Unit 13: Software Testing

Delivery guidance

This unit presents learners with the engaging challenge of first evaluating different testing methodologies before moving on to test a software development project to assure their performance and overall quality before release.

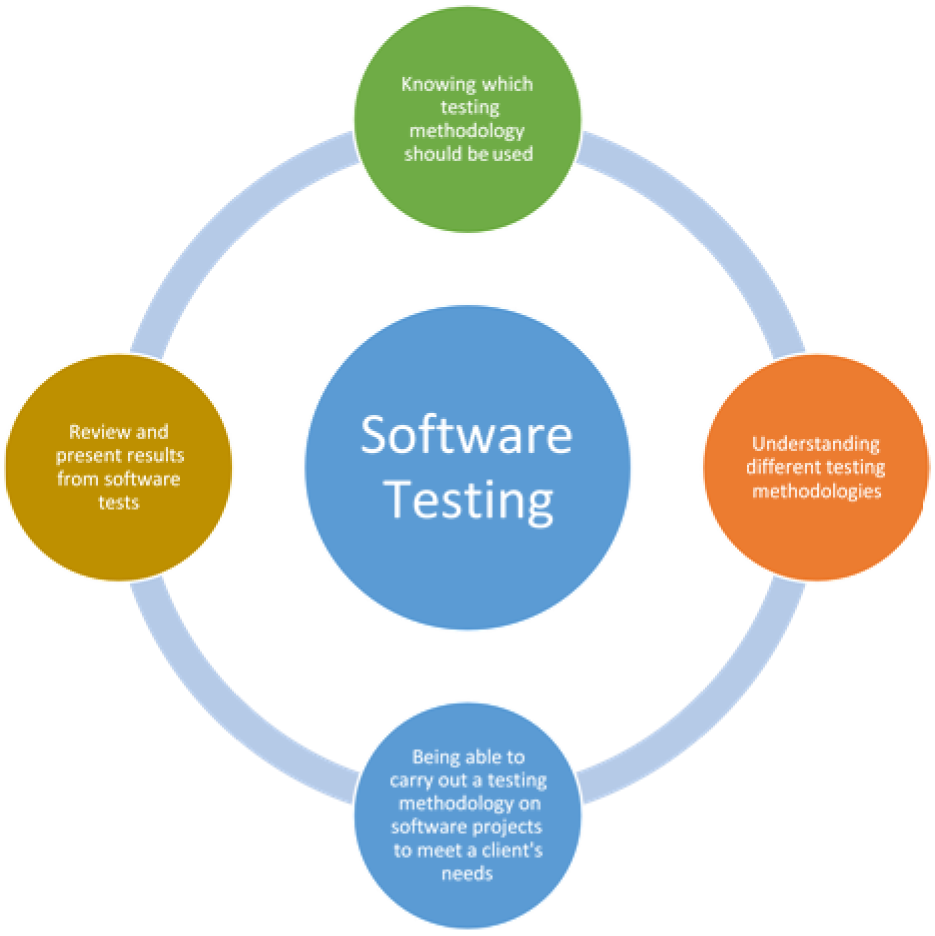
Without doubt, testing is a critical part of the software development life cycle and its impact on the overall success of a project simply cannot be ignored.

Done well, testing provides customers and users with a product that they can use with confidence, minor and major bugs having been identified and fixed through appropriate change controls and comprehensive testing.

However, bringing poorly tested code to the marketplace can lead to a financial loss, reputational damage and, potentially, a client's irreversible loss of confidence in the developers and the product.

Indeed, in some extreme cases such as aviation, traffic management and safety systems, it is possible that a simple undocumented software bug could actually lead to loss of life.

The key to becoming a successful software tester relies on four interlocking core skills:


Knowing which testing methodology should be used
Review and present results from software tests
Software Testing
Understanding different testing methodologies
Being able to carry out a testing methodology on software projects to meet a client's needs

Of course, there are many different methodologies and tools (automated and manual) that can be used to test a software; knowing which is the most appropriate to use is not always immediately obvious – it is a developed skill, pushing learners' analytical and problem-solving abilities to the highest levels.

Learners will also find themselves needing to refine their communication, analysis, evaluation and presentation skills, as reviewing and presenting the results from the software tests to an emotionally invested (and possibly very resistant) development team is a core aspect of this unit.

Learners wishing to pursue a programming career could be encouraged by reminding them that software testing is a well-trodden path towards a full-fledged software development career.

Approaching the unit

This unit lends itself to a three-fold approach to learning:

- research-based investigation into different software development and testing methodologies commonly used during the development life cycle to quality assure software
- practical testing of a software product using a range of appropriate testing methodologies
- analysis and presentation of test results to the development team, offering areas for improvement by evaluating characteristics of testing, methodologies and software products.

Learners are likely to enjoy the problem-solving aspects of the unit as identifying bugs and errors in a program’s code and behaviour is typically a satisfying experience.

Many different automated tools exist to test software products, particular web-based solutions and learners should encounter a wide range of these during their studies, both commercial and open source. Where possible try to link practical testing to actual software development (mobile, desktop, application, web-based or hybrid) taught in other units, possibly even using products from those sessions as ‘live’ case studies.

Ideally, this unit could be delivered alongside, or after, Unit 4: *Programming* and Unit 9: *IT Project Management*. Other potential links include Unit 6: *Website Development*, Unit 7: *Mobile Apps Development*, Unit 8: *Computer Games Development*, Unit 15: *Customising and Integrating Applications* and Unit 19: *The Internet of Things*.

Any opportunities to involve local employers as clients will enhance both the unit and the learner experience, particularly in terms of working with a live software development brief.

Delivering the learning aims

For learning aim A, learners need to understand both the different functions that software needs to perform, and the processes of software development and people/skills engaged in each stage of the process.

Initially, you will focus on the purpose of the software and the user requirements. Then learners should be directed to the process of development and the typical job roles that exist in the field of software development. Coverage of a wide variety of job roles will help learners appreciate the range of skills utilised and, possibly, its career progression pathway.

To cover the features of software testing and methodologies, an independent research may be beneficial. Learner can produce an outline of each feature and methodology and produce informative posters on them – these can be used in the classroom to help others at a later date.

Perhaps the best way to understand user requirements is to examine sample project briefs that demonstrate the functional requirements that a software development team has been expected to resolve. Alternatives include briefs where user requirements are requesting additional or revised features (to resolve problems in an existing system) or simply existing processes that need to be optimised.

An examination of most online IT sector recruitment websites will reveal a wide range of software development roles. Typically each has a detailed job description that not only lists the required technical expertise but also outlines the responsibilities of the post.

Encouraging learners to use these sites as a reference point is recommended. In addition, you will find similar content to brief learners at the Tech Partnership, a network of employers working to create skills for the UK's digital economy. Guest speakers working in these roles in industry (possibly previous learners) are also a useful addition to any presentation.

By the completion of this learning aim learners should understand the typical contents of a user requirements list and the roles connected to the process of testing, including implicit ones such as the actual product owners.

The next sub-topic involves an examination of the different types of software testing, their applications and outputs. The content breaks this range of software testing into nine different sub-categories, although, in truth, you may find some commercial and open source products that span multiple categories. Some examples of available open source windows application testing software include AutoIT and Winium. Learners should be able to name each form of testing, describe its purpose and how each is implemented and who is typically responsible for performing the test. Some simpler test methodologies, e.g. unit testing can be routinely performed in other units, offering opportunities for a more holistic delivery approach. Others such as penetration testing are very involved and have coverage in topics outside the normal software development family of units, e.g. learning aim C, Unit 11: *Cyber Security and Incident Management*.

The final part of learning aim A focuses on evaluating the features of different software development methodologies, specifically including Agile, Waterfall and Kanban. Although others exist, it is advised, given their complexity, to limit coverage to these three named examples. Without suitable frames of reference, this topic can prove to be a very nebulous concept for learners. Having a guest speaker, preferably a developer/programmer or a project manager explaining the methodology they use in their workplaces and how it works on a day-by-day basis during the development of a new product is probably the most effective and grounded approach possible.

Learning aim B primarily focuses on the preparation, planning and physical act of testing a software product. You can simplify this as a four-stage process:

- knowing how to set up common testing tools and processes
- selecting the appropriate test methodology/methodologies for a given software product
- creating a test plan
- testing the product (following the test plan with the appropriate test methodology/methodologies).

Learners should be encouraged to work methodically through this process, documenting each stage completely before continuing. You should create suitable test scenarios (using different test methodologies) for learners to practise before summative assessment occurs. It would be beneficial for learners to keep an ongoing log of testing and results to ensure an organised approach.

It is important in this learning aim that testing is clearly linked to specific client needs and ensuring a product meets these needs. It may be useful for learners to produce evidence for this unit alongside other units, such as Unit 7 *Mobile app development* or Unit 8 *Computer Games development*, so that they have a clear set of user/client needs to work from.

Learning aim C focuses on evaluating the test results and presenting findings to the development team. A suggested generic approach could be:

- compare passed, failed and skipped tests
- identify undocumented bugs (i.e. those not known to development team)
- make evaluation with recommendations
- present the findings (using a variety of communication skills and media)

- add new test cases based on new bugs
- plan regression testing
- repeat the presentation of findings.

Learners should be supported when approaching the challenge of evaluating outcomes effectively as this is typical of a higher-order skill. This could be done by completing a class wide evaluation of a separate set scenario. This will ensure learners are aware of the requirements and processes involved in evaluation. Typically this can become limit grade expectations. Where possible encourage learners to practise their evaluative skills using sample products and tests. In addition, the presentation of findings and the expected mixture of media (written word, verbal and graphical information such as charts and diagrams) can prove challenging. Again, preparation and practice is a key objective here, so learners should be given repeated opportunities to practise the presentation of findings effectively, paying particular attention to their tone, language and the use of jargons. Above all, remind learners that testing is a positive activity and its aim is to improve a product, not be used as a tool to punish developers; a constructive feedback is vital.

Learners should present their results in professional manner, as if to another member of a software development team. This could make use of a verbal presentation supported by a set of slides, or in the form of a narrated screen cast video. For this activity the tutor, may act out the role of a team member. You may also wish to support the evidence by videoing the presentation as well as producing a witness statement describing the quality and effectiveness of the presentation.

Assessment model

Learning aim	Key content areas	Recommended assessment approach
A Understand the software development and testing methodologies commonly used during the development life cycle to quality assure software	**A1** User requirements and typical software project job roles **A2** Characteristics of common software testing methodologies **A3** Features of testing for different software development methodologies	A report into the characteristics of different testing methodologies used in two different software development projects and how the choice of project methodology affects the testing method, software product, user requirements and team members.
B Carry out a range of testing methodologies on a software product to meet a client's needs	**B1** Common tools and processes used in software testing **B2** Selecting appropriate test methodologies **B3** Test plan **B4** Product testing	A portfolio of evidence from testing a software product. The evidence should include an overview of testing tools used, why the methodology was chosen, test plans and the results of product testing.
C Review and present the results from software tests to meet a client's needs and suggest improvements	**C1** Test evaluation and presentation of results **C2** Test plan improvements	A written summative report on the testing carried out and the results of the testing. Statistical analysis on time taken to test bugs and pass/fail/skip rates. Additions to test plan, to include newly found bugs that were not initially considered in the preparation stage.

Assessment guidance

The assessment for this internally assessed unit would benefit from division into three assignments as shown above.

Assignment 1 should cover learning aim A. A thoroughly researched report, evaluating the characteristics of different testing methodologies used in two different software development projects and how the choice of project methodology affects the testing method, software product, user requirements and team members. Alternatively, learners could produce and deliver a presentation on these characteristics and methodologies.

Potentially, software projects could be selected from the current/previous learners' work in other units, e.g. Unit 4: *Programming,* Unit 6: *Website Development* or Unit 7*: Mobile Apps Development*. Opportunities therefore also exist to assess elements of different assessments in an integrative fashion, depending on the delivery timings and unit composition of the learners' programme of study.

Things to remember to offer the best opportunity for learner success: Learners must have access to a developed software program with a clear specification (based on the user requirements).

Learners will need to access a range of both published and online sources to provide the academic content.

Assignment 2 should cover learning aim B. This assessment requires learners to perform a range of testing methodologies on a software product to meet a client's needs. In order to develop a systematic approach to developing a portfolio, learners should keep accurate ongoing logs of the work they carry out.

Things to remember to offer the best opportunity for learner success:

- Access to appropriate hardware, software and tutorials is required in order for learners to test selected software projects. This could include:
    - web services, e.g. SoapUI or a free alternative
    - test management software, e.g. Tarantula (free), and TestLodge and HipTest (commercial and paid for alternatives).
- Learners will need to access a range of both published and online sources to provide the academic content.
- Distinction level learners need to perform a comprehensive and appropriate range of tests systematically and meticulously on a software product against the user requirements, using an effective test plan.

Assignment 3 should cover learning aim C. This assessment requires learners to review and evaluate the testing carried out on a chosen software development project and the results of its testing. They should provide statistical analysis on time taken to test bugs and pass/fail/skip rates.

Additionally, they should identify non-documented bugs and prepare for further regression testing through the identification of reasoned improvements. For higher grades, present a cohesive and clear evaluation of the results of testing using graphical and written methods; compare the passed, failed and skipped tests; and suggest reasoned improvements to the test plan.

Ideally, the written report should be presented by the learner to a member of the development team.

Things to remember to offer the best opportunity for learner success:

- Learners should have the ability to represent numeric data in graphical format using charts and diagrams where appropriate.
- Learners should present their findings in a clear, concise and unambiguous manner that meets the needs of the development team members.
- Learners will need to access a range of both published and online sources to provide the academic content.

Getting started

This gives you a starting point for one way of delivering the unit, based around the recommended assessment approach in the specification.

Unit 13: Software Testing

Introduction

Introduce this unit by ascertaining the learners' experience with programming and using software (via class discussion), particularly their experiences of programs behaving badly, e.g. freezing, crashing, working slowly or calculating the wrong answer and detail the full expectation of the unit's outcomes and the skills (and professional behaviours) it is hoped the learners will develop before its completion.

Providing an overview of the IT sector and the number of entry-level roles available for software developers and software testers is a useful motivating factor; gauging learners' interest levels in pursuing such a route post-qualification can also be beneficial. Learners performing an independent research and feeding back findings may be an engaging approach.

You may consider measuring initial learner skills by using a simple skills and behaviours audit, which details different testing tools, techniques and methodologies listed in the specification, permitting the creation of individualised starting points based on the prior experience. This would help you manage practical sessions more appropriately, spending limited support time in the most efficient manner. Repeat this process at the end of the unit using the same document to measure the distance travelled.

You may also consider appointing (or asking for volunteers) learners with more experience, particularly those with stronger programming skills, to act as classroom support.

Learning aim A: Understand the software development and testing methodologies commonly used during the development life cycle to quality assure software

A1: User requirements and typical software project job roles

You will discuss both the software purposes (i.e. what is it intended to do? How does it improve work flow?) and the job roles in the process with regard to developing, checking etc.

- Outline the basic user requirements; typically this can be achieved by:
    - investigating software development project briefs
    - examining real-world scenarios
    - interviewing potential clients and ascertaining their requirement types, e.g. whether they want a new functionality or problems with the current system, or a specific process improved.
- Ask learners to investigate roles associated with a software development project, including members of its team and its stakeholders. The use of online recruitment agencies or sector-specific job recruitment companies and trade adverts often provides a useful source of reference, along with suitable sector bodies, e.g. The Tech Partnership. Learners could then present their findings to their peers, creating a definitive checklist.
- This learning topic should take approximately two hours.
- Review the learner topic using directed question and answer techniques during a suitable plenary.

A2: Characteristics of common software testing methodologies

You will introduce the different types, application and output of different software testing methodologies by getting learners to research each of them and present the findings to the group.

- Identify each of the different types, e.g. unit testing, acceptance testing and functional testing by presentation and discussion with the class.
- For each type, detail:
  - What each test is designed to prove or identify, e.g. security flaws and meeting client expectations.
  - How each test is created.
  - How each test is performed.
  - Who performs each test (e.g. client, target users and development team).
  - At which part of the software development lifecycle is the testing performed, i.e. not all are auctioned during the traditional testing phase, some could occur during development (e.g. unit testing) and others in post-development (e.g. performance testing).
  - What form the outputs from each type of testing actually take.
- Select a suitable case study with sufficient scope, documentation and available sample code, and demonstrate how to apply each type of tests.
- Ask learners to tackle each type of testing with a similar case study.
- This learning topic should take approximately six hours.
- Review the learner topic using directed question and answer techniques during a suitable ple-nary.

A3: Features of testing for different software development methodologies

You will introduce different software development methodologies and how testing is applied with them, e.g. Agile (scrum) development, Waterfall development and Kanban development. This could be done by presenting different development projects and working through each development methodology as a class – this will enable the class to see the benefits of different approaches.

- Present each software development methodology:
  - describe its basic characteristics
  - discuss its distinct advantages and disadvantages
  - describe when each software development methodology is typically employed
  - describe how testing occurs within each methodology.
- Ask learners to create a poster, podcast or wiki that provides a basic comparative overview of each software development methodology discussed.
- Task learners with examining a number of different businesses and software development needs and ask them to select an appropriate software development methodology, justifying their selections to their peers.
- Reveal model answers and give feedback to learners.
- This learning topic should take approximately eight hours.
- Review the learner topic using directed question and answer techniques during a suitable ple-nary.

Learning aim B: Carry out a range of testing methodologies on a software product to meet a client's needs

B1: Common tools and processes used in software testing

You will detail the common tools and processes used in software testing.

- Demonstrating how different testing tools are used – while giving learners an opportunity to try tools themselves, including:
  - How different tools record information, e.g. text based and automation tools.

-
    - Ask learners to investigate the advantages and disadvantages of each type of testing tools, including ease of use, production of meaningful audit trails and their reporting functionality. Learners could create a table of comparison to summarise their findings or present them to their peers.
    - Demonstrate how automated tools can be used to test developer projects, particularly those under source control, e.g. versioning software such as Git and SVN. For example, this may require the build of a test server with appropriate versioning, hosting software and testing tools for a web-based project.
    - Discuss the role of external testing companies and debate their relative advantages and disadvantages.
- Present a breakdown via the presentation of testing processing which includes:
    - quality assurance processes for dealing with software bugs, e.g. description, steps to reproduce, affected version, fix version, actual result, expected results and importance/severity (minor, major and critical)
    - change requests, including policies
    - debugging of program code.
- For each type of test processes, describe its purpose and where possible demonstrate and encourage learners to duplicate the process. A possible activity flow which links all types could encourage learners to:
    - identify a bug in a software project (the complexity of this can be differentiated for learners' abilities)
    - follow and apply the correct quality assurance process
    - complete a change request
    - debug the program code and fix it
    - review the completed testing processes.
- This learning topic should take approximately five hours.
- Review the learner topic using directed question and answer techniques during a suitable plenary.

B2: Selecting appropriate test methodologies

You will detail and (where possible) demonstrate different test methodologies.

- Identify and describe different types of test methodologies, i.e. Service testing, e.g. Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) services:
    - application testing
    - automation testing
    - regression testing.
- Demonstrate different test methodologies using commercially available or open source suites and tools. The reputational complexity of the tool should not be a limiting factor here; choose a suite or tool that is most accessible for learners, allowing the core principles of the methodology to be easily highlighted, e.g. Selenium. This is a popular and portable software testing framework for web applications, released under the Apache 2.0 license that can be installed and configured within a reasonable timeframe.
- For each test methodology, it is important to distil its most significant features and characteristics, particularly in terms of linking these to possible applications.
- This learning topic should take approximately five hours.
- Review the learner topic using directed question and answer techniques during a suitable plenary.

B3: Test plan

You will demonstrate the correct contents of a test plan, including the selection of the test methodologies most suitable for a given software project and the creation and documentation of suitable test scripts.

- For a selected sample software development project, walk learners through the process of selecting the most appropriate test methodology and for each of these, the content of each test script including:
    - title, e.g. 'Viewing paginated transactions for a given customer account'.
    - description of the test case, e.g. 'Test script to see if the bug fix for customer account pagination is working correctly'
    - steps required to produce the test case, e.g. sequence or combinations of user actions (e.g. for a customer portal: logging in, selecting a customer account and viewing transactions) and/or situations that need to exist (e.g. a customer account having more than 100 transactions)
    - expected result, e.g. transactions are displayed for the selected account in batches of 10, with a navigable page list displayed. Customers are then able to freely move forward or backward through the transaction list without any errors
    - actual result, e.g. what occurred, screen capture, video capture or observation report
    - importance of test, e.g. why the test has been chosen, e.g. to illustrate its importance within the functionality of the project (e.g. it's a basic customer function) or to demonstrate the resolution of a previous bug (e.g. there was no effective pagination previously).
- Pair learners and task them with creating a suitable series of test scripts for a sample software project, identifying typical areas to test and completing the appropriate details for each.
- Ask learners to review their peers' efforts.
- Give feedback, highlighting any gaps in coverage, e.g. significant logical/functional areas not identified.
- This learning topic should take approximately three hours.
- Review the learner topic using directed question and answer techniques during a suitable plenary.

B4: Product testing

You will discuss the concept of product testing, particularly in reference to:

- acceptance testing, including breaking down a specification into a number of user requirements, each identified by a unique number
- regression testing
- role-play elements of acceptance testing with learners playing the role of project clients and testers.

- Practise regressing testing using a sample software development project.
- Learners should exhibit a systematic approach to testing using a test plan.
- This learning topic should take approximately four hours.
- Review the learner topic using directed question and answer techniques during a suitable plenary.

Learning aim C: Review and present the results from software tests to meet a client's needs and suggest improvements

C1: Test evaluation and presentation of results

You will discuss and lead a session for learners, demonstrating how to review and present the results from software tests. Detail each of the items that would be expected in the test evaluation and presentation of the results, e.g.:

- A summary that includes an overview which tests were passed, which failed and those that were skipped (and the reasons why).
- Raising the issue of undocumented bugs which testing has revealed
- How to present effectively, utilising good communication skills, paying attention to:
    - Ensuring that communication is clear, whether delivered verbally or in written format and there is no room for ambiguity in statements made. Provide examples of good and bad statements and ask learners to differentiate them and justify their choices. Ask learners how to avoid ambiguity. Demonstrate possible solutions, e.g. keeping sentences short, using words consistently throughout and placing all adjectives close to the words they modify.
    - Considering the use of the best way to include quantitative data, e.g. visually as pie charts, bar charts or line graphs rather than as a table or a dense paragraph of text. The best way to demonstrate this is to give learners a detailed (i.e. 'wordy') written extract to present and then simply replace this with a chart that illustrates the intended points or comparisons much more clearly and concisely.
    - Discuss how documentation should be recorded, the difference between informal and formal reporting techniques.
    - Demonstrate how visual aids may be used to enhance presentation use, e.g. test captures, animations, led-pointers and pre-prepared handouts.
    - Differentiate for learners the communication requirements of one-to-one and group-based presentations in both informal and formal situations. Draw on learners' personal experiences to identify occasions where they have faced similar challenges and how they managed them. Suggest possible improvements they could make and consider how this could affect the outcome.
    - Ask learners to identify the tone and language used in two sample feedback extracts, one that uses a positive tone and the other that engages in negative rhetoric. Remember to pepper one sample with unnecessary jargons for learners to identify and recommend suitable alternatives for different audience types.
    - Challenge learners to give peers the feedback on their work in session, teasing out the need to be supportive, constructive and managing the tone of the conversation.
- This learning topic should take approximately three hours.
- Review the learner topic using directed question and answer techniques during a suitable plenary.

C2: Test plan improvements

You will present how test plan improvements are discussed with the development team and detail the possible resulting jobs.

- Using a suitable software development project/case study:
    - Demonstrate that undocumented bugs will generate new test cases that need to be resolved.
    - Show how learners can prepare for regression testing, adding new test cases to the test suite based on the newly documented bugs.
    - Explain how testing feedback should be fed back to the development team, focusing on test results (passed, failed and skipped), format of findings and communication skills.
- Ask learners to work in small groups to review their case study's test results, identify new test cases and add them to a test suite and feed back to their peers. An alternative approach might be to ask groups to review actual software development projects created by other groups of

learners and use this for the testing and feedback phase.

- Learners should individually produce a clear and cohesive analysis and evaluation of their plans. Learners should be made aware, via a whole class activity, of the requirements of this. As a class analyse and evaluate a set test plan – ensure learners are aware that this is simply not description and suggestion – the plan needs to be fully evaluated – what are its strengths and weaknesses, how could it be improved etc..
- You should now reissue the skills and behaviours audit completed by learners in the first session so that they can now revisit the document and make additional observations about where and how they feel they have improved. Learners may also find it useful to reference their ILPs when completing this task.
- This learning topic should take approximately five hours.
- Review the learner topic using directed question and answer techniques during a suitable plenary.

Details of links to other BTEC units and qualifications, and to other relevant units/qualifications

This unit links to:

- Unit 4: Programming
- Unit 6: Website Development
- Unit 7: Mobile Apps Development
- Unit 8: Computer Games Development
- Unit 14: Customising and Integrating Applications
- Unit 18: The Internet of Things.

Resources

In addition to the resources listed below, publishers are likely to produce Pearson-endorsed textbooks that support this unit of the BTEC Internationals in Information Technology. Check the Pearson website at http://qualifications.pearson.com/endorsed-resources for more information as titles achieve endorsement.

Textbooks

- Hambling B (ed.), Samaroo A, Morgan P, Thompson G and Williams P, *Software Testing: An ISTQB-BCS Certified Tester Foundation Guide* (Third Edition), BCS Learning & Development Limited, 2015 ISBN 9781780172996.
- Kaner C, Bach J and Pettichord B, *Lessons Learned in Software Testing*, Wiley, 2001 ISBN 9780471081128.
- Patton R, *Software Testing* (Second Edition), Sams, 2005 ISBN 9780672327988.
- Whittaker J A, Arbon J and Carollo J, *How Google Tests Software,* Addison Wesley, 2012, ISBN 9780321803023.

Journals

- *Journal of Software Engineering Research and Development* – https://jserd.springeropen.com/.
- *Test Magazine* – http://www.testingmagazine.com/.

Videos

- Popular videos – acceptance testing (https://www.youtube.com/watch?v=rj4q7F2DgJ0&list=PLvGFpz7ufL_4LJySYXgaObJnQPNVUzprW).
- Manual Testing Training – *What is Test Case – Example* (https://www.youtube.com/watch?v=mjB9XTpQmgc).
- Selenium IDE Demo – *Quick Beginner's Tutorial (https://www.youtube.com/watch?v=gsHyDIyA3dg).*
- *How to write a Test Case* (https://www.youtube.com/watch?v=BBmA5Qp6Ghk).
- *Testing a RESTful Web Service with SOAP UI* (https://www.youtube.com/watch?v=7YpJS--BqiI)
- *What is Software Testing & Why Testing is Important?* (https://www.youtube.com/watch?v=TDynSmrzpXw).
- *Introduction to Scrum* – 7 minutes (https://www.youtube.com/watch?v=9TycLR0TqFA).

Websites

- *https://www.associationforsoftwaretesting.org/* – Association for Software Testing
  www.cwjobs.co.uk – CWJOBS.

- *www.seleniumhq.org* – SeleniumHQ – A specialist IT recruitment website.
- *www.tutorialspoint.com* – tutorialspoint – Web browser automation tools used for testing via scripting.
- *www.testiatarantula.com* – Tarantula test management – Online programming and testing tutorials and resources.
- *www.thetechpartnership.com/*– The Tech Partnership – An open source tool for managing software testing in agile software projects.
- *https://www.thetechpartnership.com/tech-future-careers/what-is-it-like/job-roles/*– Job Roles – A network of employers working to create skills for the UK's digital economy.