represent system structure and behaviour for an identified scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the creation of the system designs and documentation in relation to how comprehensively they meet the requirements of the identified scenario. |
| **P5** Develop the design of logical models to represent system structure and behaviour for an identified scenario.<br>**P6** Create the design of physical models to represent system structure and behaviour for an identified scenario. | **M3** Evaluate the effectiveness of the logical and physical model designs in representing system structure and behaviour for the identified scenario. | |
| **LO4** Present the system design for an identified scenario to a range of key stakeholders | | |
| **P7** Develop a presentation on the system design for an identified scenario to a range of key stakeholders.<br>**P8** Present the system design for an identified scenario to a range of key stakeholders to obtain feedback. | **M4** Optimise, using the results of the stakeholder feedback, the system design so that it comprehensively meets the requirements of an identified scenario. | |

## Recommended Resources

### Textbooks

Dennis, A., Wixom, B.H. and Roth, R.M. (2021) *Systems Analysis and Design*. 8th edn. Hoboken, NJ: John Wiley & Sons.

Kendall, K.E. and Kendall, J.E. (2023) *Systems Analysis and Design*. 11th edn. Harlow: Pearson Education.

Laplante, P.A. and Kassab, M.H. (2022) *Requirements Engineering for Software and Systems*. 4th edn. Boca Raton, FL: CRC Press.

Tegarden, D.P. *et al*. (2025) *Systems Analysis and Design: An Object-Oriented Approach with UML*. 7th edn. Hoboken, NJ: Wiley.

Tilley, S. (2024) *Systems Analysis and Design*. 13th edn. Boston, MA: Cengage Learning.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/system-design/system-analysis-system-design/ | GeeksforGeeks<br>System analysis and system design reference<br>(General reference) |
| www.omg.org/spec/UML/ | Object Management Group (OMG)<br>specification for the Unified Modelling Language<br>(General reference) |
| www.geeksforgeeks.org/software-engineering/software-engineering-requirements-elicitation/ | GeeksforGeeks<br>Requirements elicitation methods, including stakeholder interviews, workshops and prototyping techniques for defining system needs<br>(General reference) |
| www.visual-paradigm.com/guide/uml-modeling-a-complete-guide-from-requirements-to-sequence-diagrams/ | Visual Paradigm<br>End-to-end UML modelling from requirements through use cases and sequence diagrams, linking analysis to design artefacts<br>(Tutorial) |

| | |
|---|---|
| www.go-uml.com/gathering-requirements-with-uml-guide/ | GO-UML<br>Use of UML diagrams (use case, activity) to capture and analyse functional requirements and workflows<br>(Tutorial) |
| the-requirements-engineer.com/elicitation-in-requirements-engineering/ | The Requirements Engineer<br>Stakeholder identification, communication techniques and requirement elicitation practices for system development<br>(General reference) |
| programguru.org/uml/uml-in-sdlc/ | ProgramGuru<br>Application of UML across the SDLC, linking requirements, design and implementation phases<br>(Tutorial) |
| www.geeksforgeeks.org/system-design/unified-modeling-language-uml-introduction/ | GeeksforGeeks<br>Overview of UML diagrams for modelling system structure and behaviour and supporting communication between stakeholders<br>(General reference) |
| www.ibm.com/think/topics/data-flow-diagram | IBM<br>Explanation of data flow diagrams and their use in analysing and improving information systems and processes<br>(General reference) |
| www.tutorialspoint.com/system_analysis_and_design/data_flow_diagram_development.htm | TutorialsPoint<br>Step-by-step development of data flow diagrams, including system boundaries, processes, data stores and validation<br>(Tutorial) |

**Journals and articles**

Gadhi, A. *et al*. (2023) 'Applying UML and Machine Learning to Enhance System Analysis and Design', *International Journal of Communications, Network and System Sciences*, 16(5), pp. 67–76. Available at: doi.org/10.4236/ijcns.2023.165005.

Godage, P., Siribaddana, P. and Hewapathirana, R. (2023) 'Sustaining digital health systems through key stakeholder engagement: a critical systems heuristics perspective', *Oxford Open Digital Health*, 1, oqad014. Available at: doi.org/10.1093/oodh/oqad014.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*.

# Unit 528: Robotics

**Unit code:** R/652/3314

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Robotics and autonomous systems play an increasingly important role across a wide range of industries, including manufacturing, logistics, healthcare and service environments. These systems combine mechanical design, electronic control and intelligent decision-making to carry out tasks with speed, precision and consistency. As organisations continue to adopt automation to improve efficiency and quality, there is a growing need for individuals who understand how robotic systems are structured, how they operate, and how they can be applied effectively in real-world contexts.

The aim of this unit is to develop students' understanding of how robotic systems are analysed, designed, implemented and evaluated within defined scenarios. The unit focuses on applying core robotics principles in practical contexts, supporting students to move beyond basic operation and towards making informed decisions about system design, control and performance. It also introduces key considerations such as system reliability, safety and suitability for purpose, reflecting current expectations within the engineering and automation sectors.

The unit covers a structured range of topics that support the development of this knowledge. Students begin by exploring the fundamental principles of robotics, including system architecture, kinematics, sensing and actuation. They then move on to designing robotic systems using appropriate tools, modelling techniques and development approaches. This is followed by the implementation of control algorithms to enable systems to operate effectively, and finally the evaluation of system performance, safety and limitations in relation to specific requirements.

On successful completion of this unit, students will be able to analyse robotic systems, design and develop solutions for defined scenarios, implement control strategies and evaluate system performance and safety. These skills support progression into roles such as Robotics Technician, Automation Engineer, Control Systems Technician, Mechatronics Technician and Robotics Systems Integrator, as well as further study in robotics, automation or related engineering disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core principles of robotics and autonomous systems

LO2 Develop a robotic system using appropriate tools and methodologies for an identified scenario

LO3 Implement control algorithms to allow a robotic system to meet the requirements of an identified scenario

LO4 Review the performance, safety and limitations of the robotic system in meeting the requirements of an identified scenario.

## Essential Content

### LO1 **Explain the core principles of robotics and autonomous systems**

*Robotics system fundamentals:*

Architecture and configuration: Robot architectures, e.g. articulated arms, Selective Compliance Articulated Robot Arm (SCARA) robots; kinematic chains, e.g. open-chain linkages, closed-chain linkages; degrees of freedom, e.g. translational axes, rotational axes; serial manipulators, e.g. 6-axis manipulators, gantry manipulators

Operating limits: operational constraints, e.g. joint limits, speed limits; mechanical limitations, e.g. backlash, compliance; payload capacity, e.g. rated payload, dynamic payload; end effectors, e.g. parallel grippers, vacuum grippers.

*Kinematics and motion principles:*

Kinematics modelling: forward kinematics, e.g. Denavit-Hartenberg (DH) modelling, homogeneous transforms; inverse kinematics (IK), e.g. analytic IK, numerical IK; co-ordinate frames, e.g. base frames, tool frames

Motion representation: position vectors, e.g. Cartesian vectors, joint-space vectors; velocity analysis, e.g. Jacobian analysis, twist representation; motion trajectories, e.g. joint trajectories, Cartesian trajectories

Constraint conditions: singularities, e.g. wrist singularities, elbow singularities; spatial representation, e.g. Euler angles, quaternions; dynamic constraints, e.g. torque limits, jerk limits.

*Actuation and sensing technologies:*

Actuation systems: electric actuators, e.g. servo motors, stepper motors; hydraulic actuators, e.g. hydraulic cylinders, servo-hydraulics; pneumatic actuators, e.g. pneumatic cylinders, pneumatic grippers

Control and positioning: speed control, e.g. servo drives, pulse-width modulation (PWM) control; position sensors, e.g. rotary encoders, linear encoders; force sensors, e.g. load cells, force–torque sensors

Proximity and vision sensing: proximity sensors, e.g. inductive sensors, ultrasonic sensors; vision sensors, e.g. red-green-blue (RGB) cameras, depth cameras; sensor resolution, e.g. encoder resolution, depth resolution.

*Autonomy and control models:*

Control architectures: reactive control, e.g. behaviour-based control, reflex control; deliberative control, e.g. planning-based control, symbolic planning; hybrid control architectures, e.g. three-layer architectures, subsumption architectures

Distributed autonomy: decentralised control, e.g. distributed controllers, multi-agent control; autonomy levels, e.g. teleoperation, supervised autonomy; decision-making models, e.g. finite state machines, Markov decision processes

Adaptive intelligence: behavioural control, e.g. behaviour trees, task behaviours; adaptive control, e.g. gain scheduling, model-reference adaptive control; feedback loops, e.g. proportional–integral–derivative (PID) loops, sensor feedback.

LO2 **Develop a robotic system using appropriate tools and methodologies for an identified scenario**

*System requirements and scenario definition:*

Requirement analysis: Operational constraints, e.g. joint limits, duty cycles; environmental conditions, e.g. IP ratings, dust ingress; performance requirements, e.g. positional accuracy, cycle times; safety considerations, e.g. safety-rated monitored stop, safeguarded zones

Specification development: functional specifications, e.g. pick-and-place tasks, waypoint sequences; user requirements, e.g. teach pendant controls, human-machine interface (HMI) inputs; system objectives, e.g. throughput targets, uptime targets.

*Mechanical and structural design principles:*

Mechanical configuration: mechanism selection, e.g. 6-axis manipulators, SCARA mechanisms; load analysis, e.g. static load cases, dynamic torque loads; material properties, e.g. yield strength, fatigue limits; structural stability, e.g. modal stiffness, vibration damping

Kinematic design: degrees of freedom optimisation, e.g. axis selection, redundancy resolution; kinematic configuration, e.g. DH parameters, link geometry; joint design, e.g. harmonic drive joints, ball-screw joints; actuator placement, e.g. base-mounted motors, direct-drive joints.

*Design tools and modelling techniques:*

Modelling tools: computer-aided design (CAD) modelling, e.g. parametric assemblies, tolerance stack-ups; kinematic simulation, e.g. inverse kinematics validation, workspace simulation; dynamic simulation, e.g. multibody dynamics, torque profiling

Simulation environments: digital prototyping, e.g. digital twin models, virtual commissioning; simulation environments, e.g. Gazebo simulation, MATLAB/Simulink models; virtual testing, e.g. collision checking, scenario simulation.

*Methodologies for robotic system design:*

Design approaches: systems engineering approaches, e.g. V-model life cycle, requirements traceability; modular design principles, e.g. modular end effectors, quick-change tooling; iterative design processes, e.g. prototyping iterations, design sprints

Validation strategies: simulation-driven validation, e.g. model-in-the-loop, scenario-based validation; design trade-offs, e.g. payload versus reach, speed versus precision; integration strategies, e.g. Robot Operating System (ROS) integration, fieldbus integration.

*Testing and validation approaches:*

Testing methods: unit testing, e.g. sensor driver tests, control module tests; integration testing, e.g. interface testing, communications bus testing; system testing, e.g. acceptance testing, mission profile testing

Validation processes: hardware-in-the-loop testing, e.g. real-time plant models, controller-in-loop; performance testing, e.g. accuracy benchmarking, cycle-time benchmarking; reliability testing, e.g. endurance cycling, mean time between failure (MTBF) testing.

LO3 **Implement control algorithms to allow a robotic system to meet the requirements of an identified scenario**

*Control system foundations:*

Control architectures: open-loop control, e.g. feedforward control, open-loop actuation; closed-loop control, e.g. feedback control loops, error correction systems; feedback systems, e.g. PID feedback, sensor feedback loops

Control dynamics: control loops, e.g. single-loop systems, multi-loop systems; real-time control, e.g. real-time processing, deterministic control; system response, e.g. transient response, steady-state response.

*Control algorithms and strategies:*

PID control: PID control, e.g. PID loops; proportional control, e.g. gain response; integral control, e.g. accumulated error correction; derivative control, e.g. rate of change damping

Algorithm tuning and performance: gain tuning, e.g. manual tuning, Ziegler–Nichols method; control parameters, e.g. gain coefficients; stability analysis, e.g. response stability; responsiveness optimisation, e.g. rise time, overshoot control

Advanced strategies: trajectory tracking, e.g. path-following control; adaptive control, e.g. parameter adaptation; model-based control, e.g. system model prediction; algorithm efficiency, e.g. computation time.

*Path planning and motion control:*

Motion planning: point-to-point motion, e.g. joint-space motion, waypoint control; trajectory generation, e.g. spline interpolation, polynomial trajectories; path optimisation, e.g. shortest-path algorithms, energy-efficient paths

Navigation control: obstacle avoidance, e.g. potential field methods, collision avoidance algorithms; motion interpolation, e.g. linear interpolation, spline interpolation; route planning, e.g. graph-based planning, grid-based planning

Spatial control: kinematic constraints, e.g. joint limits, workspace limits; dynamic constraints, e.g. acceleration limits, torque limits; motion profiling, e.g. trapezoidal profiles, S-curve profiles.

*Embedded control and implementation platforms:*

Control hardware: microcontrollers, e.g. Arduino controllers, ARM-based controllers; real-time controllers, e.g. programmable logic controller (PLCs), embedded real-time operating system (RTOSs); embedded systems, e.g. control units, embedded processors

System integration: firmware development, e.g. embedded firmware, control firmware; input–output interfaces, e.g. general-purpose input/output (GPIO) interfaces, analog-to-digital converter (ADC) channels; communication protocols, e.g. universal asynchronous receiver-transmitter (UART) protocols, CAN bus

Resource management: sensor integration, e.g. encoder integration, sensor fusion inputs; actuator interfaces, e.g. motor drivers, servo interfaces; timing constraints, e.g. scheduling constraints, real-time deadlines.

LO4 **Review the performance, safety and limitations of the robotic system in meeting the requirements of an identified scenario**

*Performance evaluation criteria:*

Measurement characteristics: accuracy, e.g. absolute position error, end-effector error; repeatability, e.g. cycle deviation, return accuracy; speed, e.g. joint velocity, end-effector velocity

Operational performance: throughput, e.g. parts-per-hour, cycle output; operational efficiency, e.g. energy efficiency, utilisation rate

Assurance and improvement: system reliability, e.g. MTBF, fault rate; benchmarking, e.g. reference benchmarks, baseline tests.

*Safety principles and standards:*

Safety engineering: functional safety concepts, e.g. safety functions, safety life cycle; safety integrity levels (SIL), e.g. SIL targets, SIL ratings; fail-safe mechanisms, e.g. safe torque off, brake systems; emergency stop systems, e.g. E-stop circuits, safety relays

Risk management: risk assessment, e.g. risk matrices, scoring models; hazard identification, e.g. collision hazards, pinch points; protective measures, e.g. safety fencing, interlocks

Standards and collaboration: compliance standards, e.g. audit requirements, conformity checks; ISO 10218, e.g. industrial robot safety, integration safety; human-robot collaboration, e.g. collaborative operation, shared workspace.

*System limitations and constraints:*

Physical constraints: mechanical limitations, e.g. backlash, compliance; payload constraints, e.g. payload limits, moment limits; range limitations, e.g. reach envelope, joint limits

Sensing and control constraints: sensor uncertainty, e.g. measurement noise, drift; control instability, e.g. oscillation, instability; system latency, e.g. control delay, processing delay

Environment and integration: environmental sensitivity, e.g. thermal drift, vibration; integration constraints, e.g. interface incompatibility, timing mismatch; energy consumption, e.g. power draw, thermal load.

*Failure and risk analysis:*

Fault analysis: fault identification, e.g. fault codes, diagnostics; failure modes, e.g. actuator failure, sensor failure; root cause analysis, e.g. causal analysis, fault tracing

Reliability and resilience: reliability analysis, e.g. failure mode and effects analysis (FMEA), fault trees; fault tolerance, e.g. redundancy, failover mechanisms

Recovery and mitigation: recovery strategies, e.g. restart procedures, safe recovery; maintenance planning, e.g. preventive maintenance, scheduled servicing.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explain the core principles of robotics and autonomous systems | | |
| **P1** Explain, using a selection of real-world examples, the core principles of robotics systems, including architecture, kinematics, sensing, and actuation.<br><br>**P2** Review, using a selection of real-world examples, how autonomy and control models enable robotic system behaviour in applied contexts. | **M1** Analyse, using a selection of real-world examples, how robotics system components and control models interact to enable autonomous behaviour. | **D1** Critically evaluate, using a selection of real-world examples, the effectiveness of robotics principles and control approaches in supporting autonomous system functionality across different application contexts. |

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO2 Develop a robotic system using appropriate tools and methodologies for an identified scenario</td><td rowspan="6">LO2, LO3 and LO4<br><br>D2 Justify the decisions made in the design, development and implementation of the robotic system and control algorithms that enable it to comprehensively meet the needs of an identified scenario as well as performance and safety considerations.</td></tr>
<tr><td>P3 Design a robotic system using appropriate tools and methodologies to meet the requirements of an identified scenario.<br><br>P4 Develop a robotic system using appropriate tools and methodologies to meet the requirements of an identified scenario.</td><td>M2 Evaluate methodologies used in the design and development of a robotic system to meet the requirements of an identified scenario.</td></tr>
<tr><td colspan="2">LO3 Implement control algorithms to allow a robotic system to meet the requirements of an identified scenario</td></tr>
<tr><td>P5 Implement control algorithms to enable a robotic system to meet the requirements of an identified scenario.<br><br>P6 Review the implementation of the control algorithms to identify strengths and areas for improvement.</td><td>M3 Optimise the implemented control algorithms, using results from the review, to improve system performance against the requirements of the identified scenario.</td></tr>
<tr><td colspan="2">LO4 Review the performance, safety and limitations of the robotic system in meeting the requirements of an identified scenario</td></tr>
<tr><td>P7 Apply a test plan to determine the performance, safety, and limitations of the robotic system and control algorithms in meeting the requirements of an identified scenario.<br><br>P8 Review, using the results of the test plan, the performance, safety, and limitations of the robotic system and control algorithms in relation to the identified scenario.</td><td>M4 Evaluate the performance of the implemented robotic system and control algorithms in how well they meet the requirements of an identified scenario.</td></tr>
</table>

## Recommended Resources

### Textbooks

Corke, P. (2023) *Robotics, Vision and Control: Fundamental Algorithms in MATLAB* 2nd edn. Cham: Springer.

Craig, J. (2024) *Introduction to Robotics: Mechanics and Control*. 4th edn. Harlow: Pearson.

Dudek, G. and Jenkin, M. (2024) *Computational Principles of Mobile Robotics*. 3rd edn. Cambridge: Cambridge University Press.

Kakoty, N., Goswami, R. and Vinjamuri, R. (2025) *Introduction to Embedded Systems and Robotics: A Practical Guide*. Cham: Springer.

Ross, L., Fardo, S. and Walach, M. (2023) *Industrial Robotics Fundamentals: Theory and Applications*. 4th edn. Tinley Park, IL: Goodheart-Willcox.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| https://blog.ansi.org/ansi/iso-10218-1-2025-robots-and-robotic-devices-safety/ | American national Standards Institute<br>ISO 10218-1:2025—Robots And Robotic Devices Safety<br>(Blog) |
| thinkrobotics.com/blogs/tutorials/ros2-tutorial-for-beginners-your-complete-guide-to-robot-operating-system-2 | ThinkRobotics<br>ROS2 Tutorial for Beginners<br>(Tutorial) |
| wiki.ros.org/ROS/Tutorials | ROS Wiki<br>Core ROS Tutorials covering nodes, topics and system setup<br>(Tutorial) |
| www.mathworks.com/discovery/inverse-kinematics.html | MathWorks<br>Overview of inverse kinematics and motion planning in robotics<br>(General reference) |
| scikit-robot.readthedocs.io/en/latest/tutorials/inverse_kinematics.html | Scikit-Robot<br>Inverse kinematics usage and implementation guidance<br>(Tutorial) |

| | |
|---|---|
| docs.wpilib.org/en/stable/docs/software/advanced-controls/introduction/introduction-to-pid.html | FIRST<br>Robotics Explanation of PID control concepts and control system terminology<br>(General reference) |
| www.societyofrobots.com/programming_PID.shtml | Society of Robots<br>PID control theory with robotics programming examples<br>(Tutorial) |

**Journals and articles**

Demir, N. *et al*. (2026) 'Design, Implementation, and Performance Optimization of a ROS Based Autonomous Mobile Robot for Intralogistics in Manufacturing Facilities', *Journal of Automation*, *Mobile Robotics and Intelligent Systems*, 20(1), pp. 93–102. Available at: doi.org/10.14313/jamris-2026-010.

Farkh, R. and Aljaloud, K. (2023) 'Vision Navigation Based PID Control for Line Tracking Robot', *Intelligent Automation & Soft Computing*, 35(1), pp. 901–911. Available at: doi.org/10.32604/iasc.2023.027614.

Shao, Z. and Zhang, F. (2026) 'Advances in Robotics and Autonomous Systems: Transitioning from Automation to Intelligent Collaboration', *Applied Sciences*, 16(4), Article 1966. Available at: doi.org/10.3390/app16041966.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set).*

 © Pearson Education Limited 2026

# Unit 529: Game Design and Scripting

**Unit code:** T/652/3315

**Unit level:** 5

**Credit value:** 15

---

## Introduction

The interactive digital games sector continues to expand as a significant creative and technical industry, combining artistic design, software development and user experience. Modern games depend on scripting to control gameplay systems, interactions and real-time responses within virtual environments. This unit sits within the wider games development and interactive media landscape, supporting progression into industry roles and reflecting current practices used in game engines and development pipelines.

The aim of this unit is to develop students' ability to design and implement scripts that support interactive gameplay. It focuses on how programming constructs are applied to control behaviour, how assets and scripts are brought together within a working prototype, and how gameplay decisions influence player experience. The unit encourages students to apply structured development approaches, reflecting professional workflows used in the production, testing and refinement of games.

The unit covers a range of topics central to game design and scripting. Students explore scripting languages, programming constructs and event-driven systems used in gameplay development. They work with asset integration and behaviour systems to build functional prototypes, while also considering user interface design and interaction. The unit introduces gameplay balancing concepts, including difficulty, pacing and progression, and concludes with evaluation methods that use stakeholder feedback to review and improve a prototype.

On successful completion of this unit, students will be able to design and develop scripted game prototypes, integrate assets and behaviours effectively, and evaluate gameplay experiences using feedback and testing. These skills prepare students for progression to further study or employment in areas such as Game Programmer, Gameplay Designer, Technical Designer, UI Developer and Interactive Media Developer, while also supporting transferable skills in problem-solving, analysis and collaborative development.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Develop game scripts using appropriate programming constructs that meet the requirements of an identified game scenario

LO2 Integrate assets and scripted behaviours into a prototype that meets the requirements of an identified game scenario

LO3 Implement gameplay balancing concepts into the prototype that meets the requirements of an identified game scenario

LO4 Review the effectiveness of the prototype with a range of key stakeholders.

## Essential Content

LO1 **Develop game scripts using appropriate programming constructs that meet the requirements of an identified game scenario**

*Game scripting languages and environments:*

Language structures: scripting languages, e.g. Lua scripting, JavaScript scripting; syntax structures, e.g. syntax rules, code structure; interpreted execution, e.g. runtime execution, script interpretation

Execution environments: compiled execution, e.g. bytecode compilation, script compilation; engine scripting environments, e.g. Unity scripting, Unreal scripting; runtime environments, e.g. game runtime, execution context

Development integration: scripting APIs, e.g. engine Application Programming Interfaces (APIs), library APIs; middleware integration, e.g. physics middleware, audio middleware; development environments, e.g. integrated development environments (IDEs), scripting tools.

*Programming constructs for gameplay:*

Control and logic: control flow structures, e.g. branching instructions, execution pipelines; conditional statements, e.g. boolean expressions, switch-case logic; iterative loops, e.g. loop iteration control, frame update loops

Code structuring: functions and methods, e.g. callable routines, method definitions; variables and data types, e.g. primitive types, structured data types; object-oriented programming, e.g. class hierarchies, polymorphism

Modularity and reuse: modular scripting, e.g. modular architectures, script components; encapsulation principles, e.g. data encapsulation, interface abstraction; code reuse strategies, e.g. reusable modules, shared libraries.

*Event-driven scripting:*

Input and triggers: input handling, e.g. input polling systems, event-based input; event listeners, e.g. listener interfaces, callback registration; triggers, e.g. collision triggers, event triggers

Event processing: callbacks, e.g. callback functions, delegate methods; state transitions, e.g. finite state transitions, state updates; interaction events, e.g. player interaction events, system events

System co-ordination: event queues, e.g. event buffering, queue processing; messaging systems, e.g. publish–subscribe messaging, event dispatching; game state management, e.g. state machines, global state controllers.

*Script performance and reliability:*

Performance optimisation: code efficiency, e.g. algorithmic optimisation, execution profiling; performance optimisation, e.g. CPU optimisation, frame rate optimisation; memory management, e.g. heap allocation, garbage collection

Code quality: readability standards, e.g. coding conventions, code formatting; maintainability practices, e.g. modular architecture, documentation standards; error handling, e.g. exception handling, fault tolerance

Testing and debugging: debugging techniques, e.g. breakpoint debugging, runtime inspection; profiling tools, e.g. CPU profilers, memory profilers; execution performance, e.g. frame timing, latency optimisation.

LO2 **Integrate assets and scripted behaviours into a prototype that meets the requirements of an identified game scenario**

*Asset integration pipelines:*

Asset workflows: asset import workflows, e.g. FBX import pipelines, automated asset ingestion; asset formats, e.g. FBX models, PNG textures; resource management, e.g. asset bundling, memory allocation

Pipeline management: dependency management, e.g. asset dependency graphs, build dependencies; asset compression, e.g. texture compression formats, mesh compression; optimisation techniques, e.g. level of detail optimisation, texture atlasing

Asset organisation: asset versioning, e.g. Git version control, asset revision tracking; pipeline automation, e.g. build pipelines, continuous integration; file organisation, e.g. asset hierarchies, directory structures.

*Scripted behaviour integration:*

Behaviour linking: script–asset linking, e.g. component references, prefab binding; behaviour controllers, e.g. state controllers, behaviour scripts; animation triggers, e.g. animation events, state triggers

Behaviour logic: interaction logic, e.g. player interaction logic, physics interactions; state-based behaviours, e.g. finite state machines, animation states; event-driven behaviours, e.g. event handlers, observer patterns

Runtime co-ordination: component-based scripting, e.g. entity component systems, modular components; runtime integration, e.g. engine runtime integration, system execution; behaviour synchronisation, e.g. animation synchronisation, network synchronisation.

*User interface and interaction elements:*

Interface components: heads-up display (HUD) components, e.g. health bars, score counters; menu systems, e.g. main menu systems, pause interfaces; user feedback systems, e.g. hit markers, audio cues

Interaction design: input mapping, e.g. configurable key bindings, controller mappings; control schemes, e.g. keyboard layouts, gamepad schemes; accessibility features, e.g. subtitle systems, colourblind modes

User experience flow: interface responsiveness, e.g. frame rate responsiveness, input latency handling; interaction design, e.g. user interface (UI) interaction patterns, usability heuristics; navigation systems, e.g. menu navigation flows, UI routing.

*Prototype functionality requirements:*

Functional requirements: scenario requirements, e.g. gameplay specifications, design constraints; functional consistency, e.g. deterministic behaviour, state consistency; gameplay flow, e.g. level progression, gameplay sequencing

System performance: responsiveness, e.g. input latency optimisation, frame timing; system stability, e.g. runtime stability, crash handling; performance optimisation, e.g. CPU optimisation, graphics processing unit (GPU) optimisation

Quality assurance: bug handling, e.g. issue tracking systems, defect resolution; state management, e.g. game state machines, session states; integration testing, e.g. automated testing, regression testing.

LO3 **Implement gameplay balancing concepts into the prototype that meets the requirements of an identified game scenario**

*Gameplay balancing principles:*

Difficulty management: difficulty curves, e.g. exponential curves, logarithmic curves; challenge scaling, e.g. dynamic difficulty adjustment algorithms, player performance scaling; pacing control, e.g. time-step pacing, event timing systems

Progression systems: game progression, e.g. XP progression curves, skill trees; reward systems, e.g. reward loops, loot tables; game economy balancing, e.g. resource economies, currency sinks

Balance frameworks: risk–reward balance, e.g. probability weighting, reward distribution; level design balance, e.g. spatial difficulty scaling, encounter balancing; fairness principles, e.g. rule consistency, system predictability.

*Player experience considerations:*

Engagement factors: player engagement, e.g. engagement metrics, session analytics; immersion, e.g. environmental immersion, system feedback loops; usability, e.g. control responsiveness, interaction latency

User response: accessibility, e.g. adaptive UI systems, assistive gameplay features; frustration management, e.g. fail-state thresholds, retry systems; skill development, e.g. player skill progression, mechanic mastery

Experience modelling: cognitive load, e.g. input complexity, decision density; learning curves, e.g. onboarding curves, progression gradients; player retention, e.g. retention analytics, session frequency.

*Balancing techniques:*

Tuning methods: parameter tuning, e.g. stat balancing, coefficient tuning; difficulty adjustment, e.g. dynamic difficulty systems, scaling modifiers; mechanic calibration, e.g. physics parameter tuning, AI behaviour tuning

Iterative refinement: playtesting feedback, e.g. telemetry data, playtest analytics; refinement, e.g. iteration cycles, build iteration; feedback integration, e.g. data-driven adjustments, feedback pipelines

System balancing: adaptive difficulty systems, e.g. AI difficulty scaling, dynamic balancing systems; variable balancing, e.g. stat normalisation, system coefficients; configuration adjustments, e.g. config files, runtime parameters.

*Impact of balancing decisions:*

Gameplay outcomes: challenge versus accessibility, e.g. difficulty trade-offs, accessibility thresholds; player motivation, e.g. reward incentives, progression drivers; engagement levels, e.g. engagement analytics, interaction metrics

Player response metrics: retention rates, e.g. retention analytics, churn metrics; perceived difficulty, e.g. difficulty perception models, player feedback data; user satisfaction, e.g. satisfaction metrics, usability scores

System performance impact: behavioural responses, e.g. player behaviour patterns, interaction analytics; progression effectiveness, e.g. progression metrics, completion rates; gameplay consistency, e.g. system determinism, mechanic consistency.

LO4 **Review the effectiveness of the prototype with a range of key stakeholders**

*Stakeholder identification:*

Stakeholder groups: players, e.g. target player segments, playtest cohorts; developers, e.g. gameplay programmers, systems engineers; designers, e.g. gameplay design roles, level design teams

Production roles: testers, e.g. quality assurance (QA) test pipelines, automated test suites; clients, e.g. commissioning stakeholders, product owners; publishers, e.g. distribution platforms, publishing pipelines.

*Review and feedback approaches:*

Feedback collection: user feedback, e.g. player telemetry data, usability analytics; playtesting sessions, e.g. controlled playtests, A/B testing sessions; observation analysis, e.g. gameplay telemetry tracking, behavioural analytics

Evaluation methods: performance monitoring, e.g. frame timing metrics, performance profiling logs; usability testing, e.g. heuristic evaluation, interaction testing frameworks; analytics data, e.g. gameplay telemetry, session analytics

Reporting tools: survey tools, e.g. digital survey platforms, feedback aggregation tools; interview techniques, e.g. structured stakeholder interviews, usability interviews; issue tracking, e.g. issue tracking systems, defect management tools.

*Evaluation criteria for effectiveness:*

Functional assessment: functionality, e.g. feature validation, system state validation; gameplay experience, e.g. interaction flow, player experience metrics; alignment with requirements, e.g. design specifications, acceptance criteria

Technical assessment: technical performance, e.g. frame rate profiling, resource utilisation; stability, e.g. crash diagnostics, runtime stability; responsiveness, e.g. input latency metrics, frame timing response

User evaluation: user satisfaction, e.g. satisfaction metrics, feedback scores; accessibility, e.g. Web Content Accessibility Guidelines (WCAG) compliance, assistive feature integration; engagement levels, e.g. engagement analytics, session duration metrics.

*Improvement considerations:*

Issue identification: strength identification, e.g. feature performance metrics, system benchmarks; limitation analysis, e.g. bottleneck analysis, usability issue tracking; prioritisation of changes, e.g. issue ranking matrices, backlog prioritisation

Refinement strategy: iterative refinement, e.g. agile iteration cycles, sprint-based updates; optimisation strategies, e.g. performance profiling optimisation, resource optimisation; bug fixing, e.g. defect resolution workflows, patch deployment

Release management: feature enhancement, e.g. feature iteration pipelines, gameplay updates; update planning, e.g. release scheduling, version control planning; continuous improvement, e.g. continuous integration/continuous development (CI/CD) pipelines, iterative deployment.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Develop game scripts using appropriate programming constructs that meet the requirements of an identified game scenario</td><td rowspan="10">LO1, LO2 and LO3<br><br>D1 Justify the decisions made in the development and integration of the game scripts, behaviours, assets and gameplay balancing that allow the prototype to effectively meet the requirements of an identified game scenario.</td></tr>
<tr><td>P1 Develop game scripts using appropriate programming constructs to support defined gameplay functionality.<br><br>P2 Review, using the results of testing, the performance and reliability of the developed scripts in meeting the requirements of an identified game scenario.</td><td>M1 Evaluate the effectiveness of the developed game scripts in meeting the requirements of the identified game scenario.</td></tr>
<tr><td colspan="2">LO2 Integrate assets and scripted behaviours into a prototype that meets the requirements of an identified game scenario</td></tr>
<tr><td>P3 Integrate assets and scripted behaviours into a functional prototype that meets the requirements of an identified game scenario.<br><br>P4 Implement user interface and user interaction elements into a functional prototype that meets the requirements of an identified game scenario.</td><td>M2 Evaluate the effectiveness of the scripted behaviours and integrated assets in meeting the requirements of the identified game scenario.</td></tr>
<tr><td colspan="2">LO3 Implement gameplay balancing concepts into the prototype that meets requirements of an identified game scenario</td></tr>
<tr><td>P5 Implement gameplay balancing concepts within the prototype to meet the requirements of the identified game scenario.<br><br>P6 Review, using the results of testing, the performance of the implemented balancing concepts within the prototype in meeting the requirements of the identified scenario.</td><td>M3 Evaluate the effectiveness of gameplay balancing concepts applied in optimising player engagement and challenge.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Review the effectiveness of the prototype with a range of key stakeholders | | |
| **P7** Present the prototype to a range of key stakeholders to obtain feedback.<br><br>**P8** Review the effectiveness of the prototype using feedback from a range of key stakeholders to identify strengths and limitations of the prototype. | **M4** Optimise, using the results of the stakeholder feedback, the functional game prototype so that it effectively meets the requirements of the identified game scenario. | **D2** Critically evaluate the results of the stakeholder feedback against the optimised prototype to identify areas where the development phases could be improved in meeting the requirements of an identified scenario. |

## Recommended Resources

### Textbooks

Clark, O. (2026) *Playing with Balance: Game Economy Design*. Boca Raton, FL: CRC Press.

Fullerton, T. (2024) *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. 4th edn. Boca Raton, FL: CRC Press.

Mycka, M. (2026) *Games User Research Cookbook: Tools and Techniques for Better Player Experience*. Boca Raton, FL: CRC Press.

Rabin, S. (2024) *Game Development and Production*. 2nd edn. Boca Raton, FL: CRC Press.

Schell, J. (2023) *The Art of Game Design: A Book of Lenses*. 3rd edn. Boca Raton, FL: CRC Press.

Schreiber, I. and Romero, B. (2021) *Game Balance*. Boca Raton, FL: CRC Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| gamedevbeginner.com/unity/scripting-basics/ | Game Dev Beginner<br>Introductory Unity scripting tutorials covering variables, functions and gameplay logic<br>(Tutorial) |
| moldstud.com/articles/p-beginners-guide-to-scripting-in-unity-first-steps-to-master-game-logic | MoldStud<br>Beginner overview of scripting concepts in games including input handling and basic gameplay logic<br>(Tutorial) |
| algocademy.com/blog/introduction-to-event-driven-programming-a-comprehensive-guide/ | AlgoCademy<br>Explanation of event-driven programming including events, handlers and asynchronous behaviour<br>(Blog) |
| www.numberanalytics.com/blog/ultimate-guide-event-handling-game-programming | Number Analytics<br>Guide to event handling in games covering input events, state changes and interaction systems<br>(Article) |

 © Pearson Education Limited 2026

| | |
|---|---|
| pulsegeek.com/articles/game-asset-pipeline-explained-from-dcc-to-runtime/ | PulseGeek<br>Overview of asset pipelines from creation tools through to runtime use in game engines<br>(General reference) |
| www.thefastcode.com/en-usd/article/understand-the-unity-asset-import-pipeline | The FastCode<br>Step-by-step tutorial on importing and organising assets and integrating them into a game project<br>(Tutorial) |
| gamedesignskills.com/game-design/game-balance/ | Game Design Skills<br>Comprehensive guide to balancing mechanics, difficulty and player progression systems<br>(General reference) |
| moldstud.com/articles/p-playtesting-guide-for-designers-creating-engaging-player-experiences | MoldStud<br>Guide to playtesting methods including feedback collection, analytics and iterative improvement<br>(Tutorial) |

### Journals and articles

Deininger, M. *et al*. (2019) 'Prototyping for context: exploring stakeholder feedback based on prototype type, stakeholder group and question type', *Research in Engineering Design*, 30, pp. 305–325. Available at: doi.org/10.1007/s00163-019-00317-5.

Pfau, J. and El-Nasr, M. S. (2023) 'Balancing Video Games: A Player-Driven Instrument', *CHI PLAY Companion Proceedings*. Available at: doi.org/10.1145/3573382.3616097.

Marín-Lora, C. and Chover, M. (2025) 'GameScript: a simplified scripting language for video game development', *Multimedia Systems*, 31, Article 70. Available at: doi.org/10.1007/s00530-024-01574-8.

Rubio Tamayo, J. L. and Serna Bernal, M. A. (2026) 'Development of a Generative AI-Based Workflow for the Design and Integration of 3D Assets in XR Environments for Research', *Multimedia*, 2(2), Article 6. Available at: doi.org/10.3390/multimedia2020006.

### Links

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*.

# Unit 530: Information Security Management

**Unit code:** Y/652/3316

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Information security management underpins the safe and reliable use of digital systems across modern organisations. As organisations become increasingly dependent on data and interconnected technologies, the protection of information assets has become a core business function. Effective information security is not limited to technical controls, but involves governance, risk management and organisational practices that support resilience, compliance and trust.

The aim of this unit is to develop students' ability to understand and apply the principles that guide information security management within organisational contexts. It focuses on how security is planned, governed and implemented, enabling students to examine how risks are identified, assessed and managed. The unit prepares students to make informed decisions about protecting information assets while considering regulatory and organisational requirements.

The unit covers four interconnected areas. Students first examine the core principles of information security, including governance frameworks, standards and organisational culture. They then explore risk assessment and risk management approaches used to identify and respond to threats in digital environments. This is followed by an exploration of security control models, access management and technical mechanisms used to protect systems and data. Finally, students apply this knowledge to develop and present a regulatory-compliant information management strategy that meets an identified organisational need.

On successful completion of this unit, students will be able to analyse information security practices, evaluate risk management approaches and design appropriate strategies to protect organisational information assets. These skills support progression into roles such as Information Security Analyst, Security Operations Analyst, IT Risk Officer, Information Governance Officer and Cybersecurity Consultant.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the core principles of information security management

LO2 Review the organisational risk assessment and risk management theories used in securing digital environments

LO3 Explore the models for designing and implementing security controls to protect organisational information assets

LO4 Present a regulatory compliant information management strategy that meets an identified organisational need to a range of stakeholders.

 © Pearson Education Limited 2026

## Essential Content

LO1 **Discuss the core principles of information security management**

*Information security principles:*

Security principles: confidentiality, e.g. role-based access control (RBAC), data encryption at rest; integrity, e.g. hashing (SHA-256), checksums; availability, e.g. load balancing, failover clustering

Data protection: data protection, e.g. General Data Protection Regulation (GDPR) compliance controls, data classification schemes; privacy principles, e.g. consent logging, purpose limitation; secure data handling, e.g. Transport Layer Security (TLS) 1.3 transmission, encrypted storage (AES-256)

Governance and trust: accountability, e.g. audit logging, Security Information and Event Management (SIEM) tracking; trust management, e.g. Public Key Infrastructure (PKI), certificate authorities; non-repudiation, e.g. digital signatures, signed transaction logs.

*Information security governance:*

Organisational structures: organisational governance structures, e.g. ISO 27001 information security management system (ISMS) frameworks, security steering committees; roles and responsibilities, e.g. Chief Information Security Officer (CISO) ownership, data owners; senior management accountability, e.g. board-level risk oversight, security governance reporting

Control and oversight: policy frameworks, e.g. information security policy suites, acceptable use policies; internal controls, e.g. access control enforcement, segregation of duties; compliance monitoring, e.g. internal audits, regulatory compliance checks

Governance management: governance models, e.g. centralised Security Operations Centre (SOC), federated security models; risk ownership, e.g. asset-based risk ownership, control ownership; governance alignment, e.g. alignment to ISO 27001 clauses, NIST Cybersecurity Framework (CSF) functions.

*Information security standards and frameworks:*

Standards and frameworks: ISO/IEC 27001, e.g. Annex A controls, certification scope; ISO/IEC 27002, e.g. control guidance, implementation techniques; NIST CSF, e.g. identify, protect, detect functions; knowledge currency, emerging trends, contemporary practice

Compliance and certification: regulatory compliance standards, e.g. General Data Protection Regulation (GDPR), Network and Information Systems (NIS) Regulations; certification processes, e.g. ISO certification audits, external assessments; audit requirements, e.g. audit scope definition, evidence collection

Implementation and integration: control frameworks, e.g. Center for Internet Security (CIS) Critical Security Controls, ISO control sets; implementation guidelines, e.g. secure configuration baselines, hardening standards; framework integration, e.g. ISO-NIST mapping, integrated Governance, Risk and Compliance (GRC) tooling.

*Information security culture:*

Behaviour and awareness: security awareness, e.g. phishing simulation campaigns, secure coding awareness; organisational behaviour, e.g. policy adherence, least privilege usage; insider threat awareness, e.g. behavioural monitoring, anomaly detection

Ethics and responsibility: ethical responsibilities, e.g. responsible disclosure, professional codes (ISC2/BCS); risk ownership, e.g. accountable control owners, risk acceptance sign-off; compliance culture, e.g. audit readiness, policy compliance tracking

Training and improvement: user training programmes, e.g. role-based security training, SOC analyst training; security communication, e.g. incident reporting channels, alert escalation processes; continuous improvement, e.g. corrective action plans, maturity model improvement.

LO2 **Review the organisational risk assessment and risk management theories used in securing digital environments**

*Risk concepts:*

Core risk elements: risk, e.g. organisational risk, security risk; threat, e.g. cyber threats, insider threats; vulnerability, e.g. system vulnerabilities, configuration weaknesses

Risk evaluation factors: asset value, e.g. information assets, system assets; impact, e.g. business impact, operational impact; likelihood, e.g. probability estimation, threat likelihood

Risk tolerance parameters: risk appetite, e.g. organisational risk appetite, strategic risk thresholds; risk tolerance, e.g. acceptable risk levels, tolerance limits; exposure identification, e.g. threat exposure, vulnerability exposure.

*Digital environment risks:*

Threat categories: cyber threats, e.g. ransomware, phishing attacks; insider threats, e.g. privilege abuse, data exfiltration; supply chain risk, e.g. third-party compromise, software supply chain attacks

Technology risks: cloud security risk, e.g. cloud misconfiguration, shared model risk; data breaches, e.g. unauthorised access, data exposure; system vulnerabilities, e.g. software flaws, patch weaknesses

Emerging risks: emerging technologies, e.g. AI systems, Internet of Things (IoT) devices; malware threats, e.g. ransomware, spyware; network security threats, e.g. network intrusion, denial-of-service.

*Risk assessment approaches:*

Assessment methods: qualitative risk assessment, e.g. risk matrices, expert judgement; quantitative risk assessment, e.g. annualised loss expectancy (ALE), statistical modelling; semi-quantitative models, e.g. weighted scoring models, numerical ranking scales

Analysis techniques: threat modelling, e.g. STRIDE modelling, attack trees; asset-based assessment, e.g. asset inventories, data classification registers; vulnerability scanning, e.g. Nessus scans, OpenVAS detection

Evaluation processes: risk scoring models, e.g. Common Vulnerability Scoring System (CVSS) scoring, risk prioritisation matrices; likelihood estimation, e.g. frequency analysis, probability scoring; impact evaluation, e.g. business impact analysis (BIA), financial impact modelling.

*Risk management theories and frameworks:*

Risk management stages: risk identification, e.g. asset-threat mapping, vulnerability identification; risk analysis, e.g. likelihood-impact analysis, quantitative risk calculation; risk evaluation, e.g. risk ranking, prioritisation scoring

Risk treatment approaches: risk treatment, e.g. control selection, mitigation planning; risk mitigation, e.g. patch management, access control implementation; risk transfer, e.g. cyber insurance, outsourcing risk

Risk response strategies: risk acceptance, e.g. documented acceptance decisions, residual risk approval; risk avoidance, e.g. system decommissioning, process redesign; control implementation, e.g. firewall deployment, encryption configuration.

LO3 **Explore the models for designing and implementing security controls to protect organisational information assets**

*Security control models:*

Control types: preventive controls, e.g. access control lists (ACLs), encryption enforcement; detective controls, e.g. intrusion detection systems (IDS), log analysis; corrective controls, e.g. patch deployment, incident remediation

Control functions: administrative controls, e.g. security policies, standard operating procedures; technical controls, e.g. endpoint protection platforms, system hardening; physical controls, e.g. biometric access systems, CCTV monitoring.

Access control types: discretionary access control, e.g. file permission ownership, ACL-based access; mandatory access control, e.g. classification labels, multilevel security (MLS); role-based access control, e.g. role assignment, least privilege enforcement

Identity and access management: identity management, e.g. Active Directory, identity federation; authentication mechanisms, e.g. multi-factor authentication (MFA), public key authentication; authorisation processes, e.g. role-based permission allocation, access approval workflows

Access governance: least privilege principle, e.g. minimal role permissions, privilege restriction; access policies, e.g. identity-based access policies, conditional access rules; access auditing, e.g. access logs, periodic access reviews.

*Technical security mechanisms:*

Cryptography and protocols: encryption techniques, e.g. AES-256 encryption, Rivest–Shamir–Adleman (RSA) encryption; cryptographic algorithms, e.g. SHA-256 hashing, elliptic curve cryptography (ECC); authentication protocols, e.g. OAuth 2.0, Kerberos

Network and system protection: network security controls, e.g. firewalls, network segmentation; IDS, e.g. signature-based IDS, anomaly-based IDS; endpoint protection, e.g. endpoint detection and response (EDR), antivirus solutions

Monitoring and vulnerability management: security monitoring systems, e.g. SIEM platforms, real-time alerting; vulnerability management, e.g. vulnerability scanning tools, patch prioritisation; malware protection, e.g. sandboxing, behavioural analysis.

*Security architecture and policies:*

Architecture models: defence-in-depth, e.g. layered perimeter and endpoint controls, multiple control tiers; layered security architecture, e.g. network, application, data security layers; zero-trust architecture, e.g. continuous authentication, device trust validation

Policy and governance: security policy frameworks, e.g. ISO 27001 policy sets, internal governance policies; procedural controls, e.g. incident response procedures, change management processes; policy enforcement, e.g. automated compliance enforcement, configuration management

Integration and life cycle management: security standards alignment, e.g. ISO/NIST control mapping, compliance alignment; risk-based architecture, e.g. threat-driven design, risk prioritisation; security life cycle management, e.g. secure development life cycle (SDL), patch life cycle management.

LO4 **Present a regulatory compliant information management strategy that meets an identified organisational need to a range of stakeholders**

*Regulatory and legal requirements:*

Legal frameworks: data protection legislation, e.g. Data Protection Act 2018, UK GDPR; regulatory compliance obligations, e.g. NIS Regulations 2018, Privacy and Electronic Communications Regulations

Regulatory standards: sector-specific regulations, e.g. Payment Card Industry Data Security Standard (PCI DSS), NHS Data Security and Protection (DSP) Toolkit; international standards, e.g. ISO/IEC 27001, ISO/IEC 27002; GDPR requirements, e.g. data subject rights, lawful processing principles.

*Information management strategy components:*

Strategic planning: security governance, e.g. ISO/IEC 27001 governance, National Institute of Standards and Technology (NIST) governance functions; incident response planning, e.g. NIST SP 800-61, ISO/IEC 27035; business continuity planning, e.g. ISO 22301, continuity frameworks

Resilience management: disaster recovery planning, e.g. ISO/IEC 27031, recovery frameworks; organisational resilience, e.g. ISO 22316, resilience frameworks

Resource and control management: risk management integration, e.g. ISO 31000, NIST risk management framework; control frameworks, e.g. ISO/IEC 27002 controls, CIS Controls.

*Incident and response management:*

Incident life cycle: incident response life cycle, e.g. NIST incident life cycle, ISO/IEC 27035 life cycle; threat detection, e.g. SIEM systems, intrusion detection systems; incident containment, e.g. containment strategies, isolation procedures

Response execution: threat eradication, e.g. malware removal, system cleansing; system recovery, e.g. service restoration, disaster recovery; corrective actions, e.g. remediation actions, control improvements

Investigation and reporting: post-incident review, e.g. lessons learned, review processes; incident reporting, e.g. regulatory reporting, breach notification; forensic analysis, e.g. digital forensics, evidence analysis.

*Stakeholder considerations:*

Stakeholder groups: internal stakeholders, e.g. employees, management; external stakeholders, e.g. clients, partners; regulatory bodies, e.g. Information Commissioner's Office, National Cyber Security Centre (NCSC)

Stakeholder management: assurance requirements, e.g. ISO assurance requirements, compliance assurance; communication strategies, e.g. reporting frameworks, communication plans

Governance and accountability: organisational impact, e.g. business impact, operational impact; accountability structures, e.g. reporting structures, governance frameworks.

*Presentation techniques:*

Presentation design: presentation structure, e.g. logical structure, content organisation; audience adaptation, e.g. stakeholder requirements, audience needs; visual communication, e.g. dashboards, graphical reporting

Delivery techniques: data visualisation, e.g. charts, dashboards; presentation timing, e.g. time management, delivery pacing.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Discuss the core principles of information security management</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Critically evaluate the effectiveness of information security principles, risk management strategies and security control models in securing digital environments and protecting digital assets.</td></tr>
<tr><td>P1 Discuss, using a selection of real-world examples, the core principles of information security management including relevant governance and frameworks.<br><br>P2 Explore, using a selection of real-world examples, how an effective security culture can contribute to the management of information security within organisational contexts.</td><td>M1 Analyse, using a selection of real-world examples, how information security management principles are applied within organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Review the organisational risk assessment and risk management theories used in securing digital environments</td></tr>
<tr><td>P3 Review, using a selection of real-world examples, the concepts of risks to digital environments in the context of identified digital threats.<br><br>P4 Discuss, using a selection of real-world examples, the approaches to risk assessment and management for a range of organisational contexts.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of applying risk assessment and management strategies within organisational contexts.</td></tr>
<tr><td colspan="2">LO3 Explore the models for designing and implementing security controls to protect organisational information assets</td></tr>
<tr><td>P5 Explore, using a selection of real-world examples, the physical and digital security models used to design and implement security controls for protecting organisational information assets.<br><br>P6 Review, using a selection of real-world examples, the architectural and governance security strategies used to design and implement security controls for protecting organisational information assets.</td><td>M3 Analyse, using a selection of real-world examples, the effectiveness of security control models in protecting organisational information assets.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Present a regulatory compliant information management strategy that meets an identified organisational need to a range of stakeholders | | |
| **P7** Present a regulatory-compliant information management strategy that meets an identified organisational need to a range of stakeholders to obtain feedback.<br><br>**P8** Review the stakeholder feedback to identify strengths and limitations of the regulatory compliant information management strategy in meeting identified organisational needs. | **M4** Evaluate, using the stakeholder feedback, the effectiveness of the information management strategy in meeting organisational and stakeholder needs. | **D2** Justify the decisions made in the development of the regulatory compliant information management strategy that allows it to effectively meet the requirements of an identified scenario. |

## Recommended Resources

### Textbooks

Calder, A. and Watkins, S. (2024) *IT Governance: An International Guide to Data Security and ISO 27001/ISO 27002*. 8th edn. Ely: IT Governance Publishing.

Green, J. *et al*. (2024) *Information Security Management Principles*. 4th edn. Swindon: BCS, The Chartered Institute for IT.

Hopkin, P. (2024) *Fundamentals of Risk Management: Understanding, Evaluating and Implementing Effective Risk Management*. 7th edn. London: Kogan Page.

Maleh, Y. *et al*. (2021) *IT Governance and Information Security: Guides, Standards, and Frameworks*. Boca Raton, FL: CRC Press.

Merkow, M.S. and Breithaupt, J. (2023) *Information Security: Principles and Practices*. 2nd edn. Hoboken, NJ: Wiley.

Peltier, T.R. (2023) *Information Security Policies, Procedures, and Standards*. 2nd edn. Boca Raton, FL: CRC Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/security/a-guide-to-data-security/ | Information Commissioner's Office (ICO)<br>Guide to data security principles and measures including confidentiality, integrity and availability<br>(General reference) |
| devguide.owasp.org/en/05-implementation/01-documentation/01-proactive-controls/ | OWASP Developer Guide<br>Overview of proactive security controls such as access control, cryptography and secure configuration<br>(General reference) |
| www.standardfusion.com/blog/understanding-the-relationship-between-nist-csf-and-iso-27001 | StandardFusion<br>Explanation of ISO 27001 and NIST CSF security frameworks and governance structures<br>(Article) |

| | |
|---|---|
| www.ncsc.gov.uk/collection/risk-management | National Cyber Security Centre<br>Guidance on identifying, assessing and managing cyber security risks within organisations<br>(General reference) |
| www.gov.uk/government/publications/secure-connected-places-playbook-documents/conducting-a-stride-based-threat-analysis | UK Government (DSIT)<br>STRIDE-based threat modelling approach for identifying risks and vulnerabilities in systems<br>(Tutorial) |
| drata.com/learn/risk/stride-threat-model | Drata Trust Academy<br>Tutorial explaining STRIDE threat modelling for analysing threats and applying mitigation controls<br>(Tutorial) |
| learn.microsoft.com/en-us/training/modules/tm-use-a-framework-to-identify-threats-and-find-ways-to-reduce-or-eliminate-risk/ | Microsoft Learn<br>Interactive module on threat modelling frameworks and applying security controls to reduce risk<br>(Tutorial) |
| securecodingpractices.com/stride-threat-modeling-example/ | Secure Coding Practices<br>Practical examples of STRIDE threat modelling and how it supports risk identification and mitigation<br>(Article) |
| www.tunedsecurity.com/owasp-top-10-a-comprehensive-guide-to-web-application-security/ | Tuned Security<br>Detailed explanation of OWASP Top 10 security risks and mitigation strategies<br>(Article) |
| techternet.com/post/understanding-nist-iso-27001-cis-cybersecurity-frameworks/ | Techternet<br>Guide to Cybersecurity Frameworks including NIST, ISO 27001 and CIS controls and their integration<br>(Article) |

**Journals and articles**

Kamil, Y., Lund, S. and Islam, M.S. (2023) 'Information security objectives and the output legitimacy of ISO/IEC 27001: stakeholders' perspective on expectations in private organizations in Sweden', *Information Systems and e-Business Management*, 21, pp. 699–722. Available at: doi.org/10.1007/s10257-023-00646-y.

Ojo, C. *et al*. (2024) 'Incident response: a structured model from detection to containment and recovery', *World Journal of Advanced Research and Reviews*, 24(1), pp. 1401–1407. Available at: doi.org/10.30574/wjarr.2024.24.1.3148.

Putra, A.P. and Soewito, B. (2023) 'Integrated Methodology for Information Security Risk Management using ISO 27005:2018 and NIST SP 800-30 for Insurance Sector', *International Journal of Advanced Computer Science and Applications*, 14(4). Available at: dx.doi.org/10.14569/IJACSA.2023.0140468.

Woods, D.W. and Seymour, S. (2024) 'Evidence-based cybersecurity policy: A meta-review of security control effectiveness', *Journal of Cyber Policy*, 8(3), pp. 365–383. Available at: doi.org/10.1080/23738871.2024.2335461.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*.

# Unit 531: E-Commerce and Strategy

**Unit code:** **A/652/3317**

**Unit level:** **5**

**Credit value:** **15**

---

## Introduction

E-commerce is no longer a standalone sales channel but a core component of organisational strategy, embedded within complex digital ecosystems shaped by platforms, data and evolving technologies. Organisations now operate in environments where customer interactions span multiple channels, decisions are increasingly data-driven, and competitive advantage is influenced by the ability to integrate technology, insight and experience at scale.

This unit draws on principles from digital business, e-commerce strategy and data-informed decision-making. It is designed for students preparing for roles such as E-Commerce Analyst, Digital Strategist, Customer Experience Specialist or Digital Transformation Practitioner, where understanding how digital environments function and how strategic decisions are made is essential.

The unit develops a strategic understanding of how organisations design and manage e-commerce activities within platform-based markets. Students will explore e-commerce models and strategic frameworks, review the technological, regulatory and security factors that shape digital operations, and discuss how data-driven and AI-enabled techniques are used to optimise customer experience and organisational performance. Students will apply their knowledge to develop and present a data-informed and scalable e-commerce solution, considering organisational context, stakeholder requirements, technological constraints and performance objectives.

On completion of this unit, students will be able to explore complex digital ecosystems, review risks and opportunities, discuss strategic options, and present decisions using evidence-based reasoning. These capabilities are essential for effective practice in contemporary digital business environments and support progression into employment or further study in areas such as digital strategy, marketing analytics and technology management.

 © Pearson Education Limited 2026

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore core e-commerce models and strategic frameworks

LO2 Review technological, legal and security factors shaping e-commerce ecosystems

LO3 Discuss data-driven techniques for optimising online customer experience

LO4 Present a strategic plan for implementing a robust and scalable e-commerce solution for an identified scenario to a range of key stakeholders.

## Essential Content

### LO1 Explore core e-commerce models and strategic frameworks

*E-commerce models and digital ecosystems:*

E.g. business-to-business (B2B), business-to-consumer (B2C), consumer-to-consumer (C2C), direct-to-consumer (D2C), subscription commerce, marketplace and platform-based models, and social commerce

Omni-channel and unified commerce strategies, including integration across digital and physical touchpoints

Overview of intermediary roles, fulfilment ecosystems and platform dependencies.

*Revenue models and value creation:*

E.g. transaction-based, subscription, advertising, affiliate and data-driven revenue models

Dynamic pricing, bundling strategies and value-based pricing

Development of customer value propositions through personalisation, convenience, trust and experience-led differentiation.

*Strategic frameworks for competitive positioning:*

Application of digital strategy frameworks including value chain analysis, ecosystem strategy, platform competition and capability alignment

Consideration of network effects, switching costs, scalability and digital competitive advantage.

*Contemporary strategic challenges:*

E.g. Platform versus direct-to-consumer strategies, global versus local market positioning, sustainability in digital operations, and ethical considerations in data-driven commerce

Trade-offs between control, reach, cost and customer ownership.

### LO2 Review technological, legal and security factors shaping e-commerce ecosystems

*Technology architecture and regulatory frameworks:*

Front-end and backend systems, APIs, microservices architecture, headless and composable commerce, cloud-based infrastructure, and third-party integrations.

Consideration of scalability, resilience and performance

Data protection, consumer rights, digital contracts, cross-border trading, taxation, accessibility requirements and platform governance

Organisational responsibilities for compliance and accountability

Current awareness, developments, evolving practice.

*Cybersecurity, risk and AI:*

E.g. Authentication, encryption, identity and access management, fraud prevention, secure payment systems, and incident response

Organisational approaches to risk governance and resilience. Understanding impact of cyber security and AI-associated risk

Use of AI in e-commerce, e.g. recommendation engines, chatbots and pricing systems

Ethical considerations including bias, transparency, accountability and responsible AI use.

*Organisational implications:*

Impact of technological capabilities, regulatory constraints and security risks on scalability, operational efficiency, profitability and decision-making in e-commerce environments.

LO3 **Discuss data-driven techniques for optimising online customer experience**

*Customer insight and behavioural analytics:*

Customer journey mapping, behavioural tracking, funnel analysis, cohort analysis and attribution modelling. Integration of first-party and third-party data.

*Optimisation and experimentation:*

E.g. A/B testing, multivariate testing, experimentation frameworks, statistical validity and optimisation strategies across digital touchpoints.

*Acquisition and optimisation tools:*

E.g. search engine optimisation (SEO), search engine marketing (SEM), paid media, social commerce, influencer marketing and content-driven strategies. Alignment between acquisition channels and experience

Recommendation engines, real-time personalisation, customer lifetime value modelling, churn prediction and behavioural forecasting

Use of dashboards, KPIs and analytics tools to evaluate performance, inform decisions and support continuous improvement.

*Discussion of application:*

Consideration of how organisations apply data-driven techniques to enhance customer engagement, improve user experience and optimise performance outcomes.

LO4 **Present a strategic plan for implementing a robust and scalable e-commerce solution for an identified scenario to a range of key stakeholders**

*Scenario and stakeholder analysis:*

Organisational objectives, stakeholder requirements, operational constraints and performance priorities.

*Solution design, scalability and implementation planning:*

Architectural approaches including microservices, headless commerce and cloud solutions. Consideration of scalability, resilience and integration

Roadmaps, sequencing, governance structures, change management and organisational readiness.

*Commercial decision-making and performance sustainability:*

E.g. Cost–benefit analysis, return on investment (ROI) modelling, investment decisions, risk mitigation and trade-off analysis

Key performance indicators (KPI) development, monitoring frameworks, continuous improvement and long-term sustainability.

*Presentation and stakeholder communication:*

Methods for presenting strategic plans to stakeholders, e.g. communication approaches, use of data visualisation and digital tools, and responding to stakeholder feedback and questions.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explore core e-commerce models and strategic frameworks | | |
| **P1** Explore different e-commerce models used by organisations.<br>**P2** Describe strategic frameworks applied in digital market environments. | **M1** Assess how e-commerce models and strategic frameworks support organisational objectives. | **D1** Critically analyse the suitability of e-commerce models and frameworks in digital market ecosystems. |
| **LO2** Review technological, legal and security factors shaping e-commerce ecosystems | | |
| **P3** Review technological, legal and security factors in e-commerce environments.<br>**P4** Describe how these factors affect organisational operations and compliance. | **M2** Analyse how technological, legal and security factors influence e-commerce practices and decision-making. | **D2** Evaluate how technological, legal and security factors influence strategic decision-making and scalability in e-commerce ecosystems. |
| **LO3** Discuss data-driven techniques for optimising online customer experience | | |
| **P5** Outline data-driven techniques used to optimise online customer experience.<br>**P6** Discuss how data-driven and AI-enabled tools support customer interactions. | **M3** Analyse how data-driven and AI-enabled techniques improve customer experience, personalisation and engagement. | **D3** Evaluate the effectiveness of data-driven and AI-enabled techniques in achieving customer engagement and conversion outcomes. |
| **LO4** Present a strategic plan for implementing a robust and scalable e-commerce solution for an identified scenario to a range of key stakeholders | | |
| **P7** Create a strategic e-commerce plan, with recommendations, for implementing an e-commerce solution in an organisational context.<br>**P8** Present a strategic e-commerce plan to stakeholders. | **M4** Assess how the plan addresses organisational objectives and constraints.<br>**M5** Appraise the suitability of the strategic plan in relation to scalability, performance and stakeholder requirements. | **D4** Justify strategic decisions within the plan in relation to organisational goals, data insights, risks and expected outcomes. |

## Recommended Resources

### Textbooks

Chaffey, D., Hemphill, T. and Edmundson-Bird, D. (2024) *Digital Business and E-commerce*. 8th edn. Harlow: Pearson Education.

Kingsnorth, S. (2026) *The Digital Marketing Handbook: A Step-by-Step Guide to Delivering Consistent Digital Marketing Success*. 2nd edn. London: Kogan Page.

Ryan, D. (2024) *Understanding Digital Marketing: Marketing Strategies for Engaging the Digital Generation*. 6th edn. London: Kogan Page.

Wirtz, B.W. (2024) *Digital Business and Electronic Commerce: Strategy, Business Models and Technology*. Cham: Springer.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.mdpi.com | MDPI<br>Electronic commerce research and open-access journals<br>(Research) |
| www.nist.gov | National Institute of Standards and Technology<br>(General reference) |
| www.springer.com | Springer – Digital business and e-commerce research<br>(Research) |
| www.statista.com | Statista<br>E-commerce market and consumer data<br>(Datasets) |
| www.tandfonline.com | Taylor & Francis<br>Digital commerce and customer experience research<br>(Research) |
| www.wiley.com | Wiley<br>Big data analytics and digital marketing research<br>(Research) |

 © Pearson Education Limited 2026

**Journals and articles**

Guntakandla, A.R. (2025) 'Microservices Architecture: Decomposing E-Commerce Monoliths into Scalable, Independent Services', *Journal of Information Systems Engineering and Management*, 10(58s).
Available at: doi.org/10.52783/jisem.v10i58s.12665.

Kostopoulos, G. *et al*. (2026) 'Deep Learning for e-Commerce: Recent Developments in Prediction, Personalization and Decision Intelligence', *Applied Sciences*, 16(5).
Available at: doi.org/10.3390/app16052263.

Wang, S. *et al*. (2025) 'Security risks and regulation of cross-border e-commerce in the digital economy', *Journal of Electronic Business & Digital Economics*, 4(2), pp. 385–401.
Available at: doi.org/10.1108/JEBDE-12-2024-0053.

Wang, T.-C. *et al*. (2025) 'Multi-Stage Data-Driven Framework for Customer Journey Optimization and Operational Resilience', *Mathematics*, 13(7), 1145.
Available at: doi.org/10.3390/math13071145.

**Links**

This unit links to the following related units:

*Unit 411: Core Data Analytics*

*Unit 420: Core AI*

*Unit 430: Emerging Technologies*

*Unit 503: Risk Assessment and Change Management*

*Unit 512: Advanced Data Analytics*

*Unit 515: Secure System Design*

*Unit 527: System Analysis and Design.*

# Unit 532: Prototyping with AI

**Unit code:** D/652/3318

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Prototyping plays a central role in modern digital development, helping organisations shape ideas, test assumptions and refine solutions before full production begins. As artificial intelligence becomes embedded in design and engineering practices, professionals increasingly need to understand how traditional prototyping methods are evolving. This unit sits within the wider computing and digital innovation field and is relevant to roles where problem-solving, iteration and technical judgement are essential.

The aim of this unit is to give students a clear understanding of prototyping within contemporary, AI-enabled environments. It introduces the principles that guide effective prototype creation and highlights where AI tools can add value, particularly in speeding up routine tasks, generating alternatives and supporting decision-making. Students will learn how both conventional and AI-assisted approaches contribute to the development of reliable, well-reasoned design outcomes.

The unit covers key aspects of prototyping practice. Students will explore core concepts such as fidelity, iteration and validation, and examine how AI-based systems can be integrated into design workflows. They will develop several prototype iterations for a chosen scenario using both manual and AI-supported methods, before comparing their effectiveness using appropriate evaluation criteria. This provides a practical route for understanding how different approaches influence the quality and suitability of a proposed solution.

On successful completion of this unit, students will be able to create, assess and compare prototype designs with confidence, drawing on both conventional techniques and emerging AI tools. These skills are valuable across a wide range of roles, including Digital Product Designer, UX/UI Designer, Software Developer, AI Prototyping Specialist, Systems Analyst and Technical Consultant. The experience gained will support progression into areas involving digital solution design, automation-enhanced workflows and applied artificial intelligence.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core principles of prototyping

LO2 Explore how AI tools are integrated into prototype design workflows

LO3 Develop several prototype iterations for an identified scenario using both conventional and AI methodologies

LO4 Compare the effectiveness of the conventional and AI-assisted prototypes developed for the identified scenario.

## Essential Content

### LO1 **Explain the core principles of prototyping**

*Prototyping purpose and role:*

Purpose: idea exploration e.g. concept sketches, wireframes; requirement clarification e.g. feature lists, acceptance criteria; stakeholder communication e.g. mock-ups, walkthrough diagrams

Evaluation: feasibility assessment e.g. constraint mapping, dependency checks; risk reduction e.g. assumption checks, early testing probes; design validation e.g. usability trials, prototype walkthroughs

Alignment: user needs e.g. persona mapping, scenario analysis; business needs e.g. value checks, organisational alignment.

*Prototyping life cycle:*

Stages: ideation e.g. brainstorming outputs, concept notes; low-fidelity prototyping e.g. sketches, block models; iterative refinement e.g. revision cycles, feedback updates

Process: feedback incorporation e.g. review notes, usability comments; validation stages e.g. requirement confirmation, scenario testing; transition to development e.g. annotated screens, handoff assets

Models: iterative development models e.g. rapid loops, Plan, Do, Check, Act (PDCA) stages; incremental development models e.g. staged builds, feature increments.

*Prototype fidelity levels:*

Low fidelity: low-fidelity representations e.g. sketches, wireframes

Medium fidelity: medium-fidelity functional models e.g. clickable prototypes, mid-detail screens

High fidelity: high-fidelity artefacts e.g. simulations, near-production screens

Trade-offs: speed e.g. quick revisions, fast creation; accuracy e.g. interaction realism, layout precision; development effort e.g. complexity demands, time requirements.

*Concept validation principles:*

Validation: assumption testing e.g. hypothesis checks, reasoning tests; user feedback relevance e.g. task reviews, interview notes; evidence-based decision-making e.g. observation notes, test data

Scope: validation scope e.g. boundaries, focus limits; prototype limitations e.g. missing functions, simplified flows; prototype constraints e.g. technical limits, resource limits

Suitability: prototype detail e.g. specificity level, representation depth; early artefact suitability e.g. scenario fit, requirement alignment.

*Examples of real-world AI prototyping applications:*

A range of real-world AI prototyping tools e.g.

Prototyping and design tools: Figma AI e.g. wireframes, interface design; Adobe Firefly e.g. generative design, visual assets; Uizard e.g. user interface (UI) generation, design automation

Collaboration and ideation tools: e.g. brainstorming, idea mapping; Miro AI e.g. collaboration boards, concept planning; FigJam AI e.g. workshop facilitation, design ideation

High-fidelity prototyping tools: Figma Interactive Prototypes e.g. clickable interfaces, screen simulation; Framer AI e.g. interactive design, layout generation; Webflow AI e.g. web prototyping, responsive design

Validation and testing tools: Maze AI e.g. usability testing, feedback analysis; UserTesting AI e.g. user insights, behaviour tracking; Hotjar AI e.g. user analytics, interaction tracking

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO2 **Explore how AI tools are integrated into prototype design workflows**

*AI-supported design workflows:*

AI-assisted ideation: concept suggestions e.g. design prompts, layout alternatives; idea expansion e.g. variation proposals, pattern insights

Workflow augmentation: automated layout suggestions e.g. structure proposals, spacing adjustments; data-driven insights e.g. feature insights, trend-based refinements.

*Automation and optimisation mechanisms:*

Process automation: repetitive task reduction e.g. auto-formatting, auto-naming; template generation e.g. structured layouts, predefined screens

Optimisation mechanisms: layout optimisation e.g. spacing refinement, balance adjustment; adaptive updates e.g. interface modifications, structural improvements

Pattern recognition: pattern detection e.g. interface patterns, flow trends; structural recognition e.g. component relationships, layout structures.

*AI tool integration considerations:*

Interoperability: tool compatibility e.g. shared formats, plugin workflows; workflow alignment e.g. model integration, asset exchange

Data quality: data requirements e.g. structured sets, labelled data; data limitations e.g. incomplete inputs, noisy examples

Model limitations: explainability needs e.g. transparent outputs, justification cues; dependency risks e.g. over-reliance risks, failure-point risks

Ethical considerations: governance factors e.g. approval procedures, compliance expectations; transparency expectations e.g. decision clarity, workflow clarity.

*Conventional versus AI-enhanced workflows:*

Human-led decision-making: manual judgement e.g. design evaluation, refinement decisions; oversight responsibility e.g. reviewing outputs, ensuring suitability

AI-augmented support: assisted generation e.g. draft screens, layout variations; rapid iteration e.g. fast updates, quick alternatives

Workflow differences: speed differences e.g. quicker cycles, faster adjustments; flexibility differences e.g. editable drafts, adaptive suggestions

*Examples of real-world AI prototyping applications:*

A range of real-world AI prototyping tools e.g.

AI design tools: Figma AI e.g. layout generation, design suggestions; Adobe Firefly e.g. visual generation, design assets; Uizard e.g. UI creation, rapid prototyping

Automation tools: Framer AI e.g. layout automation, interface generation; Webflow AI e.g. web design automation, structure generation; Builder.io e.g. template generation, workflow automation

Integration tools: Figma Plugins e.g. tool extensions, workflow integration; Adobe Creative Cloud e.g. asset integration, cross-tool workflows; GitHub Copilot Workspace e.g. design-development integration, workflow support

AI prototyping tools: Galileo AI e.g. UI generation, layout creation; Locofy AI e.g. design-to-code conversion, workflow generation; Anima AI e.g. design-to-development, interactive prototypes

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

LO3 **Develop several prototype iterations for an identified scenario using both conventional and AI methodologies**

*Prototype iteration characteristics:*

Design versions: versions e.g. draft screens, early layouts; variations e.g. design alternatives, workflow changes; refinement cycles e.g. iteration rounds, update passes

Evaluation basis: feedback e.g. peer notes, review comments; comparison e.g. prototype differences, scenario contrasts; responsiveness e.g. adjustment actions, response decisions.

*Conventional prototyping approaches:*

Manual methods: manual design techniques e.g. sketching, hand-drawn flows; rule-based systems e.g. defined steps, structured processes; deterministic workflows e.g. linear sequences, fixed logic

Tools: development tools e.g. interface editors, layout tools; established tools e.g. design software, modelling applications; predictable behaviour e.g. consistent outputs, stable responses.

*AI-assisted prototyping approaches:*

Generative systems: generative models e.g. screen generation, layout creation; adaptive systems e.g. behaviour adaptation, pattern adjustments; data-driven variation e.g. option generation, alternative layouts

Automation outputs: automated configuration e.g. auto-arranged screens, property adjustments; model-influenced outputs e.g. AI-shaped layouts, suggested components.

*Scenario alignment factors:*

Context and requirements: constraints e.g. time limits, resource boundaries; functional requirements e.g. required features, core functions; performance expectations e.g. response speed, stability needs

Usability and feasibility: usability considerations e.g. clarity checks, navigation ease; scalability relevance e.g. growth readiness, capacity considerations; implementation feasibility e.g. build suitability, integration potential

Effectiveness: effectiveness criteria e.g. accuracy needs, suitability checks; trade-offs e.g. speed trade-offs, complexity trade-offs.

*Examples of real-world AI prototyping applications:*

A range of real-world AI prototyping tools e.g.

Prototype iteration tools: Figma Version History e.g. version tracking, design comparison; Framer Iterations e.g. design updates, refinement cycles; Adobe XD Prototype Versions e.g. version control, iteration management

Conventional prototyping tools: Figma Manual Prototyping e.g. manual design, interface creation; Axure RP e.g. structured prototyping, interaction design; Balsamiq e.g. wireframing, low-fidelity design

AI prototyping tools: Galileo AI e.g. UI generation, layout creation; Uizard Autodesigner e.g. automated design, rapid prototyping; Figma AI Generate e.g. layout generation, component suggestions

(AI tools and platforms evolve rapidly. The examples provided are illustrative and may be replaced with more appropriate or current alternatives).

*Project and requirement tools:*

Jira Product Discovery e.g. requirement tracking, feature planning; Notion e.g. documentation, requirement management; Azure DevOps Boards e.g. task tracking, workflow management.

LO4 **Compare the effectiveness of the conventional and AI-assisted prototypes developed for the identified scenario**

*Prototype evaluation criteria:*

Performance indicators: performance indicators e.g. response speed, output accuracy; efficiency considerations e.g. task effort, resource use; reliability indicators e.g. error frequency, stability level

Usability measures: usability measures e.g. navigation ease, clarity of layout; user experience factors e.g. interaction smoothness, feedback clarity.

*Testing and evaluation frameworks:*

Comparative analysis: comparative methods e.g. side-by-side reviews, scenario comparisons; evaluation conditions e.g. consistent tasks, controlled tests

Measurement approaches: qualitative measures e.g. user comments, observation notes; quantitative measures e.g. timing data, success rates.

*Scalability and constraint considerations:*

Resource demands: resource demands e.g. processing load, memory usage; computational overhead e.g. model cost, runtime weight

Adaptability: adaptability needs e.g. scaling capacity, component expansion; deployment constraints e.g. platform limits, integration barriers.

*Effectiveness comparison factors:*

Strengths and weaknesses: strengths e.g. speed advantages, clarity improvements; weaknesses e.g. complexity barriers, inconsistency risks

Scenario suitability: suitability for requirements e.g. requirement fit, scenario alignment; control vs automation: trade-offs e.g. control trade-offs, automation trade-offs.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the core principles of prototyping</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate the effectiveness of core prototyping principles and AI integrated design workflows in improving the quality and reliability of prototype outcomes.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the purpose, role and life cycle of prototyping.<br><br>P2 Discuss, using a selection of real-world examples, the fidelity levels and validation principles used in the development of prototypes for a range of applications.</td><td>M1 Assess, using a selection of real-world examples, how prototyping stages and fidelity levels support effective concept validation.</td></tr>
<tr><td colspan="2">LO2 Explore how AI tools are integrated into prototype design workflows</td></tr>
<tr><td>P3 Explore, using a selection of real-world examples, how AI tools are integrated into prototype design workflows to support automation and optimisation.<br><br>P4 Review, using real-world examples, the considerations involved in integrating AI tools into prototype workflows and their influence on the differences between conventional and AI-enhanced approaches.</td><td>M2 Analyse, using a selection of real-world examples, the impact of AI tool integration on the efficiency and effectiveness of prototype design workflows.</td></tr>
</table>

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO3 Develop several prototype iterations for an identified scenario using both conventional and AI methodologies</td><td rowspan="5">LO3 and LO4<br><br>D2 Justify the selection of either a conventional or AI-assisted prototype as the most appropriate solution for the identified scenario, using evidence from the development process, review activities and stakeholder feedback.</td></tr>
<tr><td>P5 Develop several prototype iterations for an identified scenario using conventional methodologies.<br><br>P6 Develop several prototype iterations for an identified scenario using AI-assisted methodologies.</td><td>M3 Evaluate the suitability of the prototype iterations, developed by conventional and AI-assisted methodologies, in addressing the requirements of the identified scenario.</td></tr>
<tr><td colspan="2">LO4 Compare the effectiveness of the conventional and AI-assisted prototypes developed for the identified scenario</td></tr>
<tr><td>P7 Review the effectiveness of the conventional and AI-assisted prototypes developed for the identified scenario using appropriate criteria.<br><br>P8 Discuss the effectiveness of the conventional and AI-assisted prototypes developed for the identified scenario with a range of relevant stakeholders to obtain feedback.</td><td>M4 Optimise, using the results of the review and the stakeholder feedback, the conventional and AI-assisted prototypes developed to meet the requirements of the identified scenario.</td></tr>
</table>

## Recommended Resources

### Textbooks

Bohn, J. and Short, J. (2023) *Human-Centred AI: Designing for Collaboration Between People and Machines.* Cambridge, MA: MIT Press.

Knapp, J., Zeratsky, J. and Kowitz, B. (2024) *Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days*. Revised edn. London: Bantam.

Li, Y. and Hilliges, O. (eds.) (2021) *Artificial Intelligence for Human-Computer Interaction: A Modern Approach*. Springer.

Martin, B. and Hanington, B. (2024) *Universal Methods of Design: Expanded and Revised.* 3rd edn. Beverly, MA: Rockport Publishers.

Pennefather, P.P. (2023) *Creative Prototyping with Generative AI: Augmenting Creative Workflows with Generative AI*. Berkeley, CA: Apress.

Shneiderman, B. (2022) *Human-Centered AI*. New York: Oxford University Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| learn.microsoft.com/en-us/windows/win32/appuistart/prototyping-a-user-interface | Microsoft Learn<br>Prototyping a User Interface<br>(Article) |
| productschool.com/blog/artificial-intelligence/ai-prototyping | Product School<br>AI Prototyping: From Sketch to App in Hours<br>(Article) |
| www.smashingmagazine.com/2023/03/ai-technology-transform-design/ | Smashing Magazine<br>How AI Technology Will Transform Design<br>(Article) |
| martinfowler.com/articles/engineering-practices-llm.html | martinfowler.com<br>Engineering Practices for LLM Application Development<br>(Article) |
| help.figma.com/hc/en-us/articles/360040314193-Guide-to-prototyping-in-Figma | Figma Help<br>Guide to Prototyping in Figma<br>(Tutorial) |

 © Pearson Education Limited 2026

| | |
|---|---|
| help.figma.com/hc/en-us/articles/17146044893591-Advanced-prototyping-examples | Figma Learn<br>Advanced Prototyping Examples<br>(Tutorial) |
| www.builder.io/blog/ai-prototyping-guide | Builder.io<br>Practical Guide to AI Prototyping<br>(Article) |
| www.ibm.com/training/enterprise-design-thinking/toolkit | IBM<br>Enterprise Design Thinking Toolkit<br>(Reference) |
| decode.mit.edu/assets/papers/Sketch2Prototype.pdf | Sketch2Prototype (MIT)<br>Generative AI Pipeline: Sketch-to-Text<br>(PDF) |
| www.erikvanek.com/is-coding-dead/ | Erik Vanek<br>LLM-Assisted Prototyping Case Study<br>(Blog) |

**Journals and articles**

Choudhury, M.M., Eisenbart, B. and Kuys, B. (2025) 'Artificial intelligence in the design process - A review and analysis on generative AI perspectives', *Proceedings of the Design Society*, 5, pp. 631–640. Available at: doi.org/10.1017/pds.2025.10077.

Daareyni, A. *et al*. (2025) 'Generative AI meets CAD: Enhancing engineering design to manufacturing processes with large language models', *International Journal of Advanced Manufacturing Technology*. Available at: doi.org/10.1007/s00170-025-15830-2.

Saadi, J.I. and Yang, M.C. (2023) 'Generative Design: Reframing the Role of the Designer in Early-Stage Design Process', *Journal of Mechanical Design*, 145(4). Available at: doi.org/10.1115/1.4056799.

Wang, N.W. *et al*. (2025) 'Exploring creativity in human–AI co-creation: A comparative study across design experience', *Frontiers in Computer Science*, 7. Available at: doi.org/10.3389/fcomp.2025.1672735.

**Links**

This unit links to the following related units:

*Unit 520: Applied AI*

*Unit 521: Generative and Conversational AI.*

# Unit 533: Transport Layer Engineering

**Unit code:** F/652/3319

**Unit level:** 5

**Credit value:** 15

---

## Introduction

The transport layer plays a central role in modern computer networks, enabling communication between applications across complex and distributed environments. From cloud platforms and streaming services to real-time communication systems, transport layer technologies ensure that data is delivered efficiently, reliably and in a form that can be understood by end applications. As networks continue to evolve in scale and complexity, an understanding of transport layer behaviour is essential for those working in technical networking and systems roles.

The aim of this unit is to develop a clear and applied understanding of how transport layer protocols and mechanisms support end-to-end communication. Students will explore how data is segmented, transmitted and managed across networks, alongside the principles that govern reliability, performance and control. The unit also introduces the key strategies used to manage data flow and congestion, supporting informed decision-making when working with transport layer technologies in practical scenarios.

The unit covers a range of topics that reflect both theoretical and applied aspects of transport layer engineering. These include the architecture of the transport layer within the Open Systems Interconnection (OSI) and Transmission Control Protocol/Internet Protocol (TCP/IP) models, the behaviour of reliable and unreliable transmission mechanisms, and the operation of flow and congestion control approaches. Students will also examine configuration parameters, monitoring tools and performance optimisation techniques, applying these to realistic scenarios to evaluate network behaviour and outcomes.

On successful completion of this unit, students will be able to explain transport layer principles, analyse data transmission behaviours, and apply configuration techniques to improve network performance in defined contexts. These skills support progression into technical roles such as Network Engineer, Network Technician, Systems Administrator, Network Operations Analyst and Infrastructure Engineer, where an understanding of transport layer performance and optimisation is a key requirement.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the principles of transport layer architecture

LO2 Discuss the mechanisms for reliable and unreliable data transmission

LO3 Review transport layer flow and congestion control algorithms

LO4 Implement transport layer configuration techniques to optimise network performance for an identified scenario.

## Essential Content

LO1 **Explain the principles of transport layer architecture**

*Transport layer role and positioning:*

Architectural role: OSI model positioning, e.g. Layer 4, protocol stack layering; layered architecture, e.g. encapsulation stack, protocol layering; protocol stack integration, e.g. TCP/IP stack, kernel network stack

End-to-end scope: end-to-end communication, e.g. host-to-host delivery, endpoint communication; process-to-process communication, e.g. socket pairs, application sessions; communication endpoints, e.g. source endpoint, destination endpoint

Layer interaction: network interaction, e.g. IP packets, routing paths; application interaction, e.g. application sockets, protocol interfaces; abstraction layers, e.g. transport abstraction, network independence.

*Transport protocols and services:*

Protocol types: Transmission Control Protocol (TCP), e.g. three-way handshake, sliding window; User Datagram Protocol (UDP), e.g. UDP datagrams, connectionless delivery; connection-oriented services, e.g. session state, handshake protocols; connectionless services, e.g. stateless delivery, best-effort models

Reliability and performance: reliability mechanisms, e.g. acknowledgements, retransmissions; latency characteristics, e.g. round-trip time, retransmission delay; throughput optimisation, e.g. congestion window, window scaling; protocol overhead, e.g. header overhead, control flags

Application fit: streaming applications, e.g. adaptive streaming, Real-time Transport Protocol (RTP) flows; file transfer protocols, e.g. File Transfer Protocol (FTP) sessions, Secure File Transfer Protocol (SFTP) transfer; real-time communication, e.g. Voice over IP (VoIP) streams, Web Real-Time Communication (WebRTC) sessions.

*Segmentation and reassembly:*

Packetisation: Data segmentation, e.g. maximum segment size (MSS) segmentation, data chunking; segment sizing, e.g. MSS values, maximum transmission unit (MTU) alignment; message boundaries, e.g. byte streams, datagram boundaries

Encapsulation and reconstruction: packet encapsulation, e.g. TCP segments, UDP datagrams; header information, e.g. sequence numbers, acknowledgement numbers; sequencing data units, e.g. ordered segments, out-of-order handling; reassembly processes, e.g. receive buffers, reordering buffers

Efficiency and overhead: transmission efficiency, e.g. bandwidth utilisation, goodput; fragmentation handling, e.g. IP fragmentation, MTU mismatch; transmission overhead, e.g. header overhead, retransmission overhead.

*Port addressing and multiplexing:*

Addressing mechanisms: Port numbers, e.g. well-known ports, ephemeral ports; registered ports, e.g. service ports, application ports; socket identification, e.g. IP-port pairs, socket tuples

Multiplexing processes: multiplexing processes, e.g. port multiplexing, connection tables; demultiplexing processes, e.g. stream separation, port routing; concurrent connections, e.g. simultaneous sessions, parallel connections

Service mapping: application binding, e.g. port binding, socket binding; service identification, e.g. service ports, protocol identifier.

LO2 **Discuss the mechanisms for reliable and unreliable data transmission**

*Reliable transmission mechanisms:*

Reliability control: Acknowledgements, e.g. ACK flags, cumulative ACKs; retransmission processes, e.g. timeout retransmission, fast retransmit; sequencing mechanisms, e.g. sequence numbers, ordered delivery

Error and integrity: error detection, e.g. checksum validation, cyclic redundancy check (CRC) checks; automatic repeat request (ARQ), e.g. stop-and-wait ARQ, selective repeat ARQ; data integrity verification, e.g. checksum verification, packet validation.

*Unreliable transmission mechanisms:*

Connectionless delivery: best-effort delivery, e.g. UDP delivery, datagram forwarding; absence of acknowledgements, e.g. no ACK flags, no retransmission control; reduced overhead, e.g. minimal headers, lightweight packets

Performance behaviour: packet loss tolerance, e.g. lossy transmission, dropped packets; latency optimisation, e.g. low-latency delivery, minimal delay; real-time communication, e.g. VoIP packets, streaming datagrams.

*Transmission optimisation techniques:*

Loss recovery: selective acknowledgements, e.g. selective acknowledgement (SACK) blocks, partial ACKs; fast retransmit, e.g. duplicate ACK triggers, early retransmission; timeout management, e.g. dynamic timeout, retransmission timeout

Performance tuning: congestion window adjustment, e.g. congestion window (cwnd) scaling, window growth; throughput optimisation, e.g. window scaling, bandwidth utilisation; latency reduction, e.g. delayed ACK tuning, Nagle algorithm control.

*Quality of service considerations:*

Performance metrics: latency, e.g. round-trip time, packet delay; jitter, e.g. delay variation, inter-packet delay; packet loss, e.g. loss rate, dropped packets

Service behaviour: throughput, e.g. data rate, transfer speed; bandwidth allocation, e.g. rate limiting, bandwidth reservation; service prioritisation, e.g. Quality of Service (QoS) classes, traffic prioritisation

Trade-offs: performance trade-offs, e.g. reliability vs latency, throughput vs delay; application requirements, e.g. streaming requirements, real-time constraints.

## LO3 **Review transport layer flow and congestion control algorithms**

*Flow control mechanisms:*

Flow regulation: windowing concepts, e.g. sliding window, receive window; sender–receiver co-ordination, e.g. ACK-based flow control, window updates; buffer management, e.g. receive buffers, send buffers

Rate limiting: prevention of receiver overload, e.g. receive window limits, flow throttling; data pacing, e.g. packet pacing, rate control.

*Congestion control principles:*

Congestion detection: network congestion indicators, e.g. packet loss, Explicit Congestion Notification (ECN) marking; congestion avoidance strategies, e.g. slow start, congestion avoidance phase; rate adaptation, e.g. cwnd adjustment, bandwidth adaptation

Fairness and efficiency: fairness principles, e.g. fair bandwidth sharing, flow fairness; bandwidth sharing, e.g. multi-flow allocation, congestion sharing; traffic management, e.g. rate limiting, queue control.

*Congestion control algorithms:*

Core algorithms: TCP Reno, e.g. fast retransmit, fast recovery; TCP Cubic, e.g. cubic window growth, high-speed scaling; TCP Bottleneck Bandwidth and Round-trip propagation time (BBR), e.g. bandwidth estimation, rate-based control

Control behaviour: slow start, e.g. exponential growth, initial window increase; congestion avoidance phase, e.g. linear growth, cwnd increase; congestion window management, e.g. cwnd scaling, window reduction.

*Network stability and efficiency:*

Performance control: throughput regulation, e.g. rate limiting, cwnd regulation; responsiveness to congestion, e.g. round-trip time (RTT) response, loss response; congestion collapse avoidance, e.g. backoff strategies, rate reduction

Efficiency trade-offs: fairness between flows, e.g. flow competition, bandwidth sharing; scalability under load, e.g. high-load performance, multi-flow scaling; performance optimisation, e.g. latency vs throughput, efficiency tuning.

LO4 **Implement transport layer configuration techniques to optimise network performance for an identified scenario**

*Transport layer configuration parameters:*

Packetisation parameters: MTU, e.g. 1500-byte MTU, jumbo frames; segment size configuration, e.g. MSS setting, MTU alignment; packet sizing, e.g. payload size, header-to-payload ratio

Buffering and queues: socket buffer sizing, e.g. send buffer, receive buffer; buffer allocation, e.g. kernel buffers, application buffers; queue management, e.g. first-in-first-out (FIFO) queues, fair queues

Timing and retransmission: timeout values, e.g. retransmission timeout (RTO) values, idle timeouts; retransmission thresholds, e.g. duplicate ACK threshold, loss detection threshold; protocol parameter tuning, e.g. TCP options, selective ACK

Connection controls: congestion window size, e.g. cwnd limits, initial cwnd; port configuration, e.g. port ranges, port reuse; protocol settings, e.g. timestamps, window scaling.

*Performance optimisation techniques:*

Parallelism and protocol choice: parallel data streams, e.g. parallel TCP connections, multi-stream transfers; protocol selection strategies, e.g. TCP vs UDP, Quick UDP Internet Connection (QUIC) selection

Congestion and throughput: congestion window adjustment, e.g. cwnd scaling, window growth; throughput optimisation, e.g. goodput tuning, bandwidth utilisation; efficiency tuning, e.g. retransmission reduction, overhead reduction

Latency and scheduling: latency reduction, e.g. RTT optimisation, queue delay reduction; traffic prioritisation, e.g. QoS marking, prioritised queues; packet scheduling, e.g. fair scheduling, priority scheduling.

*Monitoring and profiling tools:*

Metrics capture: performance measurement tools, e.g. iperf, netperf; throughput monitoring, e.g. rate graphs, flow rates; latency observation, e.g. RTT tracking, delay measurement

Packet inspection: packet analysis, e.g. packet capture, trace analysis; network analysers, e.g. Wireshark, tcpdump; protocol analysers, e.g. TCP stream analysis, flow decoding

Observability: traffic monitoring tools, e.g. flow collectors, Simple Network Management Protocol (SNMP) monitors; monitoring frameworks, e.g. Prometheus, telemetry systems; performance dashboards, e.g. Grafana dashboards, key performance indicator (KPI) panels.

*Scenario-based performance evaluation:*

Assessment design: evaluation criteria, e.g. performance thresholds, acceptance criteria; test scenarios, e.g. peak load, packet loss scenarios; configuration comparison, e.g. before/after tests, A/B comparison

Measurement and results: baseline measurements, e.g. baseline RTT, baseline throughput; performance metrics, e.g. latency, loss rate; measurable outcomes, e.g. throughput gain, delay reduction

Constraints and trade-offs: network conditions, e.g. bandwidth limits, path latency; performance constraints, e.g. CPU limits, buffer limits; optimisation trade-offs, e.g. latency vs throughput, reliability vs overhead.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explain the principles of transport layer architecture</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Critically evaluate the effectiveness of transport layer architectural principles, data transmission mechanisms and flow control algorithms in delivering reliable, stable and efficient data transmission networks.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the roles and protocols involved in the transport layer architecture.<br><br>P2 Describe, using a selection of real-world examples, the roles of the transport layer architecture in message segmentation and reassembly.</td><td>M1 Analyse, using a selection of real-world examples, how transport layer architecture supports end-to-end communication within the OSI and TCP models.</td></tr>
<tr><td colspan="2">LO2 Discuss the mechanisms for reliable and unreliable data transmission</td></tr>
<tr><td>P3 Discuss, using a selection of real-world examples, the mechanisms for reliable and unreliable data transmission.<br><br>P4 Explore, using a selection of real-world examples, the techniques for data transmission optimisation considering quality of service.</td><td>M2 Analyse, using a selection of real-world examples, the impact of reliable and unreliable data transmission mechanisms on network and organisational performance.</td></tr>
<tr><td colspan="2">LO3 Review transport layer flow and congestion control algorithms</td></tr>
<tr><td>P5 Review, using a selection of real-world examples, the transport layer flow and congestion control algorithms used to manage data transmission.<br><br>P6 Discuss, using a selection of real-world examples, how transport layer flow and congestion control contribute to effective network stability and efficiency.</td><td>M3 Analyse, using a selection of real-world examples, the benefits and limitations of transport layer flow and congestion control algorithms in managing the performance of different network contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Implement transport layer configuration techniques to optimise network performance for an identified scenario | | |
| **P7** Implement transport layer configuration techniques to improve network performance for an identified scenario.<br><br>**P8** Review, using a set of identified monitoring tools and techniques, the performance of the transport layer configuration for an identified scenario network, identifying areas for improvement. | **M4** Optimise, using the results of the monitoring data, the performance of the transport layer configuration techniques implemented for an identified scenario. | **D2** Critically evaluate the effectiveness of transport layer configuration decisions made in leading up to optimised network performance for the identified scenario. |

## Recommended Resources

### Textbooks

Forouzan, B.A. (2022) *Data Communications and Networking with TCP/IP Protocol Suite*. 6th edn. New York: McGraw-Hill Education.

Freedman, A. and Adato, L. (2023) *Practical Guide to Modern Networking Telemetry*. Sebastopol, CA: O'Reilly Media.

Kaufman, C., Perlman, R. and Speciner, M. (2022) *Network Security: Private Communication in a Public World*. 3rd edn. Harlow: Pearson Education.

Kurose, J.F. and Ross, K.W. (2023) *Computer Networking: A Top-Down Approach*. 9th edn. Harlow: Pearson.

Peterson, L.L. and Davie, B.S. (2024) *Computer Networks: A Systems Approach*. 6th edn. Cambridge, MA: Morgan Kaufmann.

Tanenbaum, A.S., Feamster, N. and Wetherall, D.J. (2021) *Computer Networks*. 6th edn. Harlow: Pearson.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.networkacademy.io/ccna/network-security/tcp-and-udp-essentials | NetworkAcademy<br>Clear explanation of TCP, UDP and port addressing, including multiplexing and application mapping<br>(Tutorial) |
| www.baeldung.com/cs/transport-layer-multiplexing-vs-demultiplexing | Baeldung on Computer Science<br>Overview of transport layer services including multiplexing and demultiplexing concepts<br>(General reference) |
| www.geeksforgeeks.org/computer-networks/multiplexing-and-demultiplexing-in-transport-layer/ | GeeksforGeeks<br>Detailed explanation of multiplexing and demultiplexing using port numbers and real-world examples<br>(Article) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.geeksforgeeks.org/computer-networks/differences-between-tcp-and-udp/ | GeeksforGeeks<br>Comparison of TCP and UDP transmission mechanisms including reliability, speed and use cases<br>(Article) |
| www.geeksforgeeks.org/computer-networks/tcp-congestion-control/ | GeeksforGeeks<br>Explanation of TCP congestion control mechanisms including slow start and congestion avoidance<br>(Article) |
| eureka.patsnap.com/article/how-to-perform-bandwidth-estimation-using-iperf-and-wireshark | Patsnap<br>Eureka guide to analysing network performance using tools such as iPerf and Wireshark<br>(Tutorial) |
| www.firewall.cx/networking/network-fundamentals/network-performance-testing.html | Firewall.cx<br>Overview of network performance testing techniques including throughput, latency and packet loss<br>(General reference) |
| www.bleepingcomputer.com/tutorials/tcp-and-udp-ports-explained/ | BleepingComputer<br>Guide to how ports operate within network communication and application delivery<br>(Tutorial) |
| tcpcc.systemsapproach.org/ | Systems Approach<br>Detailed exploration of congestion control algorithms and transport layer performance strategies<br>(General reference) |
| woshub.com/testing-network-bandwidth-using-iperf | Windows OS Hub<br>Step-by-step guide to testing network throughput and bandwidth using iPerf<br>(Tutorial) |

### Journals and articles

Cardwell, N. *et al*. (2017) 'BBR: congestion-based congestion control', *Communications of the ACM*, 60(2), pp. 58–66. Available at: doi.org/10.1145/3009824.

Fairhurst, G. *et al*. (2020) 'RFC 8899: Packetization Layer Path MTU Discovery for Datagram Transports', *IETF RFC 8899*. Available at: doi.org/10.17487/RFC8899.

Garcia, J., Sundberg, S. and Brunstrom, A. (2025) 'TCP Congestion Control Performance over Starlink', *ACM Applied Networking Research Workshop*. Available at: doi.org/10.1145/3744200.3744760.

Vielhaus, C.L. *et al*. (2026) 'Evaluating Transport Layer Congestion Control Algorithms: A Comprehensive Survey', *IEEE Communications Surveys and Tutorials*, 28. Available at: doi.org/10.1109/COMST.2025.3533303.

### Links

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*

*Unit 514: Quality of Service*

*Unit 515: Secure System Design*

*Unit 516: Internet of Things.*

 © Pearson Education Limited 2026

# Unit 534: Digital Sustainability

**Unit code:** K/652/3320

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Digital technologies play a central role in modern organisations, supporting communication, service delivery and innovation across sectors. However, their increasing use has led to growing concerns around energy consumption, resource use and environmental impact. Digital sustainability addresses these challenges by focusing on how digital systems and technologies can be designed, managed and used in ways that reduce environmental impact while continuing to support organisational and societal needs.

The aim of this unit is to develop students' understanding of digital sustainability as both a technical and organisational priority. Students will examine the core principles that underpin sustainable digital practices and explore how these principles influence decision-making within organisations. The unit also considers how sustainability requirements are shaped by governance, ethical considerations and regulatory frameworks, supporting students to apply informed and responsible approaches to digital systems.

This unit covers the key concepts and practices associated with digital sustainability. Students will investigate areas such as carbon impact, resource optimisation and digital life cycle management, and review how these influence operational efficiency and long-term value. The unit also explores the organisational and societal impact of digital sustainability, alongside the role of policy, ethics and regulation. These topics are brought together through the development of a digital sustainability strategy for a defined organisational scenario, enabling students to apply their knowledge in a practical context.

On successful completion of this unit, students will be able to evaluate digital sustainability practices, assess their organisational and societal impact, and develop strategies that align digital technologies with sustainability goals. These skills are relevant to a range of roles, including Digital Sustainability Analyst, IT Operations Manager, Sustainability Consultant, Technology Governance Specialist and Cloud Solutions Architect, supporting progression into careers where sustainable digital practice is increasingly required.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the core principles of digital sustainability

LO2 Review the impact of digital sustainability processes on organisations and society

LO3 Explore the role of governments, ethics and regulatory frameworks in digital sustainability

LO4 Develop a sustainability strategy to demonstrate how digital technologies can be used to enable sustainable practices in an identified organisational scenario.

## Essential Content

LO1 **Discuss the core principles of digital sustainability**

*Digital sustainability principles:*

Core sustainability dimensions: Environmental sustainability, e.g. carbon impact reduction, energy efficiency; economic sustainability, e.g. cost-efficient systems, resource optimisation; social sustainability, e.g. digital inclusion, equitable access

Strategic integration: long-term digital value, e.g. life cycle value, sustainable return on investment (ROI); responsible technology use, e.g. ethical system use, governance controls; sustainable innovation, e.g. green IT innovation, low-impact design

Governance and responsibility: ethical digital practices, e.g. responsible AI models, bias mitigation techniques; sustainability integration, e.g. Environmental, Social, and Governance (ESG)-aligned IT strategy, policy frameworks; resource stewardship, e.g. energy governance, resource tracking.

*Carbon footprint of digital systems:*

Energy and emissions: Energy consumption, e.g. data centre kWh usage, server power draw; greenhouse gas emissions, e.g. carbon dioxide equivalent ($CO_2e$) metrics, Scope 2 emissions; cloud computing footprint, e.g. cloud workload emissions, shared infrastructure impact

Measurement and monitoring: energy monitoring, e.g. energy dashboards, usage telemetry; carbon measurement, e.g. carbon accounting tools, emissions calculators; emission tracking, e.g. reporting dashboards, monitoring platforms

Impact and reduction: infrastructure efficiency, e.g. power usage effectiveness (PUE) optimisation, efficient hardware profiles; environmental impact assessment, e.g. life cycle assessment (LCA), carbon audits; carbon reduction strategies, e.g. workload scheduling, renewable-powered hosting.

*Resource optimisation:*

System efficiency: energy efficiency, e.g. low-power processing, energy-aware workloads; computational efficiency, e.g. algorithm optimisation, compute efficiency; storage efficiency, e.g. data deduplication, storage tiering; network efficiency, e.g. traffic shaping, bandwidth optimisation

Infrastructure utilisation: hardware utilisation, e.g. CPU utilisation, server density; virtualisation, e.g. virtual machines, container orchestration; workload optimisation, e.g. autoscaling, load balancing

Resource management: resource allocation, e.g. dynamic provisioning, capacity allocation; capacity utilisation, e.g. utilisation metrics, demand forecasting; infrastructure optimisation, e.g. system tuning, efficiency adjustments.

*Digital life cycle considerations:*

Life cycle stages: design phase impacts, e.g. energy-aware design, sustainable architecture; development practices, e.g. green coding, efficient algorithms; deployment environments, e.g. cloud regions, edge deployment

Operational phase: operational usage, e.g. runtime efficiency, workload execution; maintenance processes, e.g. patch optimisation, performance tuning; system longevity, e.g. hardware reuse, life cycle extension

End-of-life management: disposal strategies, e.g. e-waste recycling, certified disposal; replacement cycles, e.g. hardware refresh cycles, upgrade planning; life cycle optimisation, e.g. reuse strategies, circular life cycle models.

LO2 **Review the impact of digital sustainability processes on organisations and society**

*Organisational impacts:*

Operational performance: operational efficiency, e.g. energy-per-transaction, processing efficiency metrics; cost optimisation, e.g. cost-per-workload, cloud cost profiling; risk management, e.g. sustainability risk registers, carbon risk exposure

Strategic alignment: sustainability objectives, e.g. net-zero targets, Environmental, Social, and Governance (ESG) KPIs; digital transformation alignment, e.g. cloud optimisation strategies, green IT roadmaps; organisational resilience, e.g. sustainable infrastructure design, resource redundancy

Resource and process impact: resource utilisation, e.g. CPU utilisation rates, storage efficiency; process optimisation, e.g. workflow automation, energy-aware processes; environmental performance, e.g. emissions reduction metrics, sustainability dashboards.

*Societal impacts:*

Social and environmental outcomes: environmental responsibility, e.g. emissions reporting, carbon disclosure; ethical technology use, e.g. responsible AI deployment, bias auditing; digital inclusion, e.g. broadband access metrics, accessibility compliance

Public and stakeholder impact: public trust, e.g. transparency reporting, sustainability disclosures; stakeholder trust, e.g. ESG reporting frameworks, governance metrics; societal value, e.g. sustainable service delivery, community impact metrics

Equity and long-term effects: digital equity, e.g. access equality metrics, service availability; sustainable development, e.g. SDG alignment, environmental targets; long-term impact, e.g. life cycle impact modelling, sustainability forecasting.

*Business value and performance:*

Value creation: reputation management, e.g. sustainability ratings, ESG scores; regulatory readiness, e.g. compliance audits, reporting requirements; innovation capability, e.g. green technology adoption, sustainable system design

Competitive positioning: competitive advantage, e.g. energy-efficient systems, low-carbon services; market positioning, e.g. sustainability benchmarking, industry comparisons; brand value, e.g. sustainability indices, impact reporting

Operational outcomes: operational performance, e.g. efficiency KPIs, system performance metrics; investment attractiveness, e.g. ESG investment scores, sustainability ratings; business continuity, e.g. resilient infrastructure, risk mitigation.

*Change implications:*

Organisational change: cultural change, e.g. sustainability culture metrics, behavioural change; organisational readiness, e.g. maturity models, capability assessments; governance structures, e.g. sustainability governance frameworks, oversight models

Workforce and capability: skills development, e.g. green IT training, sustainability competencies; training requirements, e.g. certification programmes, upskilling initiatives; workforce adaptation, e.g. role restructuring, digital skill shifts

Transformation processes: change management, e.g. structured change frameworks, adoption models; process transformation, e.g. low-energy workflows, automated systems; technology adoption, e.g. cloud migration, optimisation tools; capability development, e.g. digital maturity models, continuous improvement frameworks.

LO3 **Explore the role of governments, ethics and regulatory frameworks in digital sustainability**

*Government roles:*

Policy and strategy: National sustainability strategies, e.g. UK Net Zero Strategy, EU Green Deal; public sector digital policies, e.g. Government Digital Service standards, Green ICT strategies; international agreements, e.g. Paris Agreement, UN Sustainable Development Goals (SDGs)

Regulation and incentives: regulatory oversight, e.g. environmental reporting regulations, compliance enforcement bodies; sustainability incentives, e.g. carbon tax schemes, renewable energy incentives; environmental mandates, e.g. Streamlined Energy and Carbon Reporting (SECR), Energy Savings Opportunity Scheme (ESOS)

Governance mechanisms: policy frameworks, e.g. ESG frameworks, Organisation for Economic Co-operation and Development (OECD) digital policy frameworks; government regulation, e.g. General Data Protection Regulation (GDPR), Digital Services Act; funding programmes, e.g. green innovation funding, digital transformation grants.

*Ethical considerations:*

Core ethics: Responsible innovation, e.g. sustainable system design, low-carbon architectures; digital equity, e.g. accessibility standards (WCAG), inclusion metrics; environmental ethics, e.g. carbon accountability, life cycle responsibility

Transparency and accountability: transparency principles, e.g. sustainability reporting, open disclosure; accountability mechanisms, e.g. governance controls, audit trails; ethical governance, e.g. corporate governance codes, oversight boards

Data and decision-making: data ethics, e.g. data minimisation (GDPR principles), responsible data use; fairness principles, e.g. bias mitigation, equitable system outcomes; ethical decision-making, e.g. impact assessments, ESG risk models.

*Regulatory frameworks:*

Compliance and standards: environmental regulation, e.g. ISO 14001 environmental management, EU emissions directives; digital compliance standards, e.g. GDPR, ISO/IEC 27001; sustainability reporting frameworks, e.g. Global Reporting Initiative (GRI), Task Force on Climate-related Financial Disclosures (TCFD); evolving practice

Monitoring and enforcement: enforcement mechanisms, e.g. regulatory audits, compliance inspections; compliance monitoring, e.g. reporting platforms, audit logs; legal obligations, e.g. statutory reporting duties, compliance requirements

Integration and alignment: standards alignment, e.g. ISO 14001 integration, ESG benchmarking; compliance frameworks, e.g. ITIL governance, Control Objectives for Information and Related Technologies (COBIT) frameworks; regulatory impact, e.g. compliance cost modelling, operational constraints.

*Governance models:*

Organisational governance: governance frameworks, e.g. COBIT, ISO governance models; compliance management, e.g. compliance tracking systems, audit processes; organisational governance, e.g. board-level ESG oversight, governance structures

Control and accountability: accountability mechanisms, e.g. KPI tracking, Responsible, Accountable, Consulted, and Informed (RACI) matrices; decision-making structures, e.g. governance boards, approval workflows; control mechanisms, e.g. policy enforcement systems, monitoring tools

Collaboration and oversight: cross-sector collaboration, e.g. public-private partnerships, industry consortia; monitoring systems, e.g. sustainability dashboards, compliance tools; governance integration, e.g. integrated reporting models, enterprise governance frameworks.

LO4 **Develop a sustainability strategy to demonstrate how digital technologies can be used to enable sustainable practices in an identified organisational scenario**

*Digital sustainability strategy components:*

Strategic planning: strategic objectives, e.g. net-zero targets, ESG targets; sustainability targets, e.g. carbon reduction targets, energy efficiency KPIs; organisational alignment, e.g. ESG-aligned IT strategy, digital transformation alignment

Implementation structure: technology roadmap, e.g. cloud optimisation roadmap, infrastructure modernisation; performance indicators, e.g. energy KPIs, carbon intensity metrics; sustainability metrics, e.g. CO2e tracking, energy usage metrics

Governance integration: governance integration, e.g. ESG governance frameworks, IT governance models (COBIT); resource allocation, e.g. budget allocation, resource planning; risk management, e.g. sustainability risk registers, ESG risk models.

*Digital technologies enabling sustainability:*

Infrastructure and platforms: cloud technologies, e.g. low-carbon cloud regions, carbon-aware scheduling; monitoring platforms, e.g. energy dashboards, sustainability analytics tools; optimisation software, e.g. workload schedulers, efficiency tuning tools

Automation and intelligence: data analytics, e.g. carbon analytics, usage analytics; automation tools, e.g. autoscaling systems, energy-aware automation; AI solutions, e.g. predictive optimisation, energy forecasting

Smart systems: Internet of Things (IoT) systems, e.g. smart sensors, energy monitoring devices; digital twins, e.g. simulated system models, optimisation models; smart infrastructure, e.g. smart grids, intelligent buildings.

*Monitoring and reporting mechanisms:*

Measurement systems: sustainability metrics, e.g. CO2e metrics, energy consumption; performance measurement systems, e.g. monitoring platforms, telemetry systems; KPI tracking, e.g. ESG KPIs, efficiency metrics

Reporting frameworks: reporting frameworks, e.g. Global Reporting Initiative (GRI) standards, Task Force on Climate-related Financial Disclosures (TCFD) disclosures; compliance reporting, e.g. Streamlined Energy and Carbon Reporting (SECR) reporting, ESG disclosures; environmental reporting, e.g. carbon reports, sustainability reports

Data and analytics: data collection methods, e.g. telemetry collection, sensor data; analytics dashboards, e.g. Grafana dashboards, reporting dashboards; data visualisation, e.g. KPI charts, trend analysis

Governance and audit: audit processes, e.g. sustainability audits, compliance audits; benchmarking tools, e.g. ESG benchmarking, performance comparison; transparency mechanisms, e.g. public reporting, disclosure platforms.

*Implementation considerations:*

Feasibility and constraints: scalability planning, e.g. cloud scaling models, infrastructure scaling; resource availability, e.g. compute resources, energy capacity; organisational capability, e.g. digital maturity models, capability frameworks

Integration and delivery: technology integration, e.g. system integration, platform integration; infrastructure readiness, e.g. cloud readiness, network capacity; operational constraints, e.g. performance limits, system capacity

Risk and alignment: risk management, e.g. implementation risks, sustainability risks; feasibility analysis, e.g. cost–benefit analysis, ROI modelling; stakeholder alignment, e.g. stakeholder mapping, governance approval

Cost and performance: cost considerations, e.g. cost modelling, pricing models; trade-offs, e.g. cost vs performance, sustainability vs efficiency; strategic alignment, e.g. ESG strategy alignment, business alignment.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Discuss the core principles of digital sustainability</td><td rowspan="4">LO1 and LO2<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of implementing digital sustainability principles and processes in reducing environmental impact across digital systems and supporting long-term organisational and societal sustainability.</td></tr>
<tr><td>P1 Discuss, using a selection of real-world examples, the core principles of digital sustainability including carbon footprints and the device life cycle.<br><br>P2 Explore, using a selection of real-world examples, how digital sustainability principles contribute to resource optimisation in organisational contexts.</td><td>M1 Assess, using a selection of real-world examples, the benefits and limitations of implementing digital sustainability principles for identified organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Review the impact of digital sustainability processes on organisations and society</td></tr>
<tr><td>P3 Review, using a selection of real-world examples, the impact of digital sustainability processes on the value and performance of organisations.<br><br>P4 Discuss, using a selection of real-world examples, the impact of digital sustainability processes on society.</td><td>M2 Analyse, using a selection of real-world examples, the organisational and societal implications of adopting digital sustainability processes.</td></tr>
<tr><td colspan="2">LO3 Explore the role of governments, ethics and regulatory frameworks in digital sustainability</td><td rowspan="2">D2 Critically evaluate, using a selection of real-world examples, the extent to which governmental, ethical and regulatory frameworks support effective digital sustainability outcomes.</td></tr>
<tr><td>P5 Explore, using a selection of real-world examples, the role of governments, ethics and regulatory frameworks in digital sustainability.<br><br>P6 Discuss, using a selection of real-world examples, how governance models allow organisations to comply with governments, ethics and regulatory frameworks in delivering digital sustainability.</td><td>M3 Analyse, using a selection of real-world examples, the impact of governance, ethical considerations and regulation on digital sustainability practices.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Develop a sustainability strategy to demonstrate how digital technologies can be used to enable sustainable practices in an identified organisational scenario | | |
| **P7** Develop a sustainability strategy to demonstrate how digital technologies can be used to enable sustainable practices in an identified organisational scenario.<br><br>**P8** Review the sustainability strategy with a range of identified stakeholders to identify strengths and areas for improvement. | **M4** Evaluate, using the stakeholder feedback, the effectiveness of the digital sustainability strategy in meeting the practice requirements of the identified organisational scenario. | **D3** Justify the decisions made in the development of the digital sustainability strategy that allows it to effectively meet the practice requirements of an identified scenario. |

## Recommended Resources

### Textbooks

Galipeas, D. and Endress, T. (eds.) (2023) *A Practical Approach to Digital Technologies for Sustainability*. Boca Raton: CRC Press.

Halsey, M. (2022) *The Green IT Guide: Ten Steps Toward Sustainable and Carbon-Neutral IT Infrastructure*. Berkeley, CA: Apress.

Hazas, M. and Nathan, L. (eds.) (2021) *Digital Technology and Sustainability: Engaging the Paradox.* London: Routledge.

Lazazzara, A., Reina, R. and Za, S. (2024) *Towards Digital and Sustainable Organisations: People*, *Platforms, and Ecosystems*. Cham: Springer.

Schoormann, T. *et al*. (2025) *Digital Sustainability: Understanding and Managing Tensions*. Cham: Springer.

Sharma, V. *et al*. (eds.) (2023) *Sustainable Digital Technologies: Trends, Impacts, and Assessments*. Boca Raton: CRC Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.worldbank.org/en/topic/digital/publication/measuring-the-emissions-energy-footprint-of-the-ict-sector-implications-for-climate-action | World Bank<br>Measuring the emissions and energy footprint of the ICT sector and its implications for sustainability and policy<br>(Report) |
| www.ericsson.com/en/reports-and-papers/mobility-report/articles/ict-carbon-footprint-decreasing | Ericsson Mobility Report<br>Analysis of ICT sector carbon footprint, energy use and sustainability trends in digital infrastructure<br>(Article) |
| www.un.org/en/un75/impact-digital-technologies | United Nations<br>Overview of how digital technologies impact society, sustainability and global development goals<br>(General reference) |

| | |
|---|---|
| www.meegle.com/en_us/topics/green-it-strategies/green-it-in-cloud-computing | Meegle<br>Introduction to Green IT in cloud computing, including energy optimisation, resource use and sustainability practices<br>(Article) |
| www.techprocomp.com/blog/green-it-sustainable-technology-practices-for-a-greener-future/ | TechProComp<br>Guide to implementing Green IT practices to reduce energy use, e-waste and environmental impact<br>(Article) |
| www.gdprregister.eu/articles/esg-and-data-protection/ | GDPR Register<br>Explains the relationship between ESG frameworks and data protection regulations such as GDPR<br>(Article) |
| digitalregulation.org/monitoring-sustainability-incorporating-esg-into-ict-policy-making-and-regulation/ | Digital Regulation Platform<br>How governments use ESG frameworks and regulation to monitor sustainability in ICT sectors<br>(Article) |
| www.deloitte.com/uk/en/Industries/power-utilities-renewables/blogs/set-and-execute-your-sustainability-it-and-data-strategy.html | Deloitte<br>Guide to developing and implementing an IT and data strategy to support sustainability objectives<br>(Article) |
| www.ibm.com/think/topics/sustainable-it | IBM<br>Overview of sustainable IT, including environmental impact, energy efficiency and life cycle management of IT systems<br>(General reference) |

| | |
|---|---|
| www.microsoft.com/en/sustainability/cloud | Microsoft<br>Explains how cloud technologies and ESG data can be used to measure, manage and improve organisational sustainability<br>(General reference) |

**Journals and articles**

Istrate, R. *et al*. (2024) 'The environmental sustainability of digital content consumption', *Nature Communications*, 15, Article 3724. Available at: doi.org/10.1038/s41467-024-47621-w.

Masri, S.G. and El-Fadel, M. (2026) 'Governance of digital transformation for sustainable development: aligning digital innovation with the sustainable development goals', *Frontiers in Sustainable Cities*, 8. Available at: doi.org/10.3389/frsc.2026.1743552.

Sahar, R. *et al*. (2026) 'Adoption of digital technologies in operation to improve sustainability', *Future Business Journal*, 12, Article 33. Available at: doi.org/10.1186/s43093-026-00745-5.

Schork, S., Özdemir-Kaluk, D. and Zerey, C. (2025) 'Understanding Innovation and Sustainability in Digital Organizations: A Mixed-Method Approach', *Sustainability*, 17(2), 415. Available at: doi.org/10.3390/su17020415.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*.

# Unit 535: Virtual and Augmented Reality

**Unit code:** L/652/3321

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Virtual and augmented reality technologies are now firmly established across a wide range of digital sectors, including software development, immersive media, education, healthcare, training and enterprise systems. Advances in hardware, software platforms and development tools have made immersive experiences more accessible and commercially viable, leading to increased demand for professionals who understand how Virtual Reality (VR) and Augmented Reality (AR) systems are designed, developed and applied in real-world contexts.

The aim of this unit is to develop students' understanding of the technical foundations and development processes that underpin VR and AR systems. Students will examine how hardware and software components work together to create immersive experiences and will review the tools, frameworks and techniques used to design, build and deploy VR and AR solutions. The unit also enables students to apply this knowledge through the development of a practical solution for an identified scenario.

The unit covers key concepts required for effective immersive development, including VR and AR hardware components, tracking and sensing technologies, system performance considerations and software platforms. Students will explore industry-standard development engines, software development kits and content creation pipelines used in immersive production. The unit also addresses interaction design, usability, testing and validation, alongside the presentation of immersive solutions to stakeholders, taking into account ethical, accessibility, safety and user well-being considerations.

On successful completion of this unit, students will be able to analyse immersive technology requirements, develop and evaluate VR or AR solutions, and communicate technical outcomes to both technical and non-technical audiences. These skills support progression into employment or further study in areas such as immersive technology and software development. Relevant job roles include Immersive Application Developer, XR Developer, Software Developer, Interaction Designer, Digital Content Developer, and Emerging Technology Specialist.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore the hardware and software requirements for VR and AR systems

LO2 Review the tools and techniques used to build and deploy VR and AR environments/solutions

LO3 Develop a VR or AR solution for an identified scenario using appropriate tools and frameworks

LO4 Present the VR or AR solution developed for the identified scenario to a range of key stakeholders.

## Essential Content

LO1 **Explore the hardware and software requirements for VR and AR systems**

*VR and AR hardware components:*

Display hardware: head-mounted displays e.g. immersive visual output, stereoscopic viewing; AR smart devices e.g. optical see-through displays, contextual overlays; examples: Meta Quest 3, Apple Vision Pro, Microsoft HoloLens 2

Processing, power and connectivity: processing units e.g. real-time rendering, graphical processing; power requirements e.g. battery capacity, thermal limits; connectivity interfaces e.g. wireless protocols, data throughput; examples: NVIDIA RTX GPU platforms, Apple M-series SoC, Qualcomm Snapdragon XR2

Input and interaction devices: input devices e.g. interaction control, user commands; controllers e.g. handheld input, haptic feedback.

*Tracking and sensing technologies:*

Tracking systems: positional tracking systems e.g. spatial positioning, co-ordinate mapping; inside-out tracking e.g. onboard camera tracking, environment recognition; outside-in tracking e.g. external sensors, fixed reference points; examples: SteamVR Tracking, Meta Insight Tracking, ARKit Motion Tracking

Motion and depth detection: motion sensors e.g. accelerometers, gyroscopes; depth sensors e.g. distance measurement, spatial depth mapping

Tracking quality constraints: latency considerations e.g. motion-to-photon delay, system responsiveness; accuracy constraints e.g. positional precision, tracking drift.

*Software platforms and system compatibility:*

Core system software: operating systems e.g. resource management, hardware abstraction; device drivers e.g. hardware communication, peripheral support; firmware e.g. low-level control, system stability

Runtime and execution layers: runtime environments e.g. application execution, system mediation

Cross-platform support: cross-platform compatibility e.g. portability, device independence; system compatibility layers e.g. interoperability, legacy support; examples: OpenXR, Unity XR Plugin Framework, Unreal Engine XR.

*Performance and optimisation requirements:*

Timing and responsiveness: frame rate stability e.g. smooth rendering, interaction consistency; latency thresholds e.g. response time limits, user perception; examples: NVIDIA VRWorks, Oculus Asynchronous Timewarp, SteamVR Motion Smoothing

Resource management: memory usage e.g. asset allocation, runtime efficiency; processing overhead e.g. computational load, background processes.

LO2 **Review the tools and techniques used to build and deploy VR and AR environments/solutions**

*Development engines and frameworks:*

Real-time 3D engines: real-time 3D engines e.g. scene rendering, interactive environments; examples: Unity, Unreal Engine, Godot XR

Physics and rendering systems: physics engines e.g. collision handling, motion simulation; rendering pipelines e.g. graphics processing, visual output

Scene and logic management: scene management systems e.g. scene organisation, object hierarchies; scripting environments e.g. interaction logic, behaviour control.

*Software development kits (SDKs) and standards:*

Software development kits (SDK) frameworks: AR and VR SDKs e.g. platform integration, device access; device-specific toolkits e.g. hardware optimisation, proprietary features; examples: ARKit, ARCore, OpenXR

Cross-platform interfaces: cross-platform standards e.g. interoperability, portability; application programming interfaces e.g. system communication, feature access.

*Content creation and asset pipelines:*

Asset creation tools: 3D modelling tools e.g. object creation, spatial modelling; animation software e.g. motion design, interaction animation; examples: Blender, Autodesk Maya, Adobe Substance 3D

Material and visual assets: texture creation e.g. surface detail, visual realism; material systems e.g. shading models, visual properties

Pipeline management: asset optimisation e.g. performance efficiency, file size reduction.

*Interaction and user interface systems:*

Input handling: input mapping systems e.g. device inputs, interaction binding

Gesture and tracking interfaces: gesture recognition e.g. hand movement interpretation; hand tracking e.g. spatial interaction, natural input; examples: Meta Hand Tracking, Ultraleap Hand Tracking, Apple Vision Pro Hand Tracking

User feedback systems: spatial user interfaces e.g. 3D UI placement, immersive menus; feedback mechanisms e.g. visual cues, interaction responses.

*Deployment and distribution methods:*

Build and packaging processes: platform-specific build processes e.g. target optimisation, system configuration; application packaging e.g. distributable builds, deployment formats; examples: Meta Quest Build Pipeline, SteamVR Distribution, Microsoft Mixed Reality Deployment

Installation and update workflows: device installation workflows e.g. sideloading, device setup; update mechanisms e.g. version updates, maintenance cycles.

*Performance optimisation techniques:*

Rendering and asset strategies: level of detail strategies e.g. model scaling, performance balancing; asset compression e.g. storage efficiency, load optimisation; examples: Unity Adaptive Performance, Unreal Nanite, NVIDIA Deep Learning Super Sampling (DLSS)

System resource control: resource management approaches e.g. memory control, processing efficiency.

LO3 **Develop a VR or AR solution for an identified scenario using appropriate tools and frameworks**

*Scenario definition and solution scope:*

Use-case characteristics: use-case characteristics e.g. user goals, task context; user context e.g. target users, usage environment; examples: KLM VR Cockpit Trainer, Walmart VR Training Academy, Meta Horizon Workrooms

Requirements definition: functional requirements e.g. system behaviour, interaction features; experiential goals e.g. immersion quality, user engagement.

*Solution architecture and design structure:*

Application structure: application structure e.g. scene hierarchy, system organisation; scene organisation e.g. logical flow, modular scenes

Component and data design: component relationships e.g. system dependencies, interaction modules; data flow models e.g. input processing, state management; examples: Unity ECS Architecture, Unreal Engine Blueprint System, NVIDIA Omniverse.

*Implementation frameworks and tools:*

Development environment selection: selected development engines e.g. Unity, Unreal Engine; SDK integration e.g. device application programming interfaces (APIs), platform features; examples: Unity XR Toolkit, Unreal Engine XR Framework, AR Foundation

Asset and interaction implementation: asset incorporation e.g. 3D models, animations; interaction logic structures e.g. event handling, interaction scripts.

*User interaction and experience considerations:*

Navigation and interaction design: navigation models e.g. teleportation, guided movement; interaction feedback e.g. visual cues, haptic response

Usability and accessibility: usability principles e.g. intuitive controls, consistency; accessibility considerations e.g. adjustable interfaces, inclusive design; examples: Meta Hand Tracking, Ultraleap Hand Tracking, Apple Vision Pro Interaction Model.

*Testing and validation considerations:*

Testing approaches: functional testing e.g. feature verification, interaction checks; performance testing e.g. frame rate stability, latency monitoring; examples: Unity Profiler, SteamVR Performance Graph, OVR Metrics Tool

Validation and compatibility: usability validation e.g. user testing, feedback analysis; device compatibility checks e.g. hardware support, platform consistency.

LO4 **Present the VR or AR solution developed for the identified scenario to a range of key stakeholders**

*Presentation structure and communication focus:*

Solution overview: solution overview e.g. purpose summary, system scope; scenario relevance e.g. contextual fit, problem alignment

Design rationale: design rationale e.g. development choices, technical justification; functionality explanation e.g. feature walkthroughs, system behaviour

Outcome demonstration: outcome demonstration e.g. live walkthroughs, recorded demonstrations.

*Stakeholder considerations:*

Stakeholder types: technical stakeholders e.g. developers, system architects; non-technical stakeholders e.g. clients, managers

Perspective and context: user perspectives e.g. usability expectations, experience needs; organisational interests e.g. business goals, deployment constraints

Decision contexts: decision-making contexts e.g. approval processes, investment considerations.

*Ethical, accessibility and well-being considerations:*

Ethical practice: inclusive design principles e.g. equitable access, bias awareness; data protection awareness e.g. user privacy, data handling

Accessibility and safety: accessibility requirements e.g. adjustable interfaces, alternative interactions; user safety e.g. physical space awareness, usage limits

Well-being impacts: user well-being impacts e.g. motion comfort, cognitive load.

*Feedback and evaluation inputs:*

Feedback collection: stakeholder responses e.g. verbal feedback, evaluation comments; usability feedback e.g. user observations, task success

Evaluation and improvement: communication effectiveness indicators e.g. clarity, audience understanding; improvement considerations e.g. refinement actions, iterative updates.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explore the hardware and software requirements for VR and AR systems</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a selection of real-world examples, the effectiveness of selected tools and techniques in supporting the development and deployment of identified hardware and software VR and AR solutions.</td></tr>
<tr><td>P1 Explore, using a selection of real-world examples, the hardware components and performance requirements of VR and AR systems in relation to system functionality.<br><br>P2 Explain, using a selection of real-world examples, the software components and compatibility requirements of VR and AR systems in relation to system operation.</td><td>M1 Analyse, using a selection of real-world examples, the relationship between hardware and software requirements in supporting effective VR and AR system performance.</td></tr>
<tr><td colspan="2">LO2 Review the tools and techniques used to build and deploy VR and AR environments/solutions</td></tr>
<tr><td>P3 Review, using a selection of real-world examples, the development engines and software development kits (SDKs) used to build VR and AR environments.<br><br>P4 Explore, using a selection of real-world examples, deployment and optimisation techniques used to deliver VR and AR solutions across platforms.</td><td>M2 Analyse, using a selection of real-world examples, the suitability of tools and techniques used to build and deploy VR and AR environments for different solution requirements.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Develop a VR or AR solution for an identified scenario using appropriate tools and frameworks | | **LO3 and LO4**<br>**D2** Justify how design and development decisions contribute to the effectiveness of the developed VR or AR solution in meeting the identified scenario requirements. |
| **P5** Design a VR or AR solution that addresses the requirements of an identified scenario using appropriate tools and frameworks.<br>**P6** Develop a VR or AR solution that addresses the requirements of an identified scenario using appropriate tools and frameworks. | **M3** Assess the design and implementation of the developed VR or AR solution in how well it meets the identified scenario requirements. | |
| **LO4** Present the VR or AR solution developed for the identified scenario to a range of key stakeholders | | |
| **P7** Present the developed VR or AR solution to a range of key stakeholders to obtain feedback on the suitability of the solution in meeting identified requirements and the professionalism of the presentation.<br>**P8** Review stakeholder feedback on the presented VR or AR solution to identify strengths and limitations in the presentation and the developed VR or AR solution. | **M4** Optimise, using the results of the stakeholder feedback, the developed VR or AR solution to better meet the requirements of the identified scenario. | |

## Recommended Resources

### Textbooks

Braun, A. and Rizzo, R. (2023) *XR Development with Unity: A Beginner's Guide to Creating Virtual, Augmented, and Mixed Reality Experiences Using Unity*. Birmingham: Packt Publishing.

Grigore, C. and Grigore, G. (2024) *Virtual Reality Technology*. Hoboken, NJ: Wiley-IEEE Press.

Hillmann, C. (2021) *UX for XR: User Experience Design and Strategies for Immersive Technologies*. Berkeley, CA: Apress.

LaValle, S.M. (2023) *Virtual Reality*. Cambridge: Cambridge University Press.

Rauschnabel, P.A., Felix, R. and Hinsch, C. (2023) *Augmented Reality Marketing: How Mobile AR Is Changing the World*. Wiesbaden: Springer Gabler.

Tanwar, S. *et al*. (2023) *Handbook of Augmented and Virtual Reality*. Berlin: De Gruyter.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| developers.meta.com/horizon/develop/spatial-sdk | Meta Horizon<br>Building with Meta Spatial SDK<br>(Development tool) |
| learn.microsoft.com/en-us/windows/mixed-reality | Microsoft Mixed Reality<br>Mixed Reality documentation<br>(Resource Hub) |
| docs.unity3d.com | Unity<br>Current Unity 6 User Manual<br>(Document Resource) |
| www.w3.org/immersive-web | W3C<br>Immersive Web Working Group<br>(General reference) |
| www.geeksforgeeks.org/techtips/virtual-reality-introduction/ | GeeksForGeeks<br>Virtual Reality Introduction<br>(General reference) |

| | |
|---|---|
| www.geeksforgeeks.org/blogs/basics-augmented-reality/ | GeeksForGeeks<br>Basics of Augmented Reality<br>(Blog) |
| developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API/Fundamentals | Mozilla<br>Fundamentals of WebXR<br>(Resource) |
| dev.epicgames.com/documentation/en-us/unreal-engine/developing-for-xr-experiences-in-unreal-engine | EpicGames<br>XR Development<br>(Resource) |
| threejs.org/manual/en/webxr-basics.html | Three.js<br>VR<br>(Tutorial) |
| forums.unrealengine.com/t/community-tutorial-unreal-engine-5-6-x-for-meta-quest-vr/2689069 | EpicGames<br>Community Tutorial: Unreal Engine 5.6.x for MetaQuest VR<br>(Resource) |
| https://www.nvidia.com/en-gb/geforce/news/fcat-vr-download-and-how-to-guide/ | NVIDIA<br>Free NVIDIA FCAT VR Performance Analysis Tool<br>(Downloadable Resource) |

## Journals and articles

Cho, Y. and Kim, J. (2022) 'A Study on the Comparison of the Virtual Reality Development Environment in Unity and Unreal Engine 4', *Journal of the Korea Computer Graphics Society*, 28(5). Available at: doi.org/10.15701/kcgs.2022.28.5.1.

Hogan, J. *et al*. (2023) 'Analyzing Performance Issues of Virtual Reality Applications', *arXiv*. Available at: arxiv.org/abs/2211.02013.

Zallio, M. *et al*. (2024) 'The Ethics of Immersion: A Scoping Review of VR and AR Technologies', *Accessibility, Assistive Technology and Digital Environments*, 121, pp. 91–101. Available at: doi.org/10.54941/ahfe1004617.

Zhang, X. and Cheng, H. (2024) 'Interactive prototype development and evaluation of low-carbon products through virtual reality technology', *International Journal of Low-Carbon Technologies*, 19, pp. 2725–2732. Available at: doi.org/10.1093/ijlct/ctae238.

## Links

This unit links to the following related units:

*Unit 401: Fundamentals of Computer Science (Pearson-set)*

*Unit 420: Core AI*

*Unit 421: AI Automation and Digital Productivity*

*Unit 423: Agentic AI*

*Unit 520: Applied AI.*

# Unit 536: Work-Based Learning

**Unit code:** M/652/3322

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Work-based learning is a key part of preparing students for employment in the digital sector, where technical knowledge must be applied alongside professional behaviours in real working environments. Organisations increasingly expect individuals to understand how digital roles operate in practice, including how teams collaborate, how work is structured and how responsibilities are managed across a range of contexts. This unit introduces students to the purpose and value of work experience, and how it supports the development of practical skills and workplace awareness.

The aim of this unit is to enable students to engage in structured work-based learning and to apply their knowledge within a professional digital environment. Students will develop an understanding of workplace expectations, the nature of work in the digital sector and the behaviours required to operate effectively in a professional setting. The unit also supports students in recognising how work is changing in response to technological developments and how this influences roles, skills and career progression.

The unit covers key areas required to understand and undertake work experience in the digital sector. Students will explore the structure of digital workplaces, including organisational types, team structures and working models such as remote and hybrid environments. They will examine roles, responsibilities and professional expectations, alongside the impact of technological change on working practices and skills demand. The unit also includes preparation for a work placement, participation in workplace activities and structured reflection on performance, learning and development.

On successful completion of the unit, students will be able to demonstrate professional behaviours in a workplace context, reflect on their performance and evaluate their learning from practical experience. They will be able to identify skills gained and plan for future development within the digital sector. These capabilities support progression into a range of roles, including IT Support Technician, Junior Software Developer, Digital Operations Assistant, Systems Administrator and Project Support Officer.

 © Pearson Education Limited 2026

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Describe the key aspects of work experience in the digital sector

LO2 Report on the changing nature of work within the digital sector

LO3 Undertake a work experience placement within the digital sector

LO4 Review professional conduct and the learning experience following a placement in the digital sector.

## Essential Content

LO1 **Describe the key aspects of work experience in the digital sector**

*Work-based learning in the digital sector:*

Purpose and outcomes: work experience purpose, e.g. placement objectives, role-based tasks; employability development, e.g. workplace skills logs, competency tracking; academic application, e.g. applying project knowledge, applying coding skills; professional exposure, e.g. team meetings, workplace systems

Career and progression: career progression, e.g. progression pathways, role advancement; workplace learning, e.g. on-the-job training, supervised tasks; industry relevance, e.g. real project environments, current tools; practical experience, e.g. task completion, system interaction

Skills and readiness: skills development, e.g. technical skills, communication skills; professional readiness, e.g. workplace expectations, performance standards; experiential learning, e.g. reflective logs, task-based learning; career awareness, e.g. role understanding, sector awareness.

*Digital sector working environments:*

Organisation types: public sector organisations, e.g. local authority IT teams, NHS digital services; private sector companies, e.g. software firms, IT service providers; small and medium-sized enterprises (SMEs), e.g. small development teams, start-ups; large enterprises, e.g. corporate IT departments

Workplace models: remote working, e.g. home-based working, virtual teams; hybrid working, e.g. split office/home schedules; digital workplaces, e.g. cloud-based systems, collaboration platforms

Team structures: cross-functional teams, e.g. development teams, support teams; collaborative environments, e.g. agile teams, project groups; organisational structures, e.g. hierarchical teams, team-based structures.

*Roles and responsibilities:*

Entry and role definition: entry-level roles, e.g. IT support roles, junior developer roles; job responsibilities, e.g. user support, coding tasks; role expectations, e.g. task delivery, communication standards; job descriptions, e.g. role specifications, task requirements

Accountability and structure: reporting structures, e.g. line managers, team leads; accountability, e.g. task ownership, responsibility tracking; supervision structures, e.g. mentor support, supervisor oversight

Team contribution: teamwork roles, e.g. team member, support role; professional duties, e.g. completing assigned tasks, meeting deadlines; workplace contribution, e.g. project support, service delivery.

*Professional expectations:*

Workplace standards: workplace conduct, e.g. professional behaviour, appropriate communication; timekeeping, e.g. attendance records, punctuality logs; reliability, e.g. consistent task completion, meeting deadlines

Communication and compliance: communication practices, e.g. email communication, team meetings; confidentiality, e.g. data handling rules, access control; data protection, e.g. General Data Protection Regulations (GDPR) compliance, secure data use

Professional behaviour: organisational policies, e.g. HR policies, IT policies; ethical practice, e.g. responsible system use, professional integrity; workplace standards, e.g. codes of conduct, performance expectations; professional communication, e.g. written reports, verbal updates.

LO2 **Report on the changing nature of work within the digital sector**

*Technological change and innovation:*

Digital technologies: automation technologies, e.g. robotic process automation (RPA) tools, workflow automation; cloud-based systems, e.g. Software as a Service (SaaS) platforms, cloud infrastructure; artificial intelligence, e.g. AI-assisted tools, intelligent systems

Data and systems: data-driven decision-making, e.g. analytics dashboards, reporting tools; data analytics platforms, e.g. business intelligence (BI) tools, data processing systems; cybersecurity practices, e.g. access controls, security protocols

Collaboration tools: collaborative platforms, e.g. Microsoft Teams, Slack; digital transformation tools, e.g. process automation tools, workflow systems; emerging technologies, e.g. Internet of Things (IoT) systems, edge computing.

*Work patterns and structures:*

Working models: remote working, e.g. home-based working, virtual team platforms; hybrid working models, e.g. split-site schedules, blended working patterns; flexible working arrangements, e.g. flexible hours, adaptable schedules

Workplace organisation: distributed teams, e.g. multi-location teams, remote collaboration; global collaboration, e.g. international teams, cross-time-zone working; workforce diversity, e.g. cultural differences, generational differences, disability and neurodiversity considerations in digital workplaces; virtual workplaces, e.g. cloud-based environments, digital workspaces

Employment models: gig economy roles, e.g. freelance platforms, contract work; portfolio careers, e.g. multiple-role working, project-based roles; platform-based work, e.g. digital platforms, on-demand services.

*Skills demand and workforce development:*

Core skills: digital literacy, e.g. system usage, tool proficiency; technical specialisation, e.g. programming skills, system support; transferable skills, e.g. communication, teamwork

Learning and development: lifelong learning, e.g. ongoing training, continuous learning; upskilling initiatives, e.g. training programmes, workshops; reskilling programmes, e.g. career transition training, new skill development

Professional capability: professional certification, e.g. vendor certifications, industry qualifications; competency development, e.g. skills frameworks, capability matrices; adaptive skills, e.g. problem solving, adaptability.

*Organisational and sector impact:*

Workforce and roles: changing job roles, e.g. new digital roles, hybrid roles; emerging occupations, e.g. cloud engineers, data analysts; evolving professional standards, e.g. digital competency frameworks, role standards; job losses through AI, e.g. automation of support roles, AI-assisted redundancy; job losses through automation, e.g. process automation, replacement of repetitive tasks

Operational impact: productivity impact, e.g. output metrics, efficiency gains; operational efficiency, e.g. process optimisation, workflow improvement; service delivery models, e.g. digital services, platform-based delivery

Inclusion and change: inclusion initiatives, e.g. accessibility tools, inclusive design; accessibility considerations, e.g. assistive technologies, usability standards; workforce transformation, e.g. restructuring, digital adoption

Sector outcomes: competitive advantage, e.g. digital capability, innovation capacity; industry innovation, e.g. new services, new technologies; organisational change, e.g. transformation programmes, adoption strategies.

## LO3 **Undertake a work experience placement within the digital sector**

*Placement preparation:*

Planning and induction: placement objectives, e.g. learning objectives, role targets; role alignment, e.g. matching skills to tasks, role mapping; induction processes, e.g. workplace induction, onboarding sessions

Compliance setup: health and safety awareness, e.g. safety briefings, risk awareness; risk assessment, e.g. hazard identification, compliance forms; documentation requirements, e.g. placement agreements, policy documents

Workplace readiness: onboarding procedures, e.g. system access, account setup; organisational briefing, e.g. team introductions, system overview; personal goal setting, e.g. skills targets, development plans.

*Professional practice during placement:*

Workplace behaviour: workplace behaviours, e.g. punctuality, professional conduct; professional communication, e.g. emails, meetings; policy adherence, e.g. following procedures, IT policies

Task management: responsibility management, e.g. task ownership, accountability; task execution, e.g. completing work, meeting deadlines; time management, e.g. scheduling, prioritisation

Collaboration and compliance: teamwork participation, e.g. team tasks, group work; organisational conduct, e.g. reporting issues, following processes; data protection compliance, e.g. secure data handling, GDPR.

*Contribution and participation:*

Workplace contribution: task engagement, e.g. assigned tasks, support activities; project contribution, e.g. assisting delivery, support roles; workflow participation, e.g. process execution, task co-ordination

Application of knowledge: application of academic knowledge, e.g. coding skills, system concepts; digital tool usage, e.g. software tools, platforms; problem solving activities, e.g. issue resolution, troubleshooting

Initiative and performance: initiative taking, e.g. proactive tasks, improvement suggestions; performance contribution, e.g. meeting key performance indicators (KPIs), task completion.

*Performance awareness:*

Feedback and review: feedback mechanisms, e.g. supervisor feedback, review meetings; performance evaluation, e.g. performance reviews, assessment reports

Self-assessment, e.g. reflective logs, self-evaluation; strengths identification, e.g. recognising skills, strengths; development areas, e.g. skill gaps, improvement needs

Reflection and improvement: reflection practices, e.g. learning logs, journals; learning progression, e.g. skill development, task progression; improvement planning, e.g. action plans, development targets.

LO4 **Review professional conduct and the learning experience following a placement in the digital sector**

*Professional conduct:*

Core behaviours: ethical behaviour, e.g. adherence to code of conduct, integrity in task delivery; reliability, e.g. attendance records, deadline adherence; accountability, e.g. task ownership logs, responsibility tracking

Workplace standards: communication effectiveness, e.g. structured emails, meeting updates; professional standards, e.g. role-based expectations, performance criteria; workplace etiquette, e.g. professional interactions, team conduct

Compliance and professionalism: confidentiality practices, e.g. secure data handling, access permissions; digital professionalism, e.g. appropriate system use, communication protocols; role responsibility, e.g. task completion records, procedural compliance.

*Reflective learning:*

Reflection practices: learning reflection, e.g. reflective journals, structured logs; self evaluation, e.g. strengths/weakness analysis, performance reviews; reflective practice, e.g. task reviews, outcome analysis

Skills and development: skills recognition, e.g. competency mapping, skills logs; knowledge application, e.g. applied tasks, project outputs; development planning, e.g. action plans, improvement targets

Progress tracking: learning outcomes, e.g. achieved targets, completed objectives; progress tracking, e.g. review logs, progress records; reflective analysis, e.g. performance summaries, learning evaluations.

*Evaluation of experience:*

Experience review: expectation alignment, e.g. planned vs actual outcomes, objective comparison; experience evaluation, e.g. placement review reports, feedback summaries; employability readiness, e.g. role readiness evidence, capability assessment

Performance and contribution: skills assessment, e.g. competency evaluation, task performance data; performance review, e.g. supervisor evaluations, appraisal records; workplace contribution, e.g. project outputs, service contributions

Outcomes and value: objective achievement, e.g. completed deliverables, target completion; feedback integration, e.g. applied feedback, improvement actions; value assessment, e.g. impact evidence, contribution outcomes.

*Career development planning:*

Future direction: skills gap identification, e.g. gap analysis reports, competency gaps; future learning needs, e.g. training plans, certification pathways; career pathways, e.g. role progression maps, job role targets

Planning and goals: progression planning, e.g. development plans, next-step planning; goal setting, e.g. SMART goals, performance targets; professional development plans, e.g. Continuing Professional Development (CPD) plans, learning frameworks

Opportunities and alignment: training opportunities, e.g. courses, structured training; career objectives, e.g. target roles, progression goals; industry alignment, e.g. skills-role mapping, employability alignment; capability development, e.g. upskilling activities, experience building; professional currency, developments, evolving practice.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Describe the key aspects of work experience in the digital sector</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate, using a selection of real-world examples, how technology and innovation have changed the nature of work and work experience for individuals in the digital sector.</td></tr>
<tr><td>P1 Describe, using a selection of real-world examples, the key aspects of work-based learning in different working environments in the digital sector.<br><br>P2 Explain, using a selection of real-world examples, the roles, responsibilities and professional expectations for individuals carrying out work experience in the digital sector.</td><td>M1 Analyse, using a selection of real-world examples, the interrelationship between workplace learning and organisational expectations for individuals carrying out work experience in the digital sector.</td></tr>
<tr><td colspan="2">LO2 Report on the changing nature of work within the digital sector</td></tr>
<tr><td>P3 Examine, using a selection of real-world examples, how technology and innovation have changed the work patterns and skills demand within the digital sector.<br><br>P4 Report, using a selection of real-world examples, on how the changing nature of work within the digital sector has had an impact on organisations and the sector itself.</td><td>M2 Analyse, using a selection of real-world examples, the impact of recent technological changes in the nature of work, roles and working practices within the digital sector.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Undertake a work experience placement within the digital sector | | **LO3 and LO4**<br>**D2** Critically evaluate how the overall learning experience, professional conduct and performance within the work experience placement contributes to future career development within the digital sector. |
| **P5** Carry out preparations to undertake a work experience placement within the digital sector.<br>**P6** Demonstrate professional practice, workplace contribution and performance awareness within a work experience placement within the digital sector. | **M3** Analyse the effectiveness of own performance and contribution during the work experience placement within the digital sector. | |
| **LO4** Review professional conduct and the learning experience following a placement in the digital sector | | |
| **P7** Review, using reflective learning and self evaluation, the professional conduct and the learning experience demonstrated during the work experience placement in the digital sector.<br>**P8** Develop, using the results of reflective learning, self-evaluation and the work experience review, a career development plan, for continued employment in the digital sector. | **M4** Assess the learning gained from the work experience placement in relation to professional practice within the digital sector. | |

## Recommended Resources

### Textbooks

Barr, M. (ed.) (2024) *Approaches to Work-Based Learning in Higher Education: Improving Graduate Employability*. 1st edn. London: Routledge.

Barr, M. (2022) *Learning for the Future Workplace: Digital Skills and Employability*. London: Routledge.

Billett, S., Olesen, H.S. and Filletz, L. (eds.) (2023) *Sustaining Employability through Work-Life Learning: Practices and Policies*. Cham: Springer.

Ng, B. (2023) *Graduate Employability and Workplace-Based Learning Development: Insights from Sociocultural Perspectives*. Singapore: Springer.

Poutanen, S., Kovalainen, A. and Rouvinen, P. (2024) *Digital Work and the Platform Economy: Understanding Tasks, Skills and Capabilities in the New Era*. London: Routledge.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| higherin.com/career-guides/it/it-placements | Higherin<br>Guide to IT placements describing roles, responsibilities and skills developed during digital work placements<br>(Article) |
| www.prospects.ac.uk/jobs-and-work-experience/work-experience-and-internships/work-placements/ | Prospects<br>Work placements overview outlining structure, tasks, skills and outcomes of placement experience<br>(General reference) |
| www.techtimes.com/articles/314266/20260124/digital-workplace-revolution-why-hybrid-work-real-remote-work-future.htm | TechTimes<br>Article explaining how hybrid and remote work models are shaping modern digital workplaces<br>(Article) |
| post.parliament.uk/research-briefings/post-pb-0049/ | UK Parliament POST<br>Research briefing on remote and hybrid working definitions, trends and organisational impact<br>(Report) |

 © Pearson Education Limited 2026

| | |
|---|---|
| www.hilarispublisher.com/open-access/the-future-of-work-adapting-to-remote-and-hybrid-work-models-in-a-digital-age.pdf | Journal of Business & Financial Affairs<br>Exploration of the future of work including benefits, challenges and technologies supporting remote and hybrid models<br>(PDF Download) |
| www.bcs.org/articles-opinion-and-research/digital-skills-demand-to-2030-skills-england-s-latest-assessment/ | BCS – The Chartered Institute for IT<br>Report summarising projected digital skills demand and growth in technology roles<br>(Article) |
| nationalskillscoalition.org/news/press-releases/new-report-92-of-jobs-require-digital-skills-one-third-of-workers-have-low-or-no-digital-skills-due-to-historic-underinvestment-structural-inequities/ | National Skills Coalition<br>Report highlighting the importance of digital skills across industries and the impact of skills gaps<br>(Report) |
| assets.ctfassets.net/2pudprfttvy6/3ELFKTA8GBPBuRkNrOzmpS/24fc7ec2372d0adb96965340069f705c/Global_Skills_Report_2025.pdf | Coursera<br>Global Skills Report<br>Analysis of global trends in digital skills, learning demand and workforce development<br>(Report) |
| grimsby.ac.uk/wp-content/uploads/2025/09/Reflective-Writing-E-Guide.pdf | Grimsby Institute<br>Guide to reflective practice and reflective writing for analysing learning from work experience<br>(Guide) |
| bpb-ap-se2.wpmucdn.com/blogs.auckland.ac.nz/dist/3/316/files/2024/09/Reflective-writing-guide-for-students.pdf | University of Auckland<br>Reflective practice guide introducing models and approaches for analysing workplace learning<br>(Tutorial) |

**Journals and articles**

Kraus, S. *et al*. (2023) 'The future of work: How innovation and digitalization re-shape the workplace', Journal of Innovation and Knowledge, 8(4), 100438. Available at: doi.org/10.1016/j.jik.2023.100438.

Mabungela, M. and Mtikib, V. (2024) 'Accelerating Graduate Employability through Work-Integrated Learning', *Research in Social Sciences and Technology*, 9(1), pp. 291–304. Available at: doi.org/10.46303/ressat.2024.17.

Machost, H. and Stains M. (2023) 'Reflective Practices in Education: A Primer for Practitioners', *CBE-Life Sciences Education*, 22(2), es2. Available at: doi.org/10.1187/cbe.22-07-0148.

Rienties, B. *et al*. (2023) 'Online work-based learning: A systematic literature review', *International Review of Education*, 69, pp. 551–570. Available at: doi.org/10.1007/s11159-023-10008-y.

Willcocks, L. (2024) 'Automation, digitalization and the future of work: A critical review', *Journal of Electronic Business and Digital Economics*, 3(2), pp. 184–199. Available at: doi.org/10.1108/JEBDE-09-2023-0018.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*.

 © Pearson Education Limited 2026

# Unit 537: Fundamentals of Blockchain Technology

**Unit code:** R/652/3323

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Blockchain technology has become an important part of modern digital systems, supporting secure data exchange, decentralised applications and new models of trust between organisations. While it is widely associated with cryptocurrencies, its use now extends into areas such as finance, supply chain management, digital identity and public services. Understanding how blockchain operates within these contexts is increasingly relevant for those working in computing and digital technology roles.

The aim of this unit is to develop students' understanding of blockchain technologies and their application within distributed digital systems. It focuses on the principles that underpin blockchain operation, the tools and techniques used to build and deploy blockchain solutions, and the considerations that influence their design. Students will explore how blockchain models can be developed and implemented for identified scenarios, enabling them to make informed decisions about appropriate architectures, platforms and development approaches.

The unit covers a structured range of topics that reflect the development life cycle of blockchain solutions. It begins with the fundamental concepts that underpin blockchain technologies, including distributed ledgers, consensus mechanisms and cryptographic techniques. It then moves on to the tools, platforms and environments used in development and deployment. Students will design a blockchain solution model for a given scenario before progressing to the implementation of a simple distributed blockchain, allowing them to connect theoretical knowledge with practical application.

On successful completion of this unit, students will be able to explain how blockchain systems operate, evaluate the suitability of tools and platforms, and design and implement basic distributed solutions. These skills support progression into roles such as Blockchain Developer, Software Engineer, Systems Analyst, Cybersecurity Specialist and Technology Consultant, where an understanding of decentralised systems and secure data management is increasingly required.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core principles of blockchain technologies

LO2 Review the tools and techniques used to build and deploy blockchain solutions

LO3 Design a blockchain solution model for an identified scenario

LO4 Implement a simple distributed blockchain model for an identified scenario.

## Essential Content

### LO1 **Explain the core principles of blockchain technologies**

*Decentralisation and distributed systems:*

Network structure: peer-to-peer networks, e.g. node-to-node communication, gossip protocols; distributed systems, e.g. distributed nodes, replicated services; absence of central authority, e.g. no central server, decentralised control

System behaviour: node independence, e.g. autonomous nodes, independent validation; redundancy, e.g. replicated nodes, duplicated data; fault tolerance, e.g. node failure handling, network recovery; system resilience, e.g. failover, continued operation

Trust and scaling: trust distribution, e.g. trustless verification, distributed trust; network scalability, e.g. node scaling, distributed load; data distribution, e.g. replicated ledgers, peer distribution.

*Distributed ledgers:*

Ledger structure: Shared ledger concept, e.g. replicated ledgers, distributed state; append-only structures, e.g. block chaining, immutable records

Integrity and transparency: immutability, e.g. hash-linked blocks, tamper-evident records; transparency, e.g. public transaction data, audit visibility; auditability, e.g. transaction traceability, verifiable history

Ledger operations: transaction recording, e.g. transaction entries, block inclusion; ledger synchronisation, e.g. node syncing, block propagation; ledger integrity, e.g. hash validation, chain verification.

*Consensus fundamentals:*

Consensus processes: consensus mechanisms, e.g. Proof of Work, Proof of Stake; transaction validation, e.g. signature checking, transaction verification; block confirmation, e.g. confirmation depth, block finalisation

Protocol behaviour: agreement protocols, e.g. Nakamoto consensus, byzantine fault-tolerant (BFT) protocols; fault tolerance models, e.g. Byzantine fault tolerance, node fault handling; validator selection, e.g. mining selection, staking selection

Performance factors: consensus efficiency, e.g. block time, throughput; consensus incentives, e.g. mining rewards, staking rewards.

*Cryptography and data integrity:*

Hashing and keys: cryptographic hashing, e.g. SHA-256, Keccak; digital signatures, e.g. Elliptic Curve Digital Signature Algorithm (ECDSA) signatures, verification; asymmetric cryptography, e.g. public-private key pairs, key exchange

Integrity controls: collision resistance, e.g. hash security, collision avoidance; tamper resistance, e.g. hash chaining, block protection; integrity verification, e.g. hash validation, signature checks.

*Merkle tree structures:*

Tree structure: Merkle trees, e.g. binary hash trees, hierarchical hashing; Merkle roots, e.g. root hash, block header hash

Verification: transaction verification, e.g. Merkle proofs, inclusion checks; data validation, e.g. integrity checks, proof verification

Efficiency: efficient verification, e.g. logarithmic verification, reduced data checks; block composition, e.g. transaction batching, block structure.

LO2 **Review the tools and techniques used to build and deploy blockchain solutions**

*Blockchain development platforms:*

Platform types: distributed ledger platforms, e.g. Ethereum, Hyperledger Fabric; permissioned blockchains, e.g. enterprise networks, restricted access systems; permissionless blockchains, e.g. public networks, open participation

Platform features: platform architecture, e.g. node types, modular design; consensus support, e.g. mining, validation; extensibility, e.g. smart contract support

Scaling and interoperability: platform scalability, e.g. sharding, Layer 2 scaling; interoperability frameworks, e.g. cross-chain communication, integration layers; scaling strategies, e.g. node scaling.

*Smart contract technologies:*

Core concepts: smart contracts, e.g. on-chain code, automated agreements; execution environments, e.g. virtual machines, runtime environments; contract deployment, e.g. deployment transactions, network publishing

Contract behaviour: life cycle management, e.g. deployment, updates; deterministic execution, e.g. consistent outputs; validation, e.g. testing, verification

System interaction: state management, e.g. storage variables; transaction logic, e.g. function execution; contract interactions, e.g. inter-contract calls, event handling.

*Development tools and environments:*

Development environments: integrated development environments, e.g. contract IDEs; local blockchain networks, e.g. test networks, local nodes; build environments, e.g. compilers, build tools

Testing and debugging: testing frameworks, e.g. unit testing, contract testing; debugging tools, e.g. transaction tracing, error logs

Development workflow: development libraries, e.g. software development kits (SDKs); version control systems, e.g. repositories; deployment pipelines, e.g. continuous integration/continuous development (CI/CD) pipelines

Tools: orchestration tools, e.g. container orchestration; version management, e.g. protocol updates.

*Deployment techniques:*

Infrastructure setup: node configuration, e.g. full nodes, validator nodes; network topology setup, e.g. peer networks, node clusters; cloud deployment, e.g. hosted environments; on-premises deployment, e.g. enterprise systems

Environment management: containerisation, e.g. container environments; configuration management, e.g. configuration files, environment variables.

*Operational and suitability considerations:*

System suitability: system requirements, e.g. throughput needs, latency constraints; platform selection, e.g. public vs private blockchain; interoperability needs, e.g. integration requirements

Performance and efficiency: transaction throughput, e.g. transactions per second (TPS); latency, e.g. confirmation delay; resource utilisation, e.g. compute usage

Constraints and trade-offs: cost considerations, e.g. transaction costs; security considerations, e.g. vulnerabilities; deployment trade-offs, e.g. decentralisation vs performance

*Real-world examples:*

Examples of blockchain platforms e.g.

Platform examples: real-world platforms, e.g. Bitcoin, Ethereum; enterprise platforms, e.g. Hyperledger Fabric, Corda; high-performance platforms, e.g. Solana, Avalanche; interoperability platforms, e.g. Polkadot, Cosmos; commercial platforms, e.g. Ripple, IBM Blockchain Platform.

LO3 **Design a blockchain solution model for an identified scenario**

*Application scenarios and requirements:*

Use cases: industry use cases, e.g. supply chain tracking, digital identity; organisational requirements, e.g. data sharing needs, trust relationships; business processes, e.g. transaction workflows, process automation

System requirements: functional specifications, e.g. transaction validation, business rules; regulatory requirements, e.g. compliance constraints, data governance; system constraints, e.g. throughput limits, latency targets.

*Blockchain architecture models:*

Network types: public blockchains, e.g. open networks, public nodes; private blockchains, e.g. restricted networks, enterprise nodes; consortium blockchains, e.g. multi-organisation networks, shared governance

Architecture structure: node roles, e.g. full nodes, validator nodes; permission models, e.g. access control, identity-based permissions; network topology, e.g. peer networks, node clusters

Governance and control: decentralised architecture, e.g. distributed control, protocol rules; governance models, e.g. on-chain governance, consortium governance; access control models, e.g. permissioned access, role control.

*Smart contract and system design:*

Logic and workflows: business logic modelling, e.g. rule execution, automated processes; transaction workflows, e.g. transaction life cycle, validation flow; process automation, e.g. contract-triggered actions, event handling

Data and state: state management, e.g. contract state, storage variables; data structures, e.g. key–value storage, structured data; validation rules, e.g. input validation, transaction checks

System interaction: contract interactions, e.g. function calls, inter-contract communication; system integration, e.g. application programming interface (API) integration, external systems; application interfaces, e.g. user interfaces, dashboards.

*Scalability, interoperability and performance considerations:*

Performance factors: transaction throughput, e.g. TPS, processing capacity; system latency, e.g. confirmation delay, network delay; storage requirements, e.g. ledger size, data growth

Interoperability: interoperability challenges, e.g. cross-chain communication, integration issues; cross-chain communication, e.g. bridging mechanisms, data exchange

Scalability models: scalability models, e.g. Layer 2 scaling, sharding; network congestion, e.g. transaction backlog, processing limits

Architecture trade-offs: architecture trade-offs, e.g. scalability vs security, performance vs decentralisation; performance constraints, e.g. bottlenecks, throughput limits.

*Design limitations and constraints:*

Technical constraints: energy consumption, e.g. mining energy use, processing demand; scalability limitations, e.g. node limits, throughput constraints; performance bottlenecks, e.g. processing delays, congestion

Governance and compliance: governance limitations, e.g. co-ordination delays, decision overhead; regulatory constraints, e.g. compliance requirements, legal frameworks; security risks, e.g. attack vectors, vulnerabilities; current developments, knowledge currency, evolving practice

Operational considerations: operational complexity, e.g. network management, deployment complexity; cost considerations, e.g. infrastructure cost, transaction fees; system dependencies, e.g. external integrations, platform dependencies.

LO4 **Implement a simple distributed blockchain model for an identified scenario**

*Distributed blockchain implementation concepts:*

Core processes: node participation, e.g. peer nodes, validator nodes; peer-to-peer communication, e.g. message propagation, network messaging; transaction creation, e.g. transaction inputs, signed transactions

Blockchain structure: block formation, e.g. transaction grouping, block creation; blockchain structure, e.g. block chain linking, ordered blocks; transaction broadcasting, e.g. propagation to peers, network distribution

Consensus and replication: distributed consensus, e.g. validation agreement, consensus execution; ledger replication, e.g. replicated ledgers, node synchronisation; decentralised validation, e.g. peer validation, transaction verification.

*System configuration and setup:*

Network configuration: network parameters, e.g. block size, timing parameters; consensus configuration, e.g. validation rules, consensus selection; node roles, e.g. validators, full nodes

Environment setup: deployment environments, e.g. local networks, virtual environments; configuration files, e.g. node configs, runtime settings; initialisation processes, e.g. genesis block setup, network start

Access and control: permission settings, e.g. access control, identity configuration; network topology setup, e.g. peer connections, node structure; infrastructure configuration, e.g. system setup, environment variables.

*Blockchain operation and behaviour:*

Transaction processing: transaction processing, e.g. validation checks, execution flow; block validation, e.g. block verification, consensus approval; consensus execution, e.g. agreement cycles, validation rounds

System behaviour: fault handling, e.g. node failure recovery, error handling; network consistency, e.g. consistent state, synchronised nodes; state transitions, e.g. ledger updates, account state changes

Network operations: block propagation, e.g. block distribution, peer updates; synchronisation processes, e.g. node syncing, state updates; transaction life cycle, e.g. creation to confirmation.

*Verification, integrity and performance checks:*

Validation and integrity: transaction validation, e.g. signature checks, rule validation; hash verification, e.g. block hashing, chain validation; consensus confirmation, e.g. block finalisation, confirmation depth

Monitoring and diagnostics: performance monitoring, e.g. throughput metrics, latency tracking; system diagnostics, e.g. logs, error detection; integrity checks, e.g. ledger consistency, validation checks

Performance evaluation: throughput measurement, e.g. transactions per second; verification procedures, e.g. validation steps, audit checks; auditing processes, e.g. traceability, validation records.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explain the core principles of blockchain technologies | | |
| **P1** Explain, using a selection of real-world examples, the core principles of blockchain technologies including decentralisation, distributed ledgers and consensus.<br>**P2** Review, using a selection of real-world examples, how cryptography and Merkle tree structures are used within blockchain technologies. | **M1** Analyse, using a selection of real-world examples, how the core principles of blockchain technologies support the implementation of secure, distributed ledgers. | **D1** Critically evaluate, using a selection of real-world examples, evaluate the extent to which blockchain principles address challenges in data integrity, trust and system resilience. |
| **LO2** Review the tools and techniques used to build and deploy blockchain solutions | | |
| **P3** Review, using a selection of real-world examples, the tools and techniques used to build blockchain and smart contract solutions.<br>**P4** Explore, using a selection of real-world examples, the tools and techniques used to deploy blockchain and smart contract solutions. | **M2** Assess, using a selection of real-world examples, the effectiveness of blockchain tools and techniques used to build and deploy blockchain solutions. | **D2** Critically evaluate, using a selection of real-world examples, the suitability of blockchain tools and deployment techniques for different implementation contexts. |

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Design a blockchain solution model for an identified scenario | | **LO3 and LO4**<br>**D3** Justify the decisions made in the design and implementation of the simple distributed blockchain model in relation to how comprehensively they meet the requirements of the identified scenario. |
| **P5** Design a blockchain solution model for an identified scenario.<br>**P6** Review the design of a blockchain solution model with a range of identified stakeholders to identify strengths and limitations and how well it meets the requirements of an identified scenario. | **M3** Optimise, using the results of the stakeholder feedback, the design of a blockchain solution model so that it fully meets the requirements of an identified scenario. | |
| **LO4** Implement a simple distributed blockchain model for an identified scenario | | |
| **P7** Implement a simple distributed blockchain model for an identified scenario.<br>**P8** Review the implementation of the simple distributed blockchain model using monitoring and performance checks to identify areas for improvement. | **M4** Evaluate the effectiveness of the implemented simple distributed blockchain in addressing the requirements of the identified scenario. | |

## Recommended Resources

### Textbooks

Coulouris, G. *et al*. (2011) *Distributed Systems: Concepts and Design*. 5th edn. Harlow: Addison-Wesley.

Jamader, A.R., Selvam, M. and Acharya, B.R. (2024) *Blockchain and Cryptocurrency: Management Systems and Technology Challenges*. Boca Raton, FL: CRC Press.

Pandey, R., Goundar, S. and Fatima, S. (2023) *Distributed Computing to Blockchain: Architecture, Technology and Applications*. Cambridge, MA: Academic Press.

Raj, P., Saini, K. and Surianarayanan, C. (2021) *Blockchain Technology and Applications*. Boca Raton, FL: CRC Press.

Vyas, S. *et al*. (2022) *Blockchain Technology: Exploring Opportunities, Challenges, and Applications*. Boca Raton, FL: CRC Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/blockchain-technology-introduction/ | GeeksforGeeks<br>Overview of Blockchain Technology including core concepts, decentralisation and distributed ledgers<br>(General reference) |
| www.ibm.com/topics/blockchain | IBM<br>Explanation of blockchain fundamentals, components and how distributed ledgers operate<br>(General reference) |
| chain.link/article/what-are-merkle-trees | Chainlink<br>Explanation of Merkle trees, structure and verification in blockchain systems<br>(General reference) |
| ethereum.org/developers/docs/consensus-mechanisms/ | Ethereum.org<br>Overview of consensus mechanisms and how distributed networks achieve agreement<br>(General reference) |

| | |
|---|---|
| www.geeksforgeeks.org/computer-networks/cryptographic-consensus-mechanisms-in-blockchain/ | GeeksforGeeks<br>Article explaining key consensus algorithms such as Proof of Work and Proof of Stake<br>(Article) |
| ethereum.org/developers/docs/ | Ethereum.org<br>Developer documentation covering blockchain tools, environments and deployment processes<br>(Documentation Resource) |
| www.codebridge.tech/articles/blockchain-architecture-explained-comprehensive-guide | CodeBridge<br>Explanation of blockchain architecture, system components and design considerations<br>(General reference) |
| serokell.io/blog/scalability-blockchain | Serokell<br>Article exploring scalability challenges and performance considerations in blockchain systems<br>(General reference) |
| crustlab.com/blog/blockchain-interoperability/ | CrustLab<br>Explanation of interoperability and cross-chain communication in blockchain solutions<br>(Blog) |
| www.blog.brightcoding.dev/2025/02/27/creating-a-blockchain-from-scratch-a-comprehensive-tutorial-for-developers/ | BrightCoding<br>Practical tutorial on building a blockchain from scratch and evaluating its operation<br>(Tutorial) |

## Journals and articles

Hussein, Z., Salama, M.A. and El-Rahman, S.A. (2023) 'Evolution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms', *Cybersecurity*, 6(30). Available at: doi.org/10.1186/s42400-023-00163-y.

Rao, I.S. *et al*. (2024) 'Scalability of blockchain: a comprehensive review and future research direction', Cluster Computing, 27, pp. 5547–5570. Available at: doi.org/10.1007/s10586-023-04257-7.

Shen, Z., Qu, Q. and Chen, X-B. (2025) 'Blockchain Consensus Mechanisms: A Comprehensive Review and Performance Analysis Framework', *Electronics,* 14(17). Available at: doi.org/10.3390/electronics14173567.

Zhou, H., Milani Fard, A. and Makanju, A. (2022) 'The State of Ethereum Smart Contracts Security: Vulnerabilities, Countermeasures, and Tool Support', *Journal of Cybersecurity and Privacy*, 2(2), pp. 358–378. Available at: doi.org/10.3390/jcp2020019.

## Links

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*

*Unit 538: Applied Cloud Cryptography*

# Unit 538: Applied Cloud Cryptography

**Unit code:** T/652/3324

**Unit level:** 5

**Credit value:** 15

---

## Introduction

As cloud computing continues to underpin modern digital services, the need to protect data, systems and communications has become a core requirement across all sectors. Cloud platforms introduce specific security challenges linked to distributed architectures, shared infrastructure and remote access, making traditional approaches to protection insufficient on their own. Cryptography provides the mechanisms that enable secure processing and trusted exchange of information in these environments, and is fundamental to maintaining confidentiality, integrity and trust in cloud-based systems.

The aim of this unit is to develop students' ability to apply cryptographic principles within cloud environments. It focuses on how encryption, key management and trust models are used to secure cloud data and services, and how these approaches support wider security objectives. Students will explore how cryptography is integrated into real cloud architectures, enabling them to make informed decisions about appropriate techniques and controls in applied scenarios.

The unit covers a range of topics required to support secure cloud operations. This includes core cryptographic principles, encryption models and their application within cloud platforms, alongside the use of symmetric and asymmetric mechanisms. Students will examine secure key management, including life cycle control and hardware-backed protection, before applying cryptographic protocols to protect an identified cloud data workflow. The content is designed to link theoretical understanding with practical implementation and evaluation.

On successful completion of this unit, students will be able to analyse and apply cryptographic techniques to secure cloud-based systems, select appropriate encryption methods, and manage cryptographic keys effectively within operational environments. These skills support progression into roles such as Cloud Security Analyst, Cyber Security Engineer, Cloud Solutions Architect, Security Operations Specialist and DevSecOps Engineer.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explain the core principles of cloud cryptography

LO2 Review symmetric and asymmetric encryption mechanisms in cloud environments

LO3 Discuss the secure key management approaches for cloud-based systems

LO4 Implement cryptographic protocols to secure a cloud data workflow for an identified scenario.

## Essential Content

LO1 **Explain the core principles of cloud cryptography**

*Cryptographic principles in cloud environments:*

Core security principles: confidentiality, e.g. AES-256 encryption, data access control lists; integrity, e.g. SHA-256 hashing, Hash-based Message Authentication Code (HMAC) verification; availability, e.g. multi-region replication, load-balanced failover

Security foundations: encryption concepts, e.g. symmetric ciphers, asymmetric key exchange; cryptographic strength, e.g. 256-bit keys, algorithm resistance; threat models, e.g. man-in-the-middle attacks, insider threats

Trust and assurance: trust assumptions, e.g. trust boundaries, zero-trust models; data protection, e.g. encrypted storage volumes, encrypted object storage; secure communication, e.g. Transport Layer Security (TLS) 1.3, Hypertext Transfer Protocol Secure (HTTPS) endpoints; risk mitigation, e.g. encryption controls, access enforcement.

*Cloud trust and shared responsibility models:*

Responsibility layers: cloud service provider responsibilities, e.g. physical data centre security, hypervisor security; customer responsibilities, e.g. application encryption, identity management; shared responsibility model, e.g. split security roles, layered controls

Service model security: infrastructure security, e.g. virtual machine (VM) isolation, network segmentation; platform security, e.g. managed runtime security, Application Programming Interface (API) protection; application security, e.g. secure coding, input validation

Trust and governance: data ownership, e.g. customer-managed data, access ownership; trust boundaries, e.g. tenant isolation, network segmentation; service models, e.g. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS); accountability, e.g. audit logs, monitoring systems; security governance, e.g. policy enforcement, compliance controls.

*Cryptographic models applied to cloud systems:*

Data protection models: data-at-rest encryption, e.g. disk encryption, database encryption; data-in-transit encryption, e.g. TLS, secure sockets; data-in-use protection, e.g. secure enclaves, confidential computing

Isolation and layering: isolation mechanisms, e.g. VM isolation, container isolation; multi-tenancy protection, e.g. logical tenancy separation, access isolation; encryption layers, e.g. transport encryption, application-layer encryption

System protection: secure storage, e.g. encrypted storage services, key-integrated storage; secure transmission, e.g. encrypted APIs, secure endpoints; cryptographic assurance, e.g. validated encryption controls, compliance checks; data confidentiality controls, e.g. identity and access management (IAM) policies, encryption enforcement.

LO2 **Review symmetric and asymmetric encryption mechanisms in cloud environments**

*Symmetric encryption in cloud systems:*

Key usage: Shared secret keys, e.g. symmetric key exchange using secure channels; symmetric algorithms, e.g. AES-128, ChaCha20; encryption performance, e.g. high-throughput encryption, low-latency processing

Data protection: bulk data encryption, e.g. storage encryption, database encryption; data-at-rest encryption, e.g. disk encryption, object storage encryption; storage protection, e.g. encrypted volumes, secure storage services

Operational considerations: key distribution challenges, e.g. secure key exchange, key storage; scalability considerations, e.g. large-scale encryption, distributed workloads; encryption efficiency, e.g. hardware acceleration, optimised processing.

*Asymmetric encryption in cloud systems:*

Key mechanisms: Public key cryptography, e.g. Rivest–Shamir–Adleman (RSA) keys, elliptic curve cryptography (ECC) keys; private key cryptography, e.g. private key storage, secure key handling; key pair generation, e.g. key generation algorithms, entropy sources

Communication security: key exchange mechanisms, e.g. Diffie-Hellman, ECDH; secure communication, e.g. Transport Layer Security (TLS) handshake, encrypted sessions; data-in-transit encryption, e.g. Hypertext Transfer Protocol Secure (HTTPS), secure application programming interfaces (APIs)

Identity and trust: identity verification, e.g. digital certificates, certificate chains; authentication processes, e.g. certificate-based authentication, key validation; trust establishment, e.g. certificate authorities, Public Key Infrastructure (PKI) models.

*Cloud cryptographic algorithms:*

Encryption algorithms: Advanced Encryption Standard (AES), e.g. AES-256; Elliptic Curve Cryptography, e.g. ECC keys, lightweight encryption

Hashing and integrity: Secure Hash Algorithm (SHA) 256, e.g. SHA-256 hashing; SHA-3, e.g. SHA-3 hashing; Hash-based Message Authentication Code (HMAC) authentication, e.g. keyed hashing, integrity validation

Signature and key exchange: RSA, e.g. RSA encryption, key exchange; Elliptic Curve Digital Signature Algorithm, e.g. ECDSA signatures; Diffie–Hellman (DH) key exchange, e.g. DH key exchange; Elliptic Curve Diffie-Hellman (ECDH), e.g. ECDH secure exchange.

*Algorithm selection in cloud security:*

Security factors: key length considerations, e.g. 128-bit vs 256-bit keys; security strength, e.g. resistance to brute-force, algorithm resilience; compliance requirements, e.g. regulatory standards, approved algorithms

Performance trade-offs: performance trade-offs, e.g. speed vs security; processing overhead, e.g. CPU usage, encryption latency; algorithm efficiency, e.g. lightweight vs high-security algorithms

Deployment context: scalability needs, e.g. high-volume systems, distributed workloads; use-case suitability, e.g. storage encryption, API security; implementation constraints, e.g. hardware limits, system compatibility.

LO3 **Discuss the secure key management approaches for cloud-based systems**

*Key management services in the cloud:*

Key storage and control: Centralised key storage, e.g. cloud key management systems (KMS) storage, encrypted key vaults; key management services, e.g. AWS KMS, Azure Key Vault; access control policies, e.g. identity and access management (IAM) policies, role-based access control

Monitoring and governance: key usage monitoring, e.g. access logs, usage tracking; audit logging, e.g. CloudTrail logs, key access logs; policy enforcement, e.g. access policies, key usage rules

Integration and distribution: service integration, e.g. API integration, encryption services; key distribution, e.g. secure key delivery, envelope encryption; service scalability, e.g. multi-region keys, distributed access.

*Hardware security mechanisms:*

Secure hardware: Hardware security modules (HSM), e.g. cloud HSM, dedicated key hardware; secure enclaves, e.g. Intel SGX, confidential computing; trusted execution environments, e.g. isolated runtime environments, secure processing

Protection features: tamper resistance, e.g. physical protection, intrusion detection; hardware-based encryption, e.g. cryptographic acceleration, secure key storage; secure boot processes, e.g. trusted startup, firmware validation

Isolation and performance: hardware isolation, e.g. secure partitions, enclave isolation; cryptographic acceleration, e.g. hardware crypto engines, encryption offload; secure processing unit, e.g. protected computation, isolated execution.

*Key life cycle management:*

Life cycle stages: Key generation, e.g. entropy-based generation, secure random number generator (RNG); key storage, e.g. encrypted storage, HSM-backed storage; key distribution, e.g. secure key exchange, envelope encryption

Key operations: key rotation, e.g. scheduled rotation, automatic rotation; key revocation, e.g. key invalidation, access removal; key expiration, e.g. life cycle expiry, time-based policies

Recovery and control: key backup, e.g. secure backups, redundant storage; key recovery, e.g. restore processes, recovery keys; key destruction, e.g. secure deletion, irreversible removal

Governance: life cycle policies, e.g. rotation policies, retention rules; compliance requirements and updates, e.g. regulatory standards, audit controls; governance controls, e.g. policy enforcement, monitoring.

LO4 **Implement cryptographic protocols to secure a cloud data workflow for an identified scenario**

*Cryptographic protocols for cloud data protection:*

Protocol types: Transport layer security, e.g. TLS 1.3 encryption, HTTPS connections; secure communication protocols, e.g. secure APIs, encrypted channels; authentication protocols, e.g. certificate-based authentication, token-based authentication

Session and validation: session management, e.g. session keys, key renegotiation; handshake protocols, e.g. TLS handshake, key exchange negotiation; protocol negotiation, e.g. cipher suite selection, protocol versions; certificate validation, e.g. certificate chains, trust verification

Integrity and protection: integrity protection mechanisms, e.g. HMAC validation, message authentication; data protection protocols, e.g. encrypted data transfer, API security; network security protocols, e.g. secure communication layers, protected endpoints.

*Encryption and hashing within cloud workflows:*

Data protection processes: data encryption processes, e.g. AES encryption, encrypted storage; hashing algorithms, e.g. SHA-256, SHA-3; key derivation functions, e.g. PBKDF2, HKDF

Authentication and integrity: message authentication codes, e.g. HMAC validation, integrity checks; integrity verification, e.g. hash comparison, data validation; digital signatures, e.g. ECDSA signatures, signature verification

Secure processing: secure data processing, e.g. encrypted processing workflows; encryption workflows, e.g. data pipelines, processing stages; authentication methods, e.g. key-based authentication, certificate validation.

*Secure workflow design considerations:*

End-to-end protection: end-to-end encryption, e.g. encrypted data pipelines, secure transmission paths; data flow protection, e.g. encrypted channels, protected flows; secure endpoints, e.g. API endpoints, service endpoints

Access and identity: access control integration, e.g. IAM policies, role-based access; identity management, e.g. identity providers, authentication systems; confidentiality controls, e.g. encryption enforcement, access restrictions

Risk and architecture: risk exposure points, e.g. attack surfaces, vulnerability points; trust boundaries, e.g. network boundaries, system separation; multi-layer security, e.g. layered encryption, defence-in-depth, segmented storage, isolated processing.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Explain the core principles of cloud cryptography | | **LO1, LO2 and LO3**<br>**D1** Critically evaluate, using a selection of real-world examples, the effectiveness of cloud cryptography principles, encryption algorithms and secure key management services in supporting secure cloud-based data and services. |
| **P1** Explain, using a selection of real-world examples, the core principles and models of cryptography as applied to cloud environments.<br>**P2** Describe, using a selection of real-world examples, the trust and responsibility models used within cloud environments. | **M1** Analyse, using a selection of real-world examples, the impact of cloud cryptography principles in a range of organisational contexts. | |
| **LO2** Review symmetric and asymmetric encryption mechanisms in cloud environments | | |
| **P3** Review, using a selection of real-world examples, the use of symmetric and asymmetric encryption mechanisms within cloud environments.<br>**P4** Discuss, using a selection of real-world examples, the selection and application of cryptographic algorithms within cloud environments. | **M2** Analyse, using a selection of real-world examples, the suitability of symmetric and asymmetric encryption algorithms for different cloud security requirements. | |
| **LO3** Discuss the secure key management approaches for cloud-based systems | | |
| **P5** Discuss, using a selection of real-world examples, the secure key management approaches and life cycle services used in cloud-based systems.<br>**P6** Review, using a selection of real-world examples, the hardware security management mechanisms for secure key management used by cloud-based systems. | **M3** Analyse, using a selection of real-world examples, the benefits and limitations of secure key management for cloud-based systems. | |

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Implement cryptographic protocols to secure a cloud data workflow for an identified scenario. | | |
| **P7** Implement cryptographic protocols to secure a cloud-based data workflow for an identified scenario.<br><br>**P8** Review the cryptographic protocols to identify strengths and areas for improvement in securing a cloud data workflow for an identified scenario. | **M4** Evaluate, using the results of the review, the effectiveness of the implemented cryptographic protocols in securing cloud-based data workflows for an identified scenario. | **D2** Justify the decisions made in the selection and implementation of the cryptographic protocols in their ability to comprehensively meet the requirements of the identified scenario**.** |

## Recommended Resources

### Textbooks

Allalouf, A. (2024) *Cloud Security Handbook: A Practical Guide for Managing Risk and Compliance*. Hoboken, NJ: Wiley.

Easttom, W. (2021) *Modern Cryptography: Applied Mathematics for Encryption and Information Security*. Cham: Springer.

Green, M. and Smith, J. (2024) *The Cryptopals Crypto Challenges: Modern Cryptography in Practice*. 2nd edn. San Francisco, CA: No Starch Press.

Kornagin, J. (2022) *Cloud Native Security Cookbook: Recipes for a Secure Cloud*. Sebastopol, CA: O'Reilly Media.

Stallings, W. (2023) *Cryptography and Network Security: Principles and Practice*. 8th edn. Harlow: Pearson.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.cloudflare.com/learning/ssl/transport-layer-security-tls/ | Cloudflare Learning Center<br>Explains Transport Layer Security, encryption principles, and secure communication in cloud/web systems<br>(General reference) |
| learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility | Microsoft Learn<br>Detailed explanation of cloud shared responsibility models across IaaS, PaaS and SaaS environments<br>(General reference) |
| developer.mozilla.org/en-US/docs/Web/Security/Defenses/Transport_Layer_Security | MDN Web Docs<br>Explains TLS protocol concepts including encryption, authentication and integrity mechanisms in network security<br>(General reference) |
| www.cybersecurity-insiders.com/symmetric-vs-asymmetric-encryption-in-the-cloud-choosing-the-right-approach/ | Cybersecurity Insiders<br>Compares symmetric and asymmetric encryption approaches and their use in cloud environments<br>(Article) |

| | |
|---|---|
| www.geeksforgeeks.org/computer-networks/difference-between-symmetric-and-asymmetric-key-encryption/ | GeeksforGeeks<br>Explains differences between symmetric and asymmetric encryption including performance and use cases<br>(Tutorial) |
| preyproject.com/blog/types-of-encryption-symmetric-or-asymmetric-rsa-or-aes | Prey Project<br>Overview of encryption types and how symmetric and asymmetric methods are combined in real-world systems<br>(General reference) |
| www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/ | Cloudflare Learning Center<br>Explains TLS handshake process including authentication, key exchange and session establishment<br>(General reference) |
| www.ibm.com/think/topics/key-management | IBM<br>Explains encryption key management fundamentals, including generation, storage, distribution, rotation and destruction across the key life cycle<br>(General reference) |
| www.geeksforgeeks.org/computer-networks/easy-key-management-in-cryptography/ | GeeksforGeeks<br>Explains key management processes and how cryptographic keys are generated, distributed and protected in systems<br>(Tutorial) |
| docs.cloud.google.com/docs/security/key-management-deep-dive | Google Cloud<br>Explains cloud KMS architecture, key types, access control and life cycle operations such as rotation and key destruction<br>(General reference) |

## Journals and articles

Ali, M., Khan, S.U. and Vasilakos, A.V. (2015) 'Security in cloud computing: Opportunities and challenges', *Information Sciences*, 305, pp. 357–383. Available at: doi.org/10.1016/j.ins.2015.01.025.

Dawson, J.K. *et al*. (2023) 'Ensuring privacy and confidentiality of cloud data: A comparative analysis of diverse cryptographic solutions based on run time trend', *PLOS ONE*, 18(9), e0290831. Available at: doi.org/10.1371/journal.pone.0290831.

Dubey, P.R. and Dixit, P. (2025) 'A Comparative Study Of Symmetric And Asymmetric Cryptographic Algorithms For Enhancing Data Security In Cloud Computing', *International Journal of Advance and Applied Research*, 12(5). Available at: doi.org/10.5281/zenodo.17112877.

Durai, K., Ramachandran, R.K. and Mittal, S. (2024) 'A systematic review of homomorphic encryption techniques to preserve confidentiality in cloud environment', *AIP Conference Proceedings*, 3209(1), 020012. Available at: doi.org/10.1063/5.0227839.

## Links

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*

*Unit 537: Fundamentals of Blockchain Technology.*

# Unit 539: Cloud Infrastructure Protection

**Unit code:** Y/652/3325

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Cloud computing underpins a wide range of modern digital services, enabling organisations to deliver scalable, resilient and on-demand infrastructure across sectors including finance, healthcare, retail and public services. As reliance on cloud platforms has increased, so too has the need to protect the infrastructure that supports them. Cloud environments combine shared resources, distributed architectures and external service provision, creating a complex security landscape that must be understood and managed effectively.

The aim of this unit is to develop students' understanding of how cloud infrastructure is protected through a combination of technical controls, governance frameworks and risk management practices. Students will examine how responsibilities are shared between providers and customers, how security decisions are informed by threat and risk analysis, and how protection strategies are selected and justified within organisational contexts.

The unit covers key areas required to secure cloud infrastructure. Students will explore threat models, risk assessment approaches and security domains, before reviewing methods for securing cloud network architectures through segmentation, identity management and traffic control. The unit also addresses physical and environmental protection in data centres, recognising their role in maintaining availability and resilience. These areas are brought together through the development of a regulation-compliant incident response policy that reflects both digital and physical security considerations.

On successful completion of this unit, students will be able to evaluate cloud infrastructure risks, apply appropriate protection strategies and design effective incident response approaches in realistic scenarios. These skills are relevant to a range of roles in the sector, including Cloud Security Analyst, Cloud Infrastructure Engineer, Cyber Security Analyst, Security Operations Centre (SOC) Analyst and Cloud Risk and Compliance Specialist.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the core concepts underpinning cloud infrastructure protection

LO2 Review a range of methods for securing cloud network architectures

LO3 Explore a range of physical protection controls and environmental safeguards that support data centre operations

LO4 Design a regulation-compliant incident response policy to secure a digital and physical architecture from an identified security threat.

## Essential Content

LO1 **Discuss the core concepts underpinning cloud infrastructure protection**

*Threat models and cloud risk landscapes:*

Cloud attack surfaces, e.g. exposed Application Programming Interfaces (APIs), public endpoints; multi-tenancy risks, e.g. tenant co-residency, isolation failure; insider threats, e.g. privileged account misuse, credential compromise; supply chain risks, e.g. compromised images; misconfiguration risks, e.g. open storage buckets, misconfigured security groups; shared infrastructure risks, e.g. hypervisor exploitation; threat vectors, e.g. phishing, credential theft

Threat modelling and identification: threat modelling approaches, e.g. STRIDE, attack trees; vulnerability identification, e.g. vulnerability scanning, common vulnerabilities and exposures (CVE) mapping; attack scenarios, e.g. lateral movement, privilege escalation

Risk exposure and assessment: risk exposure, e.g. exposed services, weak access controls; risk assessment, e.g. likelihood-impact scoring, risk analysis.

*Cloud security domains and responsibility frameworks:*

Responsibility models: shared responsibility model, e.g. provider-managed infrastructure, customer-managed workloads; provider responsibilities, e.g. physical security, hypervisor security; customer responsibilities, e.g. identity management, data protection; accountability boundaries, e.g. control ownership, responsibility allocation

Security domains: infrastructure security, e.g. virtual machine (VM) isolation, network segmentation; platform security, e.g. runtime security, managed service patching; application security, e.g. API security, input validation

Service models and scope: service models, e.g. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS); trust boundaries, e.g. tenant isolation, network domains.

*Security controls and governance alignment:*

Control types: preventive controls, e.g. firewalls, access control policies; detective controls, e.g. intrusion detection systems, monitoring logs; corrective controls, e.g. patching, incident remediation

Governance frameworks and standards: governance frameworks, e.g. ISO 27001, NIST Cybersecurity Framework (CSF); security standards, e.g. Center for Internet Security (CIS) benchmarks; compliance frameworks, e.g. General Data Protection Regulation (GDPR); knowledge currency, evolving practices

Policy and control management: policy-driven security, e.g. configuration baselines, security policies; control mapping, e.g. NIST mappings, ISO control mapping; security policies, e.g. identity and access management (IAM) policies

Risk and accountability: risk ownership, e.g. assigned roles, accountability tracking; accountability structures, e.g. governance roles; governance alignment, e.g. audit alignment.

*Data protection governance and cryptographic controls:*

Encryption models: encryption at rest, e.g. disk encryption; encryption in transit, e.g. Transport Layer Security (TLS) encryption; encryption in use, e.g. confidential computing

Access and key control: key management systems (KMS), e.g. cloud KMS, key vaults; access control policies, e.g. IAM roles; cryptographic controls, e.g. key rotation

Data governance life cycle: data life cycle governance, e.g. retention schedules, secure deletion; retention policies, e.g. time-based retention; data classification, e.g. sensitivity labels

Standards and protection mechanisms: data protection standards, e.g. ISO controls; secure storage, e.g. encrypted storage services; policy alignment, e.g. governance controls.

LO2 **Review a range of methods for securing cloud network architectures**

*Network segmentation and isolation models:*

Segmentation strategies: logical segmentation, e.g. virtual private cloud (VPC) segmentation, subnet isolation; micro-segmentation, e.g. workload-level policies, identity-based segmentation; network zoning, e.g. demilitarised zones (DMZs), internal security zones

Isolation mechanisms: network isolation, e.g. tenant isolation, private subnets; isolation boundaries, e.g. security groups, network access control lists (ACL) boundaries; network partitioning, e.g. segmented routing, traffic separation

Access segmentation: segmentation policies, e.g. policy-based access control, zero-trust rules; access segmentation, e.g. identity-based segmentation, role-linked access; traffic isolation, e.g. east-west traffic filtering, restricted communication paths.

*Identity-centric network security:*

Identity controls: identity and access management, e.g. IAM services, identity providers; role-based access control, e.g. role-based access control (RBAC) policies, role assignments; least privilege principles, e.g. minimal access rights, just-in-time access

Authentication mechanisms: authentication mechanisms, e.g. multi-factor authentication, token-based authentication; identity verification, e.g. certificate validation, identity tokens; credential management, e.g. secure credential storage, key rotation

Federation and governance: federated identity, e.g. single sign-on (SSO), identity federation protocols; service identities, e.g. workload identities, service accounts; access governance, e.g. policy frameworks, identity governance models.

*Traffic control and perimeter security approaches:*

Perimeter controls: firewalls, e.g. network firewalls, web application firewalls; network security groups, e.g. rule-based filtering, port-level controls; perimeter security, e.g. gateway firewalls, edge security controls

Traffic inspection: traffic inspection, e.g. deep packet inspection, flow monitoring; packet filtering, e.g. ACL rules, stateful filtering; intrusion detection systems, e.g. signature-based detection, anomaly detection

Threat prevention: intrusion prevention systems, e.g. automated blocking, inline mitigation; zero-trust networking, e.g. identity verification, continuous validation; software-defined networking security, e.g. policy-driven networking, programmable controls

Monitoring and control: traffic monitoring, e.g. VPC flow logs, traffic analytics; access control lists, e.g. network ACLs, rule sets; network gateways, e.g. API gateways, ingress/egress controllers.

*Cloud network service protection mechanisms:*

Service protection: secure API gateways, e.g. API authentication, request validation; load balancer security, e.g. TLS termination, backend protection; traffic filtering policies, e.g. rule enforcement, request filtering

Availability and resilience: Distributed Denial of Service (DDoS) protection services, e.g. traffic scrubbing, rate limiting; service resilience, e.g. failover routing, redundancy configurations; availability protection, e.g. autoscaling, health checks

Detection and monitoring: monitoring integration, e.g. metrics collection, observability services; logging systems, e.g. audit logs, event logs; threat detection, e.g. anomaly detection, behavioural monitoring

Security analytics: network analytics, e.g. traffic pattern analysis, flow analysis; security monitoring, e.g. Security Information and Event Management (SIEM) integration, alerting systems; anomaly detection, e.g. behavioural baselines, deviation detection.

LO3 **Explore a range of physical protection controls and environmental safeguards that support data centre operations**

*Physical access protection mechanisms:*

Access control systems: external and internal perimeter security, e.g. fenced perimeters, secure entry zones; controlled access points, e.g. mantraps, turnstile systems; authentication systems, e.g. radio-frequency identification (RFID) cards, biometric access; visitor management, e.g. access logs, identity verification

Monitoring and detection: surveillance systems, e.g. CCTV monitoring, video analytics; intrusion detection, e.g. motion sensors, alarm systems; access logging, e.g. badge logs, audit trails

Physical protection controls: entry controls, e.g. multi-factor physical access; access authorisation, e.g. role-based permissions; physical barriers, e.g. locked racks, secure enclosures.

*Environmental and operational safeguards:*

Environmental control systems: ventilation systems, e.g. airflow control; air conditioning systems, e.g. precision cooling; temperature control, e.g. thermal sensors; humidity control, e.g. environmental monitoring systems

Detection and protection: fire detection systems, e.g. smoke sensors; fire suppression systems, e.g. gas suppression; environmental monitoring, e.g. data centre infrastructure management (DCIM) tools, alerting systems; hazard detection, e.g. leak detection sensors

Operational management: airflow management, e.g. hot/cold aisle containment; climate control, e.g. automated heating, ventilation and air conditioning (HVAC) systems.

*Power and resilience infrastructure:*

Power systems: uninterruptible power supply (UPS), e.g. battery backup systems; backup generators, e.g. diesel generators with ATS; power distribution units (PDUs), e.g. rack-level PDUs with monitoring; electrical safety controls, e.g. circuit breakers, grounding systems

Resilience mechanisms: power redundancy, e.g. dual power feeds, N+1 redundancy; failover mechanisms, e.g. automatic transfer switches; load balancing, e.g. distributed power load across circuits

Monitoring and continuity: surge protection, e.g. voltage stabilisation; power monitoring, e.g. energy monitoring systems; backup systems, e.g. redundant power; continuity planning, e.g. disaster recovery strategies.

*Hardware and facility-level protection:*

Hardware security: hardware security modules (HSM), e.g. secure key storage; secure equipment housing, e.g. locked cabinets; tamper detection, e.g. intrusion alerts

Asset management and control: asset management controls, e.g. inventory systems; rack security, e.g. lockable racks; physical asset tracking, e.g. RFID tagging

Facility management and life cycle: facility management, e.g. building management systems (BMS), DCIM platforms; hardware life cycle management, e.g. deployment, decommissioning; equipment monitoring, e.g. diagnostics, telemetry.

LO4 **Design a regulation-compliant incident response policy to secure a digital and physical architecture from an identified security threat**

*Incident response governance and compliance requirements:*

Regulatory and compliance frameworks: regulatory frameworks, e.g. General Data Protection Regulations (GDPR), ISO 27001; compliance obligations, e.g. legal requirements, security controls; reporting requirements, e.g. breach notification, mandatory disclosure; documentation standards, e.g. incident logs, audit records

Governance and accountability: organisational responsibilities, e.g. incident response teams, Security Operations Centre (SOC) roles; governance structures, e.g. security governance models, control frameworks; accountability mechanisms, e.g. Responsible, Accountable, Consulted, and Informed (RACI) models, escalation ownership; policy frameworks, e.g. incident response plans, security playbooks

Audit and monitoring: audit requirements, e.g. compliance audits, control validation; compliance monitoring, e.g. SIEM monitoring, log correlation; regulatory alignment, e.g. policy compliance, standards mapping.

*Incident response life cycle structure:*

Detection and identification: incident identification, e.g. alerting systems, anomaly detection; threat detection, e.g. SIEM, IDS alerts; incident classification, e.g. severity levels, incident categorisation

Containment and eradication: containment strategies, e.g. network isolation, account lockdown; eradication processes, e.g. malware removal, patch deployment; escalation procedures, e.g. incident escalation workflows, SOC escalation paths

Recovery and co-ordination: recovery planning, e.g. service restoration, data recovery; response co-ordination, e.g. incident response teams, communication channels; workflow processes, e.g. structured playbooks, life cycle management.

*Digital and physical incident integration:*

Threat integration: cyber incidents, e.g. data breaches, ransomware; physical security breaches, e.g. unauthorised access, perimeter breach; environmental incidents, e.g. fire events, power outages

Response co-ordination: threat correlation, e.g. SIEM correlation rules, cross-domain alerting; co-ordinated response, e.g. integrated incident handling, unified response teams; cross-domain security, e.g. hybrid security controls, integrated monitoring; escalation pathways, e.g. cross-team escalation, incident workflows

Infrastructure protection: incident co-ordination, e.g. centralised response platforms; infrastructure protection, e.g. network isolation, physical safeguards; hybrid threats, e.g. cyber-physical attacks, blended threats; response integration, e.g. unified response orchestration.

*Post-incident reporting and improvement processes:*

Incident reporting and notification: incident reporting, e.g. reporting systems, ticketing platforms; regulatory notification, e.g. breach reporting, compliance submission; documentation updates, e.g. audit records, incident logs

Analysis and remediation: root cause analysis, e.g. forensic analysis, log analysis; remediation planning, e.g. patching plans, mitigation controls; policy review, e.g. control updates, governance adjustments

Continuous improvement: corrective actions, e.g. system hardening, control tuning; lessons learned, e.g. post-incident reviews, retrospectives; performance evaluation, e.g. response metrics, key performance indicators (KPI) analysis; audit reviews, e.g. compliance verification, control assessment; risk reduction, e.g. mitigation strategies, control enhancement.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Discuss the core concepts underpinning cloud infrastructure protection</td><td rowspan="8">LO1, LO2 and LO3<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of governance-led security controls, network security methods and physical protection mechanisms in securing modern cloud infrastructures.</td></tr>
<tr><td>P1 Explain, using a selection of real-world examples, the threat models and risk landscapes affecting modern cloud infrastructures.<br><br>P2 Discuss, using a selection of real-world examples, the core cloud infrastructure protection concepts in relation to security domains, security controls and data governance.</td><td>M1 Analyse, using a selection of real-world examples, the relationship between threat models, security controls, and governance frameworks in supporting cloud infrastructure protection.</td></tr>
<tr><td colspan="2">LO2 Review a range of methods for securing cloud network architectures</td></tr>
<tr><td>P3 Review, using a selection of real-world examples, the network and identity based methods for securing cloud network architectures for organisational contexts.<br><br>P4 Explore, using a selection of real-world examples, the traffic control and service protection mechanisms for securing cloud network architectures for organisational contexts.</td><td>M2 Assess, using a selection of real-world examples, the effectiveness of the different methods for securing cloud network architectures for identified organisational contexts.</td></tr>
<tr><td colspan="2">LO3 Explore a range of physical protection controls and environmental safeguards that support data centre operations</td></tr>
<tr><td>P5 Explore, using a selection of real-world examples, the physical protection controls that support secure data centre operations.<br><br>P6 Discuss, using a selection of real-world examples, the environmental safeguards that support secure data centre operations.</td><td>M3 Assess, using a selection of real-world examples, the benefits and limitations of implementing physical and environmental safeguards in a data centre infrastructure.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO4** Design a regulation-compliant incident response policy to secure a digital and physical architecture from an identified security threat | | |
| **P7** Design a regulation-compliant incident response policy to secure a digital and physical architecture from an identified security threat.<br><br>**P8** Review the regulation-compliant incident response policy with a range of identified stakeholders to identify strengths and areas for improvement in securing a digital and physical architecture from an identified security threat. | **M4** Evaluate, using the stakeholder feedback, the effectiveness of the incident response policy in securing a digital and physical architecture from an identified security threat. | **D2** Justify the decisions made in the development of the incident response policy that allows it to effectively secure a digital and physical architecture from an identified security threat. |

## Recommended Resources

### Textbooks

Dotson, C. (2023) *Practical Cloud Security: A Guide for Secure Design and Deployment*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Kim, D. and Solomon, M.G. (2021) *Fundamentals of Information Systems Security*. 4th edn. Burlington, MA: Jones & Bartlett Learning.

Mather, T., Kumaraswamy, S. and Latif, S. (2023) *Cloud Security and Privacy*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Rittinghouse, J.W. and Ransome, J.F. (2024) *Cloud Computing: Implementation, Management and Security*. Boca Raton, FL: CRC Press.

Zhang, L. and Chen, L. (2021) *Cloud Data Center Network Architectures and Technologies*. Boca Raton, FL: CRC Press.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| aws.amazon.com/trust-center/data-center/our-controls/ | AWS Trust Center<br>Data centre security controls<br>Explains how cloud data centres are designed with layered physical, environmental and operational controls including redundancy and availability zones<br>(General reference) |
| www.datacenterknowledge.com/security-and-risk-management/data-centers-integrate-cyber-and-physical-security-in-2025 | Data Center Knowledge<br>Describes how modern cloud data centres combine physical controls, identity security and monitoring into unified security frameworks<br>(Article) |
| www.intelligentdatacentres.com/2024/06/27/from-perimeter-to-rack-unpacking-the-five-layers-of-data-centre-protection/ | Intelligent Data Centres<br>Explains defence-in-depth approaches from perimeter security to rack-level controls for protecting cloud data centre infrastructure<br>(Article) |

 © Pearson Education Limited 2026

| | |
|---|---|
| drata.com/learn/risk/stride-threat-model | Drata Trust Academy<br>Introduces STRIDE and structured threat modelling to identify risks and vulnerabilities in system design<br>(Tutorial) |
| www.cloudflare.com/learning/access-management/what-is-identity-and-access-management/ | Cloudflare Learning<br>Explains IAM concepts, authentication methods and access control in cloud environments<br>(General reference) |
| www.cisco.com/site/us/en/learn/topics/security/what-is-micro-segmentation.html | Cisco<br>Describes how segmentation isolates workloads, limits east-west traffic and reduces attack surfaces in cloud and data centre networks<br>(General reference) |
| www.ituonline.com/blogs/best-practices-for-cloud-network-segmentation-and-microsegmentation/ | ITU Online<br>Explains segmentation strategies, trust zones and least privilege controls for securing cloud architectures<br>(Article) |
| encoradvisors.com/data-center-physical-security/ | Encor Advisors<br>Covers access control, surveillance and layered physical protection strategies for secure facilities<br>(General reference) |
| bladeroom.com/cybersecurity-and-physical-security-in-data-centres/ | BladeRoom<br>Explores combined digital, physical and environmental controls to secure modern infrastructure<br>(General reference) |
| www.cynet.com/security-foundations/incident-response/nist-incident-response/ | Cynet<br>Explains the phases of incident response including preparation, detection, containment and recovery<br>(General reference) |

## Journals and articles

Bruma, L.M. (2023) 'Cloud Incident Response – a Comprehensive Analysis', *Informatica Economică*, 27(4), pp. 32–43. Available at: doi.org/10.24818/issn14531305/27.4.2023.03.

Oladosu, S.A., Ige, A.B., Ike, C.C., Adepoju, P.A., Amoo, O.O. and Afolabi, A.I. (2023) 'AI-driven security for next-generation data centres: Conceptualising autonomous threat detection and response in cloud-connected environments', *GSC Advanced Research and Reviews*, 15(2), pp. 162–172. Available at: doi.org/10.30574/gscarr.2023.15.2.0136.

Pitkar, H. (2025) 'Cloud Security Automation Through Symmetry: Threat Detection and Response', *Symmetry*, 17(6), 859. Available at: doi.org/10.3390/sym17060859.

Villegas-Ch, W. and García-Ortiz, J.G. (2023) 'Authentication, access, and monitoring systems for critical areas with the use of artificial intelligence integrated into perimeter security in a data center', *Frontiers in Big Data*, 6. Available at: doi.org/10.3389/fdata.2023.1200390.

## Links

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*.

 © Pearson Education Limited 2026

# Unit 540: Improving Organisational Performance

**Unit code:** A/652/3326

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Improving organisational performance is a strategic priority for organisations operating in complex, competitive and rapidly evolving environments. Organisations across sectors require professionals who understand how individuals behave at work, how organisational systems function and how performance can be enhanced through effective leadership, management and evidence-based interventions. This unit is relevant to roles including Team Leaders, Line Managers, HR Practitioners and Project Professionals responsible for improving individual and organisational outcomes.

The purpose of this unit is to develop students' understanding of the key factors that influence organisational performance and how these can be managed and improved. It focuses on the application of organisational behaviour, leadership and management principles to support effective decision-making and performance improvement within a range of organisational contexts.

The unit explores core theories of individual behaviour and motivation, organisational structures and cultures, and the principles and practices used to develop effective individuals and teams. Students will apply this knowledge through the review of real-world case studies and the development of a business strategy designed to improve performance. These topics are structured to support the achievement of the Learning Outcomes and to develop both theoretical understanding and practical application.

On completion of this unit, students will be able to discuss key behavioural theories, review organisational practices, assess development strategies and deliver a performance improvement workshop in a professional context. The unit supports progression to higher-level study in leadership, management and organisational development and provides a strong foundation for employment in roles requiring responsibility for people management, performance improvement and organisational effectiveness.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the core theories of individual behaviour and motivation

LO2 Review, using a range of real-world case studies, the different structures and cultures implemented in a variety of organisations

LO3 Assess the principles and practices for developing effective individuals or teams within organisations

LO4 Deliver a workshop on a business strategy to improve individual or team performance for an identified organisation.

## Essential Content

LO1 **Discuss the core theories of individual behaviour and motivation**

*Theoretical perspectives on individual behaviour:*

Individual behaviour in organisations, including attitudes and values, perception and attribution processes, personality frameworks (e.g. trait theory, Big Five), individual differences

Influence of these factors on workplace behaviour, decision-making and performance outcomes.

*Motivation theories:*

Content theories of motivation, e.g. Maslow's hierarchy of needs, Herzberg's two-factor theory, McClelland's needs theory, McGregor's Theory X and Y, Alderfer's existence, relatedness, growth (ERG) Theory, Hawthorne studies

Application of these theories within organisational practices, e.g. reward systems, performance management and employee engagement strategies.

*Intrinsic and extrinsic motivation:*

Intrinsic and extrinsic motivation, including financial and non-financial incentives.

Impact on employee engagement, productivity, satisfaction and organisational performance.

*Application and contemporary context:*

Comparison of different behavioural and motivation theories in explaining performance outcomes

Consideration of their relevance in different organisational contexts, including hybrid working, digital environments, diversity and inclusion and employee well-being.

LO2 **Review, using a range of real-world case studies, the different structures and cultures implemented in a variety of organisations**

*Organisational structures:*

Types of organisational structures including hierarchical, flat, functional, divisional, matrix and networked structures

Concepts of centralisation, decentralisation and formalisation

Influence of structure on communication, control and decision-making

*Organisational culture and behavioural norms:*

Organisational culture including values, beliefs, norms and behaviours.

Cultural typologies, e.g. power, role, task, person culture and models, e.g. visible and underlying cultural elements.

Influence of leadership on organisational culture.

*Structure, culture and performance:*

Relationship between organisational structure and culture and their combined impact on performance, innovation, employee engagement and organisational effectiveness

Influence of behavioural and motivational theories, e.g. Maslow, Herzberg, McClelland and Hawthorne studies, in organisational environments, management practices and effect on motivation, productivity and performance.

Continuous improvement and performance culture, including the principles of Kaizen, employee participation, collaboration and organisational learning in quality improvement and long-term organisational success.

*Organisational context and change:*

Use of real-world organisational examples to compare structures and cultures and to review how these influence organisational performance, communication and effectiveness in different contexts

Internal and external drivers of organisational change, including digital transformation, globalisation, competitive pressures and regulatory environments

Impact on structure, culture, and organisational performance.

LO3 **Assess the principles and practices for developing effective individuals or teams within organisations**

*Principles of individual development:*

Learning and development concepts, skills development, knowledge development, competence and capability, performance management principles

Role of continuous professional development (CPD) in improving performance.

*Team dynamics and effectiveness:*

Team roles, team dynamics, stages of team development, collaboration, trust, conflict management

Contribution of effective teamwork to organisational performance.

*Leadership and management practices:*

Leadership theories, e.g. transformational, transactional, situational and management practices

Management practices including communication, decision-making, delegation and empowerment in supporting individual and team development.

*Development practices and performance improvement:*

Practices such as coaching, mentoring, feedback systems, appraisal processes and talent development.

Consideration of how these strategies contribute to improved organisational performance.

LO4 **Deliver a workshop on a business strategy to improve individual or team performance for an identified organisation**

*Organisational performance and strategic context:*

Organisational objectives, performance challenges, and performance gaps

Role of strategic alignment in improving individual and team performance.

*Performance improvement strategies:*

Range of strategies to improve performance, including leadership development, motivation strategies, team-based interventions, process improvements, and change management approaches.

*Workshop design and delivery:*

Planning and structuring a workshop, including defining objectives, sequencing content and selecting appropriate delivery methods

Communication techniques, use of visual aids, stakeholder engagement and approaches to presenting strategic recommendations effectively.

*Justification and performance outcomes:*

Use of organisational evidence, performance metrics and expected outcomes to support strategic recommendations

Consideration of risks, sustainability and long-term impact on organisational performance for a range of stakeholders.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Discuss the core theories of individual behaviour and motivation</td><td rowspan="2">D1 Evaluate the effectiveness of behaviour and motivation theories in improving organisational outcomes.</td></tr>
<tr><td>P1 Discuss different theories of individual behaviour and motivation in organisational contexts.<br>P2 Describe how behaviour and motivation influence workplace performance.</td><td>M1 Assess how behaviour and motivation theories are applied within organisations.<br>M2 Analyse how these theories contribute to improving organisational performance.</td></tr>
<tr><td colspan="2">LO2 Review, using a range of real-world case studies, the different structures and cultures implemented in a variety of organisations</td><td rowspan="2">D2 Evaluate how organisational structures and cultures interact to influence organisational performance.</td></tr>
<tr><td>P3 Review organisational structures and cultures using real-world case study examples.<br>P4 Describe key features of organisational structures and cultural types.</td><td>M3 Analyse how organisational structures influence communication and performance in real-world contexts.<br>M4 Assess how organisational culture influences behaviour and organisational effectiveness.</td></tr>
<tr><td colspan="2">LO3 Assess the principles and practices for developing effective individuals or teams within organisations</td><td rowspan="4">LO3 and LO4<br>D3 Evaluate the effectiveness of development strategies and business strategy in improving organisational performance outcomes.</td></tr>
<tr><td>P5 Assess the principles and practices used to develop individuals and teams.<br>P6 Describe how development activities support performance improvement.</td><td>M5 Analyse how leadership approaches and development practices improve individual and team effectiveness.</td></tr>
<tr><td colspan="2">LO4 Deliver a workshop on a business strategy to improve individual or team performance for an identified organisation</td></tr>
<tr><td>P7 Prepare a business strategy, with recommendations, to improve performance suitable for a range of stakeholders.<br>P8 Deliver a business strategy workshop to improve performance in an organisational context.</td><td>M6 Assess how the strategy supports organisational objectives and stakeholder needs in addressing organisational performance challenges.</td></tr>
</table>

## Recommended Resources

### Textbooks

Cameron, E. and Green, M. (2024) *Making Sense of Change Management*. 6th edn. London: Kogan Page.

Colquitt, J.A., LePine, J.A. and Wesson, M.J. (2025) *Organizational Behavior: Improving Performance and Commitment in the Workplace*. New York: McGraw-Hill.

Kotter, J.P. (2021) *Leading Change*. Rev. edn. Boston, MA: Harvard Business Review Press.

McShane, S. and Von Glinow, M.A. (2024) *Organizational Behaviour*. New York: McGraw-Hill.

Northouse, P.G. (2024) *Leadership: Theory and Practice*. 10th edn. Thousand Oaks, CA: Sage.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.cipd.org | Chartered Institute of Personnel and Development<br>Organisational performance, people management and development resources<br>(General reference) |
| hbr.org | Harvard Business Review<br>(Article) |
| www.mckinsey.com | McKinsey & Company<br>(Research) |
| www.peoplemanagement.co.uk | People Management (CIPD)<br>(Article) |
| www.shrm.org | Society for Human Resource Management<br>(General reference) |
| www.strategy-business.com | Strategy+business<br>Organisational strategy and performance improvement insights<br>(Research) |

| | |
|---|---|
| thinkers50.com | Influential management thinkers<br>(General reference) |

**Journals and articles**

Diniz, P., Marques, I. and Pereira, A. (2025). 'Leadership, human-centered management and organizational culture: pathways to well-being and innovative work based on a systematic review', *INPLASY*. Available at: doi.org/10.37766/inplasy2025.10.0101.

Ilic, K., Arpinar, T. and Ada, N. (2024) 'A systematic review study: Do human resources management practices have an impact on organizational performance?', *E3S Web of Conferences*. Available at: doi.org/10.1051/e3sconf/202455801014.

Salas, E., Reyes, D.L. and McDaniel, S.H. (2018) 'The science of teamwork: Progress, reflections, and the road ahead', *American Psychologist*, 73(4), pp. 593–600. Available at: doi.org/10.1037/amp0000334.

Prifti, K. *et al*. (2025) 'Strategic Human Resource Management and Its Impact on Organizational Performance: Empirical Insights', *International Journal of Innovative Technology and Interdisciplinary Sciences*, 8(3), pp.550-594. Available at: doi.org/10.15157/IJITIS.2025.8.3.550-594.

Wang, S. *et al*. (2025) 'Inclusive leadership and employee job performance: mediation of leader-member exchange and organizational identification', *Frontiers in Psychology*. Available at: doi.org/10.3389/fpsyg.2025.1615144.

**Links**

This unit links to the following related units:

*Unit 403: Professional Practice*

*Unit 426: Digital Project Management*

*Unit 503: Risk Assessment and Change Management*

*Unit 527: System Analysis and Design*

*Unit 531: E-Commerce and Strategy.*

# Unit 541: FinOps

**Unit code:** D/652/3327

**Unit level:** 5

**Credit value:** 15

---

## Introduction

As organisations increasingly adopt cloud-native technologies and operate within dynamic, consumption-based environments, managing cloud expenditure has become a critical organisational capability. Cloud platforms offer scalability, flexibility and rapid innovation; however, their usage-based pricing models introduce financial complexity, requiring organisations to monitor, control and optimise spending effectively.

Financial Operations (FinOps) has emerged as a discipline that integrates engineering, finance and business functions to manage cloud costs while maximising business value. It reflects a shift towards real-time financial governance, where cost visibility, accountability and optimisation are embedded within operational and strategic processes. This unit is relevant to roles such as FinOps Analyst, Cloud Financial Manager, DevOps Engineer, Cloud Architect and Digital Operations Manager.

The unit develops an understanding of how organisations apply FinOps principles within cloud environments. Students will discuss core frameworks, explore cost allocation and accountability mechanisms, and review tools used to monitor and optimise financial performance. They will apply this knowledge to develop and present a FinOps strategy that aligns cloud spending with organisational objectives.

On completion of this unit, students will be able to discuss FinOps principles, explore financial management approaches, review tools and practices, and present strategic solutions to optimise cloud expenditure. They will develop analytical, financial and decision-making skills relevant to employment in cloud, digital operations and financial management roles.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the core principles and frameworks of FinOps within an organisational cloud strategy

LO2 Explore organisational cost allocation methods and financial accountability mechanisms in cloud environments

LO3 Review the tools available to implement and monitor FinOps across an organisation

LO4 Present a FinOps strategy to maximise business value from cloud spending to a range of key stakeholders for an identified scenario.

 © Pearson Education Limited 2026

## Essential Content

LO1 **Discuss the core principles and frameworks of FinOps within an organisational cloud strategy**

*FinOps principles and organisational integration:*

Principles including cost visibility, value optimisation, shared accountability, cross-functional collaboration and continuous improvement

Integration of FinOps within organisational structures, including alignment between engineering, finance and business functions

Embedding financial awareness within operational decision-making processes.

*FinOps frameworks and operating models:*

FinOps Foundation framework, life cycle phases (inform, optimise, operate) and domains of practice

Organisational maturity models and stages of FinOps adoption

Roles and responsibilities including FinOps practitioners, finance stakeholders, engineers and leadership teams.

Relationship between cloud consumption and organisational objectives, including cost control, performance optimisation and innovation

Overview of governance structures, decision rights, policies and accountability frameworks

Integration with broader digital transformation and cloud strategies.

*FinOps developments:*

Real-time cost monitoring, cost observability and financial telemetry

Unit economics (e.g. cost per transaction, cost per user) and value-based decision-making

Use of automation and AI in cost optimisation and forecasting

Environmental and sustainability considerations in cloud financial management Knowledge currency, enhancements, professional practice.

LO2 **Explore organisational cost allocation methods and financial accountability mechanisms in cloud environments**

*Cost allocation and financial accountability:*

Resource-based, service-based, product-based and team-based allocation approaches

Use of tagging strategies, metadata and cost categorisation to improve allocation accuracy

Challenges associated with shared resources and multi-cloud environments

Showback and chargeback models and their application within organisations

Budget ownership, financial accountability frameworks and cost responsibility across teams

Behavioural impact of accountability on engineering and operational decision-making.

*Advanced cost management:*

Application of the FinOps life cycle (inform, optimise, operate) to manage expenditure

Continuous cost monitoring, forecasting, budgeting and variance analysis

Integration of financial planning within agile and DevOps environments

Unit cost metrics, benchmarking and performance measurement

Cost forecasting models, anomaly detection and predictive analytics

Strategies for improving cost efficiency while maintaining performance and scalability.

*Organisational implications:*

Impact of allocation and accountability on organisational transparency, efficiency and governance

Influence on decision-making, investment prioritisation and resource optimisation.

LO3 **Review the tools available to implement and monitor FinOps across an organisation**

*Tools and practices:*

Native cloud provider tools (e.g. AWS Cost Explorer, Azure Cost Management, Google Cloud Billing)

Third-party FinOps platforms and integrations with enterprise systems

Interoperability with DevOps pipelines and engineering workflows

Comparison of cloud financial management tools and approaches, including their capabilities, limitations and suitability for organisational use.

*Monitoring, reporting and analytics:*

Development and use of dashboards, cost metrics, key performance indicators (KPIs) and usage reports

Trend analysis, forecasting and performance monitoring across services and teams

Communication of financial data to technical and non-technical stakeholders.

*Optimisation and predictive practices:*

Rightsizing resources, workload scheduling, autoscaling and reserved instance strategies

Savings plans, commitment-based pricing models and cost control mechanisms

Automation tools for cost management and optimisation

Use of machine learning for anomaly detection, forecasting and optimisation

Predictive modelling for usage patterns and cost behaviour

Limitations and risks associated with automated decision-making.

*Evaluation of effectiveness:*

Assessment of how tools and practices support cost visibility, optimisation, governance and collaboration

Identification of limitations, trade-offs and areas for improvement.

LO4 **Present a FinOps strategy to maximise business value from cloud spending to a range of key stakeholders for an identified scenario**

*FinOps context, strategy and approach:*

Identification of organisational objectives, operational requirements, cloud usage patterns and performance challenges

Consideration of stakeholder needs, financial constraints and strategic priorities

Development of FinOps strategies including governance models, cost optimisation approaches and accountability frameworks

Alignment between financial management practices and organisational strategy.

*Strategic planning and decision-making:*

Development of implementation roadmaps, timelines and milestones

Change management, stakeholder engagement and adoption strategies

Integration with engineering, finance and operational processes

Cost–benefit analysis, Return on Investment (ROI) modelling and investment decision frameworks

Trade-offs between cost, performance, scalability and risk

Risk identification and mitigation strategies.

*Performance measurement and stakeholder engagement:*

Development of key performance indicators (KPIs), monitoring frameworks and reporting structures

Continuous improvement processes and long-term optimisation strategies

Ensuring scalability, resilience and sustainability of FinOps practices

Approaches for presenting FinOps strategies to stakeholders, including communication of financial data, use of dashboards and visualisation, and responding to stakeholder questions and feedback.

## Learning Outcomes and Assessment Criteria

| Pass | Merit | Distinction |
|---|---|---|
| **LO1** Discuss the core principles and frameworks of FinOps within an organisational cloud strategy | | |
| **P1** Discuss FinOps principles used in cloud environments.<br>**P2** Describe the use of FinOps frameworks within organisational contexts. | **M1** Analyse how FinOps principles support financial governance and value optimisation. | **D1** Critique the effectiveness of FinOps principles and frameworks in managing cloud expenditure. |
| **LO2** Explore organisational cost allocation methods and financial accountability mechanisms in cloud environments | | |
| **P3** Explore the use of financial accountability mechanisms within organisations.<br>**P4** Outline a range of cost allocation models used in cloud environments. | **M2** Assess how cost allocation and accountability influence cost control and transparency. | **D2** Evaluate the implications of cost allocation and accountability mechanisms on organisational decision-making. |
| **LO3** Review the tools available to implement and monitor FinOps across an organisation | | **LO3 and LO4**<br>**D3** Critically evaluate the effectiveness of a FinOps strategy, supported by appropriate tools and practices, in maximising business value, financial governance and cloud cost optimisation within an organisational context. |
| **P5** Review a range of tools and practices for monitoring cloud financial performance.<br>**P6** Describe how the available tools support reporting and cost management. | **M3** Assess how tools and practices support optimisation and financial monitoring. | |
| **LO4** Present a FinOps strategy to maximise business value from cloud spending to a range of key stakeholders for an identified scenario | | |
| **P7** Present a FinOps strategy suitable to a range of stakeholders.<br>**P8** Prepare a FinOps report to support cloud cost management in an organisational context. | **M4** Appraise how the FinOps strategy supports organisational objectives and cost management. | |

## Recommended Resources

### Textbooks

Kanumuri, S. and Zeier, M. (2024) *Scaling Cloud FinOps: Proven Strategies to Accelerate Financial Success*. Berkeley, CA: Apress.

Labouardy, M. (2026) *Practical FinOps*. Shelter Island, NY: Manning Publications.

Storment, J.R. and Fuller, M. (2023) *Cloud FinOps: Collaborative, Real-Time Cloud Value Decision Making*. 2nd edn. Sebastopol, CA: O'Reilly Media.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| aws.amazon.com | Amazon Web Services<br>(General reference) |
| cloud.google.com | Google Cloud Platform<br>(General reference) |
| www.finops.org | FinOps Foundation<br>(General reference) |
| www.gartner.com | Gartner<br>(Research) |
| www.ibm.com | IBM – Enterprise cloud financial Management and governance models<br>(General reference) |
| www.oracle.com | Oracle Cloud<br>(General reference) |

 © Pearson Education Limited 2026

**Journals and articles**

Deochake, S. (2025) 'ABACUS: A FinOps Service for Cloud Cost Optimization', *arXiv*. Available at: arxiv.org/abs/2501.14753.

Manchana, R. (2024) 'Driving Cloud Cost Efficiency: A Collaborative FinOps Approach for Cloud-Native SaaS', *Journal of Artificial Intelligence & Cloud Computing*, 3(1). Available at: srcpublishers.com/ai-cloud-computing/article/view/2038.

Sivakumaran, V. (2025) 'FinOps and observability: optimizing cloud resource management for cost efficiency', *International Journal of Research in Computer Applications and Information Technology*, 8(1). Available at: doi.org/10.34218/IJRCAIT_08_01_032.

**Links**

This unit links to the following related units:

*Unit 416: Core Cloud Computing*

*Unit 417: Cloud DevOps Automation*

*Unit 503: Risk Assessment and Change Management*

*Unit 518: Cloud Data Centre Operations*

*Unit 519: Cloud System Architecture.*

# Unit 542: API Design and Development

**Unit code:** **F/652/3328**

**Unit level:** **5**

**Credit value:** **15**

---

## Introduction

Application Programming Interfaces (APIs) sit at the centre of modern digital systems, enabling applications, services and platforms to exchange data reliably across cloud, web and mobile environments. From enterprise systems integration to cloud-native architectures, APIs provide the structure through which software components communicate and scale. As organisations increasingly adopt distributed and service-oriented approaches, effective API design has become essential to building robust, secure and adaptable digital solutions.

The aim of this unit is to develop students' ability to design, develop and evaluate APIs in line with current industry expectations. It focuses on building both conceptual understanding and practical capability, enabling students to work with API architectures, apply appropriate standards, and consider key factors such as security, scalability and maintainability when developing solutions.

The unit explores a range of topics that underpin professional API development. Students will examine core principles and architectural styles, including REST and alternative approaches, alongside communication models, data formats and life cycle management. Security is addressed through authentication, authorisation and protection mechanisms, before students apply their knowledge to design and implement an API within a cloud-based scenario. The unit concludes with testing, review and evaluation of API performance, usability and long-term effectiveness.

On successful completion, students will be able to design structured API solutions, implement and test functional APIs, and evaluate their effectiveness against defined requirements. These skills support progression into roles such as Software Developer, API Developer, Cloud Engineer, Systems Integrator and DevOps Engineer, as well as further study in software and cloud computing disciplines.

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Explore the theoretical principles of API design and development

LO2 Discuss the security principles underpinning secure API access

LO3 Develop a public API and associated documentation for a given cloud service scenario

LO4 Review the effectiveness of the developed API for the identified cloud-based scenario.

## Essential Content

LO1 **Explore the theoretical principles of API design and development**

*Conceptual foundations of APIs:*

Design principles: concepts e.g. API endpoints, service interfaces; abstraction principles, e.g. interface abstraction; encapsulation techniques, e.g. interface contracts; separation of concerns, e.g. service boundaries

System interaction models: client–server models, e.g. REST clients, Hypertext Transfer Protocol (HTTP) servers; interface-based interaction, e.g. contract-based communication; service communication, e.g. request–response communication

Architecture qualities: loose coupling, e.g. independent services; system interoperability, e.g. service integration; modular design, e.g. reusable components; integration interfaces, e.g. API gateways; software components, e.g. microservices.

*API architectures and styles:*

REST and web architecture: REST architecture, e.g. RESTful APIs, resource uniform resource identifier (URIs); statelessness, e.g. stateless requests; uniform interface, e.g. standard HTTP methods; resource orientation, e.g. resource-based uniform resource locators (URLs)

Architectural constraints: layered systems, e.g. client–server layering; architectural constraints, e.g. REST constraints; service design patterns, e.g. microservices patterns

Alternative API styles: Simple Object Access Protocol (SOAP) architecture, e.g. SOAP services; GraphQL queries, e.g. query-based APIs; API styles comparison, e.g. REST vs SOAP vs GraphQL; endpoint structuring, e.g. URI design.

*Request and response communication models:*

HTTP communication: HTTP methods, e.g. GET, POST, PUT, DELETE; request headers, e.g. authentication headers; request payloads, e.g. JavaScript Object Notation (JSON) payloads

Response handling: response codes, e.g. 200, 404, 500; status handling, e.g. success/error handling; API responses, e.g. JSON responses

Communication patterns: synchronous communication, e.g. blocking requests; asynchronous communication, e.g. message queues; idempotency principles, e.g. PUT idempotency

Data exchange: resource representation, e.g. JSON objects; data exchange models, e.g. request–response models; API requests, e.g. client calls, event-based exchange, service calls.

*Data formats, standards, and life cycle:*

Data formats and schemas: format, e.g. JSON objects, eXtensible Markup Language (XML) Document Object Model (DOM), attributes, elements; schema definitions, e.g. JSON schema, XML Schema Definition (XSD), XML Schema; data structures, e.g. structured payloads

Standards and specifications: API standards, e.g. REST standards; documentation standards, e.g. API documentation; OpenAPI specifications, e.g. Swagger/OpenAPI

Life cycle and versioning: API versioning, e.g. versioned endpoints; life cycle stages, e.g. development, deployment; deprecation processes, e.g. endpoint deprecation; release management, e.g. version releases; backward compatibility, e.g. version support.

LO2 **Discuss the security principles underpinning secure API access**

*Authentication and authorisation mechanisms:*

Identity and access control: identity verification, e.g. credential validation, identity tokens; access models, e.g. role-based access control (RBAC), policy-based access; token-based authentication, e.g. bearer tokens, access tokens; OAuth2 protocols, e.g. authorization code flow, client credentials; JSON Web Tokens (JWT), e.g. signed tokens, claims-based identity

Authentication processes: multi-factor authentication (MFA), e.g. one-time password (OTP), authenticator apps; session management, e.g. session tokens, expiry handling; identity federation, e.g. single sign-on (SSO), federated identity providers; authorisation policies, e.g. scope-based access, permission rules.

*Transport and data security:*

Secure communication: encryption protocols, e.g. Transport Layer Security (TLS) 1.3, HTTPS; secure communication channels, e.g. encrypted API endpoints; data confidentiality, e.g. encrypted payloads; data integrity, e.g. integrity checks; cryptographic protection, e.g. TLS encryption, key exchange; payload encryption, e.g. encrypted request bodies.

*API protection and access management:*

API protection controls: rate limiting, e.g. request throttling; throttling strategies, e.g. per-user limits; misuse prevention, e.g. abuse detection, anomaly detection

Input and gateway security: input validation, e.g. schema validation, parameter validation; input sanitisation, e.g. injection prevention; API gateways, e.g. request filtering, authentication enforcement

Threat mitigation: access management, e.g. API access policies, gateway controls; threat detection, e.g. behavioural monitoring; denial-of-service mitigation, e.g. rate limiting; injection prevention, e.g. Structured Query Language (SQL) injection protection; request filtering, e.g. header validation.

*Credential and key management:*

Credential management: secret management, e.g. API secrets, token storage; key storage, e.g. secure vaults; credential life cycle, e.g. creation, rotation; access credentials, e.g. API keys, tokens

Key protection and governance: key rotation, e.g. scheduled rotation; encryption keys, e.g. symmetric keys, signing keys; credential auditing, e.g. audit logs, usage tracking; exposure risk management, e.g. secret scanning; key distribution, e.g. secure key exchange.

LO3 **Develop a public API and associated documentation for a given cloud service scenario**

*API design structure and modelling:*

Endpoint and resource design: endpoint definition, e.g. REST endpoints, URI paths; resource naming conventions, e.g. resource-based naming, plural endpoints; HTTP method selection, e.g. GET, POST, PUT, DELETE; schema design, e.g. JSON schema, request models

Request and response structure: request structure, e.g. request payloads, headers; response models, e.g. JSON responses, structured outputs; error handling models, e.g. error codes, standard responses

Design specification: API documentation structure, e.g. OpenAPI specs, Swagger docs; versioning strategies, e.g. URI versioning, header versioning; resource modelling, e.g. entity mapping, resource relationships; interface design, e.g. API contracts, interface definitions; contract definition, e.g. request/response contracts.

*Cloud-aligned implementation considerations:*

Deployment and architecture: cloud-native deployment, e.g. containerised APIs, serverless endpoints; framework selection, e.g. FastAPI, Express; configuration management, e.g. environment variables, config files

Scalability and services: scalability considerations, e.g. horizontal scaling, autoscaling; stateless services, e.g. stateless APIs, sessionless design; cloud integration, e.g. API gateways, cloud services; service orchestration, e.g. container orchestration, service co-ordination

Infrastructure setup: infrastructure configuration, e.g. network setup, service configuration; platform compatibility, e.g. cloud platform support; deployment pipelines, e.g. continuous integration/continuous development (CI/CD) pipelines, automated deployment.

*Testing and validation approaches:*

Testing methods: functional testing, e.g. endpoint testing, request validation; integration testing, e.g. API integration tests; security testing, e.g. authentication testing, vulnerability testing

Performance validation: performance testing, e.g. load testing, stress testing; response verification, e.g. response validation, payload checks; error handling validation, e.g. error responses, failure handling

Testing tools and automation: API testing tools, e.g. Postman, Swagger tools; test automation, e.g. automated test suites; regression testing, e.g. version regression; validation procedures, e.g. verification workflows.

*Deployment contexts:*

Deployment environments: cloud deployment environments, e.g. Amazon Web Services (AWS), Azure; service accessibility, e.g. public endpoints, API exposure; runtime performance, e.g. latency, throughput.

*Documentation quality and usability:*

Documentation standards: API documentation, e.g. endpoint documentation, request/response specs; documentation standards, e.g. OpenAPI, Swagger; schema definitions, e.g. JSON schema; API specifications, e.g. standardised API definitions

Developer usability: usability guidelines, e.g. clear endpoints, consistent naming; developer experience, e.g. onboarding documentation, examples; clarity of content, e.g. readable documentation; completeness of documentation, e.g. full endpoint coverage; interface documentation, e.g. API reference docs.

LO4 **Review the effectiveness of the developed API for the identified cloud-based scenario**

*API testing and performance monitoring:*

Testing and measurement: runtime testing, e.g. endpoint tests, integration tests; response latency, e.g. response time measurement; throughput measurement, e.g. requests per second; reliability metrics, e.g. uptime, success rate

Monitoring and analysis: error rate analysis, e.g. failed requests, exception logs; availability monitoring, e.g. uptime checks, service health; performance profiling, e.g. endpoint profiling, bottleneck detection; monitoring tools, e.g. API monitoring platforms, observability tools

Load and scalability testing: load testing, e.g. concurrent user testing; stress testing, e.g. peak load scenarios; scalability metrics, e.g. capacity limits, scaling thresholds.

*Optimisation and maintainability:*

Design optimisation: design optimisation, e.g. endpoint refinement, resource optimisation; performance tuning, e.g. caching, query optimisation; code maintainability, e.g. clean code practices, modular structure

Architecture and life cycle: extensibility design, e.g. scalable endpoints, reusable services; refactoring practices, e.g. code refactoring, optimisation cycles; version control, e.g. Git-based versioning; backward compatibility, e.g. version support, API stability; life cycle management, e.g. update cycles, maintenance.

*Effectiveness in scenario context:*

Functional performance: functional suitability, e.g. feature completeness, requirement fulfilment; system robustness, e.g. error handling, fault tolerance; usage performance, e.g. response times, system load

Operational effectiveness: operational efficiency, e.g. resource usage, processing efficiency; scalability suitability, e.g. scaling behaviour, workload handling; reliability outcomes, e.g. service stability, uptime

Integration and longevity: workflow integration, e.g. system integration, API consumption; user requirements, e.g. client needs, use-case alignment; service effectiveness, e.g. API usefulness, functionality; deployment context, e.g. cloud environment fit; long-term viability, e.g. maintainability, evolution.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Explore the theoretical principles of API design and development</td><td rowspan="5">LO1 and LO2<br><br>D1 Evaluate, using a selection of real world examples, the effectiveness of different API design and security principles and mechanisms in delivering secure, scalable and maintainable API solutions.</td></tr>
<tr><td>P1 Explore, using a selection of real-world examples, the theoretical principles of API architecture design and development.<br><br>P2 Review, using a selection of real-world examples, the communication models, data formats and life cycles used in API architecture design and development.</td><td>M1 Analyse, using a selection of real-world examples, how theoretical principles of API design and development support the development of modern APIs in a range of organisational contexts.</td></tr>
<tr><td colspan="2">LO2 Discuss the security principles underpinning secure API access</td></tr>
<tr><td>P3 Discuss, using a selection of real-world examples, the authentication, authorisation and data transport mechanisms used to deliver secure API communication.<br><br>P4 Explore, using a selection of real-world examples, the API protection and key management strategies used to provide secure API access.</td><td>M2 Assess, using a selection of real-world examples, the benefits and limitations of implementing API security mechanisms in a range of organisational contexts.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Develop a public API and associated documentation for a given cloud service scenario | | **LO3 and LO4**<br>**D2** Justify the decisions made in the design and development of the API and associated documentation that enable it to comprehensively meet the requirements of an identified cloud service scenario. |
| **P5** Create a design for a public API for a given cloud service scenario using appropriate design structures and protocols.<br>**P6** Develop, using the created design, a public API and associated documentation for a given cloud service scenario. | **M3** Evaluate the effectiveness of the developed API and associated documentation in meeting the requirements of the cloud service scenario. | |
| **LO4** Review the effectiveness of the developed API for the identified cloud-based scenario | | |
| **P7** Implement a test plan and monitoring tools to determine the performance of the developed API for the identified cloud-based scenario.<br>**P8** Review, using the results of the test plan and monitoring tools, the effectiveness of the developed API for the identified cloud-based scenario. | **M4** Optimise the implemented API, using results from the review, to improve the effectiveness of the API in meeting the requirements of the identified cloud-based scenario. | |

## Recommended Resources

### Textbooks

Gough, J., Bryant, D. and Kozlowski, M. (2023) *Mastering API Architecture: Design, Operate, and Evolve API-Based Systems*. Sebastopol, CA: O'Reilly Media.

Higginbotham, J. (2022) *Principles of Web API Design: Delivering Value with APIs and Microservices*. Boston, MA: Addison-Wesley.

Masse, M. (2022) *REST API Design Rulebook*. Sebastopol, CA: O'Reilly Media.

Peralta, J.H. (2025) *Secure APIs: Design, Build, and Implement*. Birmingham: Packt Publishing.

Ramírez, J. (2024) *Building APIs with FastAPI*. Birmingham: Packt Publishing.

Zimmermann, O. *et al*. (2023) *Patterns for API Design: Simplifying Integration with Loosely Coupled Message Exchanges*. Boston, MA: Addison-Wesley.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/node-js/rest-api-introduction/ | GeeksforGeeks<br>REST API fundamentals, HTTP methods, request/response concepts and JSON data exchange<br>(General reference) |
| restfulapi.net/http-methods/ | REST API Tutorial<br>Detailed explanation of HTTP methods (GET, POST, PUT, DELETE) and their role in API communication<br>(Tutorial) |
| blog.postman.com/rest-api-examples/ | Postman<br>Overview of REST APIs with practical examples, architectural principles and common usage scenarios<br>(Blog) |
| blog.4geeks.io/how-to-secure-your-apis-with-oauth-2-0-and-jwt/ | 4Geeks<br>Explanation of OAuth 2.0 and JWT for securing APIs, including roles, tokens and access control concepts<br>(Tutorial) |

| | |
|---|---|
| www.hostragons.com/en/blog/api-security-with-oauth-2-0-and-jwt/ | Hostragons<br>Overview of API security risks and best practices using OAuth2 and JWT for protecting endpoints<br>(General reference) |
| learning.postman.com/docs/tests-and-scripts/test-apis/end-to-end-testing/ | Postman Learning Center<br>Guide to testing API endpoints and workflows using collections, scripts and automated testing<br>(Tutorial) |
| www.stackhawk.com/blog/postman-for-testing-apis/ | StackHawk<br>Explanation of API testing concepts including functional testing, validation and security testing approaches<br>(General reference) |
| www.geeksforgeeks.org/node-js/how-to-test-api-endpoints-with-postman-and-express/ | GeeksforGeeks<br>Practical tutorial on testing API endpoints using Postman, including requests, parameters and responses<br>(Tutorial) |
| moldstud.com/articles/p-ultimate-guide-to-api-versioning-swagger-openapi-best-practices-for-developers | MoldStud<br>Best practices for API versioning using Swagger and OpenAPI, including version control strategies<br>(General reference) |

## Journals and articles

Gbenle, T.P. *et al*. (2022) 'Applying OAuth2 and JWT Protocols in Securing Distributed API Gateways: Best Practices and Case Review', *International Journal of Multidisciplinary Research and Growth Evaluation*, 3(5), pp. 628–634. Available at: doi.org/10.54660/.IJMRGE.2022.3.5.628-634.

Gowda, P. and Gowda, A.N. (2024) 'Best Practices in REST API Design for Enhanced Scalability and Security', *Journal of Artificial Intelligence*, *Machine Learning and Data Science*, 2(1), pp. 827–830. Available at: doi.org/10.51219/JAIMLD/priyanka-gowda/202.

Serbout, S. and Pautasso, C. (2024) 'How Are Web APIs Versioned in Practice? A Large-Scale Empirical Study', *Journal of Web Engineering*, 23(4), pp. 465–506. Available at: doi.org/10.13052/jwe1540-9589.2341.

Singh, D. *et al*. (2024) 'Design and Implementation Strategies for Scalable RESTful APIs in Enterprise Systems', *Power System Technology Journal*, 48(1). Available at: doi.org/10.52783/pst.2103.

## Links

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*.

# Unit 543: Data Structures and Algorithms

**Unit code:** H/652/3329

**Unit level:** 5

**Credit value:** 15

---

## Introduction

Data structures and algorithms underpin the design and performance of modern software systems, forming the basis for how data is organised, processed and retrieved across a wide range of computing applications. From real-time systems and large-scale data platforms to web and enterprise applications, the ability to work effectively with structured data and efficient algorithms is essential to building reliable, scalable and high-performing solutions. This unit introduces students to the core principles that support these systems and explores their practical relevance within contemporary software development contexts.

The aim of this unit is to develop both conceptual understanding and applied skills in the selection, design and implementation of data structures and algorithms. Students will explore how different approaches to data handling influence system behaviour, efficiency and maintainability, and will be supported in making reasoned technical decisions within defined scenarios. The unit brings together theoretical knowledge and practical application to prepare students for the demands of problem solving in professional computing environments.

The content of the unit covers a range of essential topics, including abstract data types, linear and non-linear data structures, and key algorithmic techniques for searching, sorting and optimisation. Students will examine computational complexity and performance measures to understand how solutions scale under different conditions. The unit also includes the development of a programmed solution for a defined scenario, requiring students to select suitable data structures and algorithms, and to review the effectiveness of their approach through testing and evaluation activities.

On successful completion of this unit, students will be able to design and implement structured solutions, evaluate alternative approaches, and justify their technical decisions in relation to performance and requirements. These skills support progression into roles such as Software Developer, Application Developer, Systems Analyst, Data Engineer and Technical Consultant, where structured problem solving, efficient programming and critical evaluation are key professional capabilities.

 © Pearson Education Limited 2026

## Learning Outcomes

By the end of this unit a student will be able to:

LO1 Discuss the key abstract data types and how they are implemented within computer systems

LO2 Explore a range of key algorithmic techniques

LO3 Implement a programmed solution for an identified scenario using data structures and algorithms

LO4 Review the effectiveness of the chosen data structures and algorithms in delivering a programmed solution for an identified scenario.

## Essential Content

LO1 **Discuss the key abstract data types and how they are implemented within computer systems**

*Abstract data types (ADTs):*

Conceptual principles: abstraction concepts, e.g. abstract interfaces, logical data models; interface definitions, e.g. method signatures; implementation separation, e.g. interface vs implementation; data encapsulation, e.g. hidden state, controlled access

Data operations and behaviour: data operations, e.g. insert, delete, update; access methods, e.g. indexed access, sequential access; data behaviour, e.g. stack push/pop, queue enqueue/dequeue; structural modelling, e.g. logical vs physical structure

Representation and structure: memory representation, e.g. contiguous memory, pointer-based structures; data abstraction layers, e.g. interface layer, implementation layer; interface design, e.g. operation contracts.

*Linear data structures:*

Sequential structures: arrays, e.g. static arrays, dynamic arrays; linked lists, e.g. singly linked, doubly linked lists; structural properties, e.g. contiguous vs non-contiguous storage; ordering, e.g. indexed ordering, sequential traversal

Operational behaviour: stacks, e.g. push/pop operations; queues, e.g. enqueue/dequeue operations; circular queues, e.g. circular buffering; priority queues, e.g. heap-based priorities

Performance and usage: access patterns, e.g. random vs sequential access; insertion and deletion behaviour, e.g. constant vs linear time; use cases, e.g. buffering, scheduling.

*Non-linear data structures:*

Tree structures: trees, e.g. hierarchical nodes; binary trees, e.g. left/right child nodes; binary search trees, e.g. ordered nodes; hierarchical structures, e.g. parent-child relationships

Graph structures: graphs, e.g. directed, undirected graphs; network structures, e.g. weighted edges; node relationships, e.g. adjacency links; edge relationships, e.g. weighted connections

Traversal and modelling: traversal techniques, e.g. depth-first search, breadth-first search; adjacency models, e.g. adjacency lists, matrices; recursion structures, e.g. recursive traversal; structural representation, e.g. node-based models.

*Implementation considerations:*

Memory and storage: memory allocation, e.g. static allocation, dynamic allocation; pointers, e.g. pointer referencing; references, e.g. object references; storage efficiency, e.g. memory usage optimisation

Performance and trade-offs: time efficiency, e.g. constant, linear time; trade-off analysis, e.g. time vs space; performance implications, e.g. runtime cost

Design and selection: structure selection, e.g. array vs linked list vs tree; system architecture alignment, e.g. memory constraints; implementation models, e.g. iterative vs recursive.

LO2 **Explore a range of key algorithmic techniques**

*Algorithmic foundations:*

Core principles: algorithm definition, e.g. stepwise procedures; algorithm correctness, e.g. verified outputs; determinism, e.g. consistent output for given input; termination conditions, e.g. finite execution

Problem decomposition: problem decomposition, e.g. divide into subproblems; input–output mapping, e.g. parameter–output relationships; procedural logic, e.g. control flow; execution steps, e.g. ordered operations

Design concepts: algorithm design principles, e.g. structured design; computational models, e.g. stepwise execution.

*Computational complexity:*

Complexity measures: time complexity, e.g. O(1), O(n), O(log n), O(n log n); space complexity, e.g. memory allocation cost; performance metrics, e.g. execution time, memory usage

Analysis methods: Big O Notation, e.g. asymptotic upper bound; best-case analysis, e.g. minimal operations; worst-case analysis, e.g. maximum operations; average-case analysis, e.g. expected operations

Scalability and efficiency: scalability considerations, e.g. input size growth; growth rates, e.g. linear vs logarithmic vs exponential; efficiency analysis, e.g. comparing complexity classes; resource utilisation, e.g. CPU vs memory trade-offs.

*Algorithmic design paradigms:*

Recursive and iterative design: recursion, e.g. recursive calls; iterative algorithms, e.g. loop-based execution; recursion structures, e.g. call stack usage

Core paradigms: divide and conquer, e.g. merge sort decomposition; greedy algorithms, e.g. local optimal choice; dynamic programming, e.g. memoisation, tabulation; backtracking techniques, e.g. search space exploration

Optimisation strategies: optimisation strategies, e.g. pruning, caching; design strategies, e.g. paradigm selection; computational modelling, e.g. algorithm modelling.

*Sorting and searching techniques:*

Sorting algorithms: sorting algorithms, e.g. bubble sort, merge sort, quicksort; comparison-based sorting, e.g. pairwise comparisons; non-comparison sorting, e.g. counting sort; sorting stability, e.g. stable vs unstable

Searching algorithms: searching algorithms, e.g. linear search O(n), binary search O(log n); search optimisation, e.g. sorted input requirement; data ordering, e.g. sorted vs unsorted datasets

Performance characteristics: algorithm efficiency, e.g. O(n), O(log n); divide and conquer sorting, e.g. merge sort, quicksort; algorithm suitability, e.g. dataset size and structure.

LO3 **Implement a programmed solution for an identified scenario using data structures and algorithms**

*Scenario definition and requirements:*

Problem and inputs: problem constraints, e.g. input size limits, boundary conditions; input requirements, e.g. structured input data; output requirements, e.g. expected results, formatted output; functional specifications, e.g. required operations, logic rules

Decomposition and modelling: logical decomposition, e.g. subproblem breakdown; data representation, e.g. structured arrays, objects; requirement analysis, e.g. task mapping; scenario modelling, e.g. system modelling, problem mapping

Processing requirements: system objectives, e.g. functional goals; processing requirements, e.g. computation steps; task breakdown, e.g. modular tasks; requirement mapping, e.g. input–output linkage.

*Data structure selection:*

Structure suitability: structure suitability, e.g. choosing arrays, lists, trees; data organisation, e.g. indexed storage, node-based storage; data relationships, e.g. hierarchical, relational links

Performance and efficiency: storage efficiency, e.g. memory usage; access efficiency, e.g. constant vs linear access; memory usage, e.g. space utilisation; structure optimisation, e.g. efficient representation

Design comparison: structure comparison, e.g. array vs linked list; implementation choice, e.g. structure selection; structural design, e.g. data layout; data handling strategies, e.g. insertion, retrieval methods; performance alignment, e.g. matching structure to problem.

*Algorithm selection:*

Algorithm correctness and suitability: algorithm suitability, e.g. matching algorithm to problem; correctness validation, e.g. expected output verification; algorithm comparison, e.g. alternative approaches; selection criteria, e.g. input size, constraints

Performance considerations: performance efficiency, e.g. execution time; complexity considerations, e.g. time complexity, space complexity; performance trade-offs, e.g. time vs space; processing efficiency, e.g. computational cost

Execution design: computation methods, e.g. iterative, recursive; execution logic, e.g. control flow; algorithm design, e.g. structured algorithm development.

*Solution structure:*

Program design: program architecture, e.g. modular programs, layered structure; modular design, e.g. functions, reusable components; control flow, e.g. loops, conditionals; execution logic, e.g. sequence of operations

Data and system interaction: data flow, e.g. input-processing-output; component interaction, e.g. function calls, module interaction; system integration, e.g. combined components; function decomposition, e.g. modular breakdown

Implementation structure: code structure, e.g. organised code blocks; algorithm integration, e.g. embedding algorithms in code; data processing pipelines, e.g. staged processing; solution design, e.g. overall program structure.

LO4 **Review the effectiveness of the chosen data structures and algorithms in delivering a programmed solution for an identified scenario**

*Effectiveness criteria:*

Correctness and validity: correctness, e.g. expected output validation; solution accuracy, e.g. correct results; output validity, e.g. valid outputs; functional completeness, e.g. full feature coverage

Performance and efficiency: computational efficiency, e.g. execution time; performance metrics, e.g. runtime measurements; scalability, e.g. input size handling; structural clarity, e.g. readable solution structure

Design alignment: requirement alignment, e.g. meeting specifications; design coherence, e.g. consistent logic; evaluation standards, e.g. defined success criteria.

*Performance considerations:*

Execution performance: execution time, e.g. runtime duration; processing speed, e.g. operation throughput; response time, e.g. latency in computation

Memory and resource usage: memory usage, e.g. data structure allocation; space complexity, e.g. memory consumption; resource utilisation, e.g. CPU and memory usage

Efficiency analysis: algorithmic complexity, e.g. time complexity, Big O analysis; performance trade-offs, e.g. time vs space; runtime behaviour, e.g. execution patterns.

*Solution testing:*

Testing approaches: test planning, e.g. structured test cases; unit testing, e.g. function-level tests; integration testing, e.g. system-level tests; boundary testing, e.g. edge case inputs

Validation techniques: equivalence partitioning, e.g. grouped input ranges; input validation, e.g. valid/invalid inputs; assertion checking, e.g. expected conditions

Performance testing: regression testing, e.g. retesting after changes; performance benchmarking, e.g. execution comparison; memory profiling, e.g. memory usage tracking.

*Maintainability and robustness:*

Code quality: code readability, e.g. clear structure; modularity, e.g. separated functions; code reusability, e.g. reusable components

System robustness: fault tolerance, e.g. error resilience; error handling, e.g. exception handling; adaptability, e.g. adaptable logic; system flexibility, e.g. scalable code

Support and evolution: debugging support, e.g. debug tools; version control, e.g. code versioning; documentation quality, e.g. code documentation; long-term stability, e.g. maintainable systems.

*Overall solution evaluation:*

Solution assessment: solution strengths, e.g. effective features; solution limitations, e.g. constraints; implementation suitability, e.g. fit for purpose

Performance outcomes: requirement fulfilment, e.g. meeting objectives; performance outcomes, e.g. efficiency results; algorithm effectiveness, e.g. correct behaviour; structure efficiency, e.g. optimised structures

Future optimisation: optimisation potential, e.g. improvement areas; scalability analysis, e.g. growth handling; robustness evaluation, e.g. stability checks; system reliability, e.g. consistent performance.

## Learning Outcomes and Assessment Criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Discuss the key abstract data types and how they are implemented within computer systems</td><td rowspan="5">LO1 and LO2<br><br>D1 Critically evaluate, using a selection of real-world examples, the effectiveness of abstract data types, structures and key algorithms within computer systems for solving computational problems with varying constraints.</td></tr>
<tr><td>P1 Discuss, using a selection of real-world examples, the key characteristics of abstract data types and their implementation within computer systems.<br><br>P2 Explore, using a selection of real-world examples, linear and non-linear data structures and their implementation within computer systems.</td><td>M1 Analyse, using a selection of real-world examples, the impact of implementing different abstract data types and structures for identified computational use cases.</td></tr>
<tr><td colspan="2">LO2 Explore a range of key algorithmic techniques</td></tr>
<tr><td>P3 Discuss, using a range of real-world examples, the foundations of algorithms and computational complexity.<br><br>P4 Explore, using a range of real-world examples, the key algorithmic design paradigms and techniques of searching and sorting.</td><td>M2 Assess, using a selection of real-world examples, the effectiveness of algorithmic techniques and paradigms in terms of computational complexity.</td></tr>
</table>

| Pass | Merit | Distinction |
|---|---|---|
| **LO3** Implement a programmed solution for an identified scenario using data structures and algorithms | | **LO3 and LO4**<br>**D2** Justify the decisions made in the selection of data structures and algorithms used in the delivery of the programmed data solution that allows it to comprehensively meet the needs of an identified scenario. |
| **P5** Implement a programmed solution for an identified scenario using appropriate data structures and algorithms.<br>**P6** Review the implementation of the data structures and algorithms to identify strengths and areas for improvement. | **M3** Optimise the implemented data structures and algorithms, using results from the review, to improve the performance of the programmed solution against the requirements of the identified scenario. | |
| **LO4** Review the effectiveness of the chosen data structures and algorithms in delivering a programmed solution for an identified scenario | | |
| **P7** Apply a test plan to determine the performance of the chosen data structures and algorithms in delivering a programmed solution that meets the requirements of an identified scenario.<br>**P8** Review, using the results of the test plan, the effectiveness of the chosen data structures and algorithms in delivering a programmed solution that meets the requirements of an identified scenario. | **M4** Evaluate the effectiveness of the chosen data structures and algorithms in how well they allow a programmed solution to meet the requirements of an identified scenario. | |

## Recommended Resources

### Textbooks

Carrano, F.M. et al. (2024) *Data Structures and Abstractions with C++*. 8th edn. Harlow: Pearson.

Chinnathambi, K. (2023) *Absolute Beginner's Guide to Algorithms: A Practical Introduction to Data Structures and Algorithms in JavaScript*. Indianapolis, IN: Pearson.

Goodrich, M.T., Tamassia, R. and Goldwasser, M.H. (2024) *Data Structures and Algorithms in Python*. 2nd edn. Hoboken, NJ: Wiley.

Laaksonen, A. (2024) *Guide to Competitive Programming: Learning and Improving Algorithms Through Contests*. 3rd edn. Cham: Springer.

La Rocca, M. (2021) *Advanced Algorithms and Data Structures*. Shelter Island, NY: Manning Publications.

Pai, G.A.V. (2023) *Textbook of Data Structures and Algorithms, Volume 1: Mastering Linear Data Structures*. Hoboken, NJ: Wiley.

Sedgewick, R. and Wayne, K. (2024) *Algorithms*. 4th edn. Boston, MA: Addison-Wesley.

### Websites

*Website links were valid at the time of initial publication.*

| | |
|---|---|
| www.geeksforgeeks.org/dsa/introduction-to-data-structures/ | GeeksforGeeks<br>Introduction to data structures covering classification, operations and use in computing systems<br>(General reference) |
| www.tutorialspoint.com/data_structures_algorithms/stack_algorithm.htm | TutorialsPoint<br>Stack as an abstract data type including operations and structural behaviour<br>(Tutorial) |
| algocademy.com/blog/trees-and-graphs-fundamental-data-structures-for-coders/ | AlgoCademy<br>Explanation of trees and graphs as non-linear data structures with practical use cases<br>(Blog) |

| | |
|---|---|
| www.geeksforgeeks.org/dsa/analysis-algorithms-big-o-analysis/ | GeeksforGeeks<br>Overview of algorithm analysis including time complexity and asymptotic notation<br>(General reference) |
| www.geeksforgeeks.org/dsa/comparison-among-greedy-divide-and-conquer-and-dynamic-programming-algorithm/ | GeeksforGeeks<br>Comparison of key algorithm design paradigms including greedy, divide and conquer and dynamic programming<br>(General reference) |
| codelucky.com/algorithm-design-techniques/ | CodeLucky<br>Overview of algorithm design techniques and structured approaches to solving computational problems<br>(Tutorial) |
| www.meegle.com/en_us/topics/algorithm/algorithm-performance-evaluation | Meegle<br>Guide to algorithm performance evaluation including efficiency, scalability and reliability metrics<br>(Article) |

**Journals and articles**

Jansson, J. (2019) 'Editorial: Special Issue on Efficient Data Structures', *Algorithms*, 12(7), 136. Available at: doi.org/10.3390/a12070136.

Schreiner, W. (2021) 'Abstract Data Types', *in Thinking Programs*. Cham: Springer. Available at: doi.org/10.1007/978-3-030-80507-4_6.

Yurtkan, K., Kazimoglu, C. and Tekguc, U. (2014) 'Evaluation of Algorithm Implementation Assessment Methods Based on a Data Structure Course', *The Eurasia Proceedings of Educational and Social Sciences,* 1, pp. 341–345. Available at: epess.net/index.php/epess/article/view/56.

Zenil, H. (2020) 'A Review of Methods for Estimating Algorithmic Complexity: Options, Challenges, and New Directions', *Entropy*, 22, 612. Available at: doi.org/10.3390/e22060612.

**Links**

This unit links to the following related units:

*Unit 501: Computing Research Project (Pearson-set)*

*Unit 502: Applied Computing Project (Pearson-set)*.

**October 2026**

**For information about Pearson Qualifications, including Pearson Edexcel and BTEC qualifications visit qualifications.pearson.com**

**Edexcel and BTEC are registered trademarks of Pearson Education Limited**

**Pearson Education Limited. Registered in England and Wales No. 872828**
**Registered Office: 80 Strand, London WC2R 0RL.**

**VAT Reg No GB 278 537121**

