

# Unit 6: Software Design and Development

<b>Unit code:</b>	<b>L/601/6585</b>
<b>QCF Level 3:</b>	<b>BTEC Nationals</b>
<b>Credit value:</b>	<b>10</b>
<b>Guided learning hours:</b>	<b>60</b>

## ● Aim and purpose

To enable learners to understand the principles of software design and be able to use tools to develop software designs.

## ● Unit introduction

To develop a programmed software solution, which meets business and user needs, it is necessary to understand the problem and be very clear in terms of the user requirements. Issues are often caused by poor understanding of user need as well as poor planning. A wide range of different development programming languages and paradigms is available to developers with quite different characteristics and features. Learners will build an appreciation of why different high-level languages are available and why they are chosen in particular situations.

This unit focuses on the design and development process, for learners to start incorporating the systems development lifecycle, and would be an appropriate place to start looking at programming concepts before they undertake more focused programming language units. The unit examines the business context within which solutions can be developed and explores the tools that can be used to demonstrate software designs.

A major part of learners' time will be spent on familiarising themselves with fundamental software development processes and concepts. This will give learners a firm foundation to move onto the more focused programming units.

## ● Learning outcomes

**On completion of this unit a learner should:**

- 1 Know the features of programming languages
- 2 Understand the principles of software design
- 3 Be able to use tools to demonstrate software designs.

# Unit content

---

## 1 Know the features of programming languages

*Programming paradigms:* procedural; object oriented; event driven; supporting tools and environments eg CASE tools, IDE

*Types of language:* visual languages; other eg script and markup languages; simple overviews and uses

*Reasons for choice of language:* organisational policy; suitability in terms of available features and tools; availability of trained staff; reliability; development and maintenance costs; expandability

*Features:* sequence; selection eg case, if ... then ... else; iteration eg repeat – until, while ... do; variables eg naming conventions, local and global variables, logical operators; assignment statements; input statements; output statements

*Data types:* text; integer; floating point; byte; date; Boolean; other eg char, smallint; benefits of appropriate choice of data type eg additional validation, efficiency of storage

## 2 Understand the principles of software design

*Software development lifecycle:* stages eg determination of scope, requirements gathering and specification, design, code, test, maintain

*Software structures:* functions, procedures, classes and objects; abstraction of data; pre-defined code; readability eg comments, appropriate names for variables, indentation; quality of code eg efficiency, reliability, robustness, usability, portability, maintainability

## 3 Be able to use tools to demonstrate software designs

*Requirements specification:* inputs, outputs, processing, user interface; constraints eg hardware platforms, up to date developments, timescales for development

*Design:* structure eg functions, procedures, objects; data; file

*Tools:* eg structure diagrams, DFDs, ERM; algorithms eg using pseudo code

*Review:* against specifications requirements

## Assessment and grading criteria

In order to pass this unit, the evidence that learners presents for assessment needs to demonstrate that they can meet all the learning outcomes for the unit. The assessment criteria for a pass grade describe the level of achievement required to pass this unit.

Assessment and grading criteria		
To achieve a pass grade the evidence must show that the learner is able to:	To achieve a merit grade the evidence must show that, in addition to the pass criteria, the learner is able to:	To achieve a distinction grade the evidence must show that, in addition to the pass and merit criteria, the learner is able to:
<b>P1</b> describe the application and limits of procedural, object oriented and event driven programming paradigms		
<b>P2</b> describe the factors influencing the choice of programming language		
<b>P3</b> explain sequence, selection and iteration as used in computer programming		
<b>P4</b> outline the benefits of having a variety of data types available to the programmer [EP6]		
<b>P5</b> explain the role of software design principles and software structures in the IT systems development lifecycle	<b>M1</b> explain the importance of the quality of code	<b>D1</b> discuss the factors that can improve the readability of code
<b>P6</b> use appropriate tools to design a solution to a defined requirement. [IE2, CT1, EP3, SM2]	<b>M2</b> justify the choice of data types and software structures used in a design solution. [IE6]	<b>D2</b> develop algorithms to represent a design solution. [CT6]

**PLTS:** This summary references where applicable, in the square brackets, the elements of the personal, learning and thinking skills applicable in the pass criteria. It identifies opportunities for learners to demonstrate effective application of the referenced elements of the skills.

Key	IE – independent enquirers	RL – reflective learners	SM – self-managers
	CT – creative thinkers	TW – team workers	EP – effective participators

# Essential guidance for tutors

## Delivery

The software development cycle should be constantly referred to as the structure for developing programmed solutions. The various design tools, such as data flow diagrams, may have been introduced to learners in other units, for example systems analysis. Note that these are given as examples of design tools and the tools chosen should be appropriate to the task.

Learners will be designing, developing, testing and documenting programs and sufficient time must be allowed for revisiting the procedures for testing and modifying programs (including retesting after modification) and documenting the solutions both for the technician and the user.

## Outline learning plan

The outline learning plan has been included in this unit as guidance and can be used in conjunction with the programme of suggested assignments.

The outline learning plan demonstrates one way in planning the delivery and assessment of this unit.

Topic and suggested assignments/activities and/assessment
Introduction to the unit
Features of languages: <ul style="list-style-type: none"><li>• individual exercise – tutor assessment of current level of knowledge</li><li>• whole-class exercise – tutor presentation on programming languages – type, when used, why used</li><li>• whole-class exercise – tutor presentation on programming features, followed by individual exercise</li><li>• whole-class exercise – tutor presentation on data types, followed by individual exercise</li><li>• mixture of tutor-led discussions, practical exercises.</li></ul>
<b>Assignment 1 – Which Language?</b>
Software design principles: <ul style="list-style-type: none"><li>• whole-class exercise – tutor presentation on development lifecycle</li><li>• whole-class exercise – tutor presentation on structures and tools, followed by individual exercise</li><li>• whole-class exercise – tutor presentation on moving from diagrams to flowcharts to code, followed by individual exercise</li><li>• mixture of tutor-led instruction, directed learning, practical exercises.</li></ul>
<b>Assignment 2 – Design Workshop</b>
Designing a solution: <ul style="list-style-type: none"><li>• whole-class exercise – tutor presentation on determining the requirement, followed by individual exercise</li><li>• whole-class exercise – tutor presentation on choosing tools, structures and data types, followed by individual exercise</li><li>• whole-class exercise – tutor presentation on developing algorithms, followed by individual exercise</li><li>• mixture of tutor-led instruction, directed learning, practical exercises.</li></ul>

## Topic and suggested assignments/activities and/assessment

Designing a test plan:

- whole-class exercise – tutor presentation on debugging, test strategy, documenting testing, followed by individual exercise
- revision as required and practical work.

Evaluating the design:

- revision of evaluation techniques.

### Assignment 3 – Good Design!

## Assessment

To achieve a pass grade, learners must achieve all of the six pass criteria listed in the assessment and grading criteria grid.

To obtain a merit grade, learners must successfully complete all of the pass criteria, and the two merit criteria.

To achieve a distinction grade, learners must achieve all of the pass and merit criteria and the two distinction criteria.

It is suggested that this unit is assessed through three assignments as summarised in the *Programme of suggested assignments* table which follows this guidance.

As identified in the unit introduction, one key theme that should be emphasised through this unit, and the qualification, is that of creating solutions to meet defined requirements and user need. The evidence for P6 must be based on requirements that relate to a particular situation. There should be a defined client and, ideally, this should not be the tutor. If circumstances dictate that no external client can be identified and a group assignment is set, then the scenario should be richly detailed and the requirements clearly specified.

### Assignment 1 – Which Language?

A possible scenario for the first part of the assessment could be that the learner is asked to lead a session on programming languages, to be delivered to a group of trainee programmers. The evidence for P1–P3 could be provided through a presentation. The presentations do not need to be actually presented verbally if sufficient detail or speakers' notes are provided.

For P1, there are three different programming paradigms identified in the unit content for learning outcome 1, learners are expected to reference each of them.

For P2, learners must describe the factors influencing choice of programming language. Specific languages are not listed in the unit content. Learners should look at a minimum of three, including one visual language.

For P3, learners should present information about the languages they referenced in P2 but should give particular attention to whichever language they would expect to use when implementing the software design that they produce in P6.

A written response for P4 may be appropriate, or a table with perhaps three or four columns: data type, example, space occupied and comment. The data types available may be different according to which language is chosen, however there is an expectation that learners know of the wider range of data types available, six would be sufficient.

## Assignment 2 – Design Workshop

Any context used for this assignment should **not** be the same as that used for P6. A possible scenario for this assignment would be for learners to prepare to role play a teaching role, where they would be giving instruction to a small group. Evidence for P5 could be in the form of worksheets and teaching notes prepared by learners for the topics of design principles, software structures, and the development lifecycle. The role play does not need to actually be carried out, although it might be useful to do so for M1 and/or D1.

For M1, further worksheets and notes, explaining the importance of the quality of code would be required. The evidence might also be in the form of a verbal presentation with supporting notes. Role play as discussed above may be an appropriate vehicle.

For D1, a discussion of the factors that can improve the readability of code is required. This might be in the form of a verbal presentation with supporting notes. Role play as discussed previously may be an appropriate vehicle for this.

## Assignment 3 – Good Design!

The context for this assignment will depend on the case study used. As noted above, P6 should be based on a substantial activity. The design should include the original requirement. Evidence will be in the form of the design documentation.

Evidence for M2 can be included within the design documentation and based on the data types used in the design.

D2 requires learners to develop algorithms to represent a design solution. These may be in any reasonable format as long as they are understandable. It is expected that learners will annotate the algorithms to make it clear what they are doing and how they are doing it. The annotation should also indicate how the design requirements are being met.

## Programme of suggested assignments

The table below shows a programme of suggested assignments that cover the pass, merit and distinction criteria in the assessment and grading grid. This is for guidance and it is recommended that centres either write their own assignments or adapt any Pearson assignments to meet local needs and resources.

Criteria covered	Assignment title	Scenario	Assessment method
P1, P2, P3, P4, M1	Which Language?	You are to lead a session on programming languages for a group of trainee programmers.	Presentation Handouts Notes
P5, M1, D1	Design Workshop	You are to lead a workshop on software design for a group of trainee programmers.	Worksheets Teaching notes
P6, M2, D2	Good Design!	You work as a programmer for a small business and have been asked to design a program to meet a particular need.	Design documentation

## Links to other BTEC units

This unit forms part of the BTEC in IT sector suite. This unit has particular links with the following unit titles in the IT suite:

Level 1	Level 2	Level 3
		Unit 14: Event Driven Programming
		Unit 15: Object Oriented Programming
		Unit 16: Procedural Programming

## Essential resources

Learners will need individual access to a particular programming language and development environment. They will also need more limited access, possibly by demonstration, to other different types of languages.

## Employer engagement and vocational contexts

The use of vocational context is essential in the delivery and assessment of this unit. Learners will require access to computer equipment to enable them to gain a practical awareness and enable them to apply their knowledge and understanding to a practical situation.

There is a range of organisations that may be able to help to centres engage and involve local employers in the delivery of this unit, for example:

- Learning and Skills Network
- Network for Science, Technology, Engineering and Maths Network Ambassadors Scheme
- National Education and Business Partnership Network
- Local, regional Business links
- Work-based learning guidance.

## Delivery of personal, learning and thinking skills

The table below identifies the opportunities for personal, learning and thinking skills (PLTS) that have been included within the pass assessment criteria of this unit.

Skill	When learners are ...
Independent enquirers	planning and carrying out research to test a programmed solution, appreciating the consequences of decisions
Creative thinkers	generating ideas and exploring possibilities when using appropriate tools to design a solution to a defined requirement
Self-managers	working towards implementing a working programmed solution, showing initiative, commitment and perseverance
Effective participators	acting as an advocate for views and beliefs about the benefits of having a variety of data types available to the programmer that may differ from your own  propose a practical design for a programmed solution to a defined requirement, breaking it down into manageable steps.

Although PLTS are identified within this unit as an inherent part of the assessment criteria, there are further opportunities to develop a range of PLTS through various approaches to teaching and learning.

Skill	When learners are ...
Independent enquirers	justifying the choice of data types used in a programmed solution, using reasoned arguments and evidence
Creative thinkers	adapting and improving a programmed solution to benefit others as well as themselves.

### ● Functional Skills – Level 2

Skill	When learners are ...
<b>ICT – Using ICT</b>	
Plan solutions to complex tasks by analysing the necessary stages	describing the stages of the software development lifecycle
Use appropriate software to meet the requirements of a complex data-handling task	designing a software solution
Evaluate the selection, use and effectiveness of ICT tools and facilities used to present information	reviewing the design against the original requirement.