Higher National Unit specification

General information

Unit title: Software Development: Implementation and Testing (SCQF level 9)

Unit code: HA4H 36

Superclass: CB

Publication date: January 2016

Source: Scottish Qualifications Authority

Version: 01

Unit purpose

The purpose of this Unit is to introduce learners to the planning of software development projects, the use of libraries and Application Programming Interfaces (APIs) in software development and the use of code repositories.

Project planning will be carried out in line with the Agile project management process. APIs will be used to interface with major platforms. Libraries covered will include standard and add-on libraries for the relevant platform. Version control will be accomplished by the use of code repositories.

Learners will develop their programming skills by planning, designing, implementing and testing practical solutions using an appropriate software development environment.

On completion of the Unit, learners will have gained knowledge and experience of implementing complex programs using an object-oriented programming language and making use of APIs, libraries and code repositories.

Learners who have completed both this Unit and *Software Development: Analysis and Design* (SCQF level 9) could progress to the Unit *Software Development: Project* (SCQF level 9).

Outcomes

On successful completion of the Unit the learner will be able to:

1 Describe object-oriented programming techniques and approaches to software testing.
2 Develop and modify programs that make use of APIs, frameworks and libraries.
3 Maintain version control by the use of code repositories.
4 Manage the software testing process.

Higher National Unit Specification: General information (cont)

Unit title: Software Development: Implementation and Testing (SCQF level 9)

Credit points and level

2 Higher National Unit credits at SCQF level 9: (16 SCQF credit points at SCQF level 9)

Recommended entry to the Unit

Entry to this Unit is at the discretion of the centre. However, it would be beneficial if learners had prior knowledge and skills in computer programming.

It would be beneficial if learners had some prior experience of software development, and analysis and design tools that could be evidenced by having achieved the Higher National Unit HA4G 35 *Software Development: Implementation and Testing* (SCQF level 8) or equivalent.

Core Skills

Opportunities to develop aspects of Core Skills are highlighted in the Support Notes for this Unit specification.

There is no automatic certification of Core Skills or Core Skill components in this Unit.

Context for delivery

If this Unit is delivered as part of a Group Award, it is recommended that it should be taught and assessed within the subject area of the Group Award to which it contributes.

The Assessment Support Pack (ASP) for this Unit provides assessment and marking guidelines that exemplify the national standard for achievement. It is a valid, reliable and practicable assessment. Centres wishing to develop their own assessments should refer to the ASP to ensure a comparable standard. A list of existing ASPs is available to download from SQA's website **(http://www.sqa.org.uk/sqa/46233.2769.html)**.

Equality and inclusion

This Unit specification has been designed to ensure that there are no unnecessary barriers to learning or assessment. The individual needs of learners should be taken into account when planning learning experiences, selecting assessment methods or considering alternative evidence.

Further advice can be found on our website **www.sqa.org.uk/assessmentarrangements**.

Higher National Unit specification: Statement of standards

Unit title: Software Development: Implementation and Testing (SCQF level 9)

Acceptable performance in this Unit will be the satisfactory achievement of the standards set out in this part of the Unit specification. All sections of the statement of standards are mandatory and cannot be altered without reference to SQA.

Where evidence for Outcomes is assessed on a sample basis, the whole of the content listed in the Knowledge and/or Skills section must be taught and available for assessment. Learners should not know in advance the items on which they will be assessed and different items should be sampled on each assessment occasion.

Outcome 1

Describe object-oriented programming techniques and approaches to software testing.

Knowledge and/or Skills

- Describe object-oriented programming concepts
- Describe APIs, frameworks and libraries
- Describe the differences between object-oriented and traditional approaches to software development
- Describe algorithms and data structures
- Describe approaches to software testing
- Describe the use of input validation to ensure data security

Outcome 2

Develop and modify programs that make use of APIs, frameworks and libraries.

Knowledge and/or Skills

- Develop programs using APIs
- Develop programs using frameworks
- Develop programs using libraries
- Modify programs that make use of APIs, frameworks or libraries

Outcome 3

Maintain version control by the use of code repositories.

Knowledge and/or Skills

- Store source code in code repositories
- Use version control systems to record changes
- Use hosting facilities to host repositories

Higher National Unit specification: Statement of standards (cont)

Unit title: Software Development: Implementation and Testing (SCQF level 9)

Outcome 4

Manage the software testing process.

Knowledge and/or Skills

- Produce test plans, estimates and strategies
- Monitor and control the testing process
- Improve testing by the use of configuration management
- Apply risk analysis and risk management techniques to testing

Evidence Requirements for this Unit

Learners will need to provide evidence to demonstrate their Knowledge and/or Skills across all Outcomes. The Evidence Requirements for this Unit will take two forms:

1 evidence of cognitive competence (for Outcomes 1, 2, 3 and 4).
2 evidence of practical competence (for Outcomes 2, 3 and 4).

Please note that Outcome 1 covers only cognitive competences while Outcomes 2, 3 and 4 cover both cognitive and practical competencies. Each of the Knowledge and/or Skills items listed in Outcomes 2, 3 and 4 is practical in nature but has an underlying cognitive competence that needs to be evidenced.

The evidence of cognitive competence will be the definitions, descriptions and explanations required for Outcome 1. The evidence of practical competence will be the application of planning and programming skills to specific problems required for Outcomes 2, 3 and 4.

The evidence of practical competence (Outcomes 2, 3 and 4) may relate to **one or more** problems. Candidates should apply planning, programming and testing techniques. The evidence would consist of a project plan, program code and evidence of successful execution and test documentation. All of the Knowledge and Skills statements for Outcomes 2, 3 and 4 must be evidenced.

Evidence is normally required for all of the Knowledge and Skills in every Outcome. This means that every Knowledge and Skills statement must be evidenced. However, sampling may be used in a specific circumstance (see below).

The amount of evidence should be the minimum consistent with the defined Knowledge and Skills. For Outcome 2, it is sufficient to develop **at least one** program that makes use of APIs, Frameworks and Libraries. For Outcome 3, it is sufficient to maintain version control for **at least one** software project. For Outcome 4, it is sufficient for the candidate to manage the software testing process for **at least one** software project. At this level, the programs should be complex in terms of their functionality, algorithms and data structures.

Evidence may be wholly or partly produced under controlled conditions. When evidence is produced in uncontrolled or loosely controlled conditions it must be authenticated. The *Guide to Assessment* provides further advice on methods of authentication.

Higher National Unit specification: Statement of standards (cont)

Unit title: Software Development: Implementation and Testing (SCQF level 9)

There are no time limitations on the production of evidence (but see exception below). The evidence may be produced at any time during the life of the Unit. Candidates may use reference materials when undertaking assessment (but see exception below).

Sampling is permissible when the evidence of cognitive competence for Outcomes 1, 2, 3 and 4 is produced by a test of knowledge and understanding. The test may take any form (including oral) but must be supervised, unseen and timed. The contents of the test must sample broadly and proportionately from the contents of Outcomes 1, 2, 3 and 4 with approximately equal weighting for each Outcome. Access to reference material is not appropriate for this type of assessment.

The Guidelines on Approaches to Assessment (see the Support Notes section of this specification) provides specific examples of instruments of assessment.

Higher National Unit Support Notes

Unit title: Software Development: Implementation and Testing (SCQF level 9)

Unit Support Notes are offered as guidance and are not mandatory.

While the exact time allocated to this Unit is at the discretion of the centre, the notional design length is 80 hours.

Guidance on the content and context for this Unit

Outcome 1

Learners should know how to develop programs by selecting and using a combination of appropriate constructs, data types, data structures and algorithms. They should also know how to use subprograms and parameter passing.

They should be able to describe software testing methodologies including testing methods (static and dynamic, white and black box) testing levels (Unit, integration, component, system testing), test plan (exceptional, extreme and normal data), debugging techniques (dry runs, walkthroughs, breakpoints, trace tables) and types of errors (syntax, execution, logic).

Learners should also know how to test digital solutions including meaningful identifiers, indentation, providing internal commentary, Unit, integration, component and system testing, using own test data and test plan (exceptional, extreme and normal data). They should be aware that input validation should be used to ensure that only correct data is accepted and to prevent the injection of malicious code.

Outcome 2

Program development time can be reduced significantly by making use of pre-existing code provided by APIs, frameworks and libraries.

A **Library** consists of code fragments that can be called from a program, to help do things more quickly/easily. For example, a Bitmap Processing library could provide code for loading and manipulating bitmap images, saving programmers having to write that code themselves. A Framework is a big library that provides many services (rather than only one focussed ability as most libraries do). For example.NET is an application framework — it provides of the services needed to write a vast range of applications — so one ‘library’ provides support for almost everything required. A framework often supplies a base on which programmers can build their own code, rather than building an application that consumes library code.

An API (Application Programming Interface) is the functions/methods in a library that you can call upon to do things for you — it the interface to the library.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Implementation and Testing (SCQF level 9)

Outcome 3

A code repository is a file archive where source code for software and web pages can be kept, publicly or privately. Code repositories are often used by open-source projects and other multi-developer projects. They help developers submit changes to code in an organised manner. Code repositories often support version control, bug tracking, release management, mailing lists, and wiki-based documentation.

Version control is the management of changes to documents, computer programs, large web sites, and other collections of information. Changes are normally identified by a number or letter code, known as the 'revision number' or 'revision level'. Each revision is associated with a timestamp and details of the person making the change. Revisions can be compared, restored, and with some types of files, merged. Version Control Systems (VCS) are widely used in software development, where a team of people may change the same files.

Version control systems often run as stand-alone applications, but they are also embedded in various types of code repositories. Version control provides the ability to revert a document to a previous revision, track edits, correct mistakes and defend against malicious changes.

Learners should be aware that programs often require to be modified in order to meet changing requirements or to correct newly-discovered bugs.

Code repository hosting systems vary widely in the facilities provided. A comprehensive comparison can be found at:

https://en.wikipedia.org/wiki/Comparison_of_source_code_hosting_facilities

and another at:

https://blog.profitbricks.com/top-source-code-repository-hosts/

GitHub is one well-known Web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration.

It also provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

GitHub offers a Student Developer Pack which offers students free access to popular development tools and services: https://education.github.com/pack.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Implementation and Testing (SCQF level 9)

Outcome 4

Learners should be aware of the fundamentals of test planning and estimation. They should know the reasons for writing test plans and be able to explain how test plans relate to projects, test levels, test targets and test execution. They should know the factors that affect the effort involved in testing, including test strategies and how they affect testing. They should be able to explain how metrics, expertise and negotiation are used for estimating and be familiar with the terms entry criteria, exit criteria, exploratory testing, test approach, test level, test plan, test procedure and test strategy.

Learners should be able to explain the essentials of test progress monitoring and control. They should know the common metrics that are captured, logged and used for monitoring and be able to analyse, interpret and explain test metrics. They should be familiar with test status reports, test summary reports and test logs and should know the terms defect density, failure rate, test control, test coverage, test monitoring and test report.

Learners should understand the basics of configuration management as they relate to testing and should be able to explain how good configuration management assists testing. They should know the terms configuration management and version control.

Learners should be able to explain how risk and testing relate and should know that a risk is a potential undesirable or negative Outcome. They should be aware of likelihood and impact as factors determining the importance of a risk. They should be able to describe the use of risk analysis and risk management for testing and test planning and should know the terms product risk, project risk, risk and risk-based testing.

Guidance on approaches to delivery of this Unit

This Unit is a component of the PDA Software Development (SCQF) level 9. It should be delivered after, or in parallel with the Unit *Software Development: Analysis and Design* (SCQF level 9). Both of these Units should be completed before delivery of the Unit *Software Development: Project* (level 9).

The Outcomes may be delivered in the order in which they are written. They have been written with a learning sequence in mind.

The actual distribution of time between Outcomes is at the discretion of the centre. However, one possible approach is to distribute the available time as follows:

Outcome 1: 20 hours
Outcome 2: 20 hours
Outcome 3: 20 hours
Outcome 4: 20 hours

It is anticipated that the required concepts will be introduced by the teacher and reinforced by appropriate examples.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Implementation and Testing (SCQF level 9)

There is significant scope in this Unit to illustrate concepts and skills with case studies of implementation and testing. The majority of time in this Unit will be spent on the practical application of the theoretical aspects of the Unit.

Throughout this Unit, learner activities should relate to their vocational interests.

Guidance on approaches to assessment of this Unit

Evidence can be generated using different types of assessment. The following are suggestions only. There may be other methods that would be more suitable to learners.

A traditional approach to assessment would involve an end of Unit test of the knowledge and understanding (Outcomes 1, 2, 3 and 4) and a practical assessment of practical abilities (Outcomes 2, 3 and 4).

The end of Unit test would sample from the knowledge and understanding contained in Outcomes 1, 2, 3 and 4. The test could consist of selected response or constructed response questions. A selected response test could comprise a number of multiple-choice questions (MCQs) or multiple-response questions (MRQs), and would be marked and assessed traditionally. For example, the test could comprise 40 multiple-choice questions, each of which could have four options (A–D), distributed equally across all the Outcomes, with an appropriate pass mark. This test would be taken, sight-unseen, in controlled and timed conditions without reference to teaching materials. A suitable duration could be 60 minutes. Given the level of this Unit (SCQF level 9), the test would comprise questions spanning a range of cognitive skills (including higher level ones involving analysis and synthesis).

Practical assessment (Outcomes 2, 3 and 4) could involve the practical application of the skills taught in each Outcome. A single assignment could be used to cover all three Outcomes or a separate assignment may be used for each. The assignment(s) could be assessed holistically, without a marking scheme and not assigned a specific score, and given a simple ‘pass/fail’ grade. All of the Knowledge and Skills would be evidenced in this assessment. There would be no time limitations (beyond the practicality of completing the Unit within the scheduled timetable) for this assessment.

A more contemporary approach to assessment could use a web log to record learning throughout the life of the Unit. If this approach is taken, then sampling would not be appropriate. The blog would contain evidence for all Knowledge and Skills statements. The blog would record, on a daily or weekly basis, the learning that has occurred. It would contain textual definitions, descriptions and explanations as required by the Knowledge and Skills statements in the Outcomes (all Outcomes), including hyperlinks and embedded multimedia (audio, graphic or video).

The blog could encompass all Outcomes, including Outcome 2, 3 and 4 (which are practical). This could be done by the blog demonstrating how implementation and testing techniques could be applied to one or more problems. Given that the blog would, most likely, be completed at various times and locations throughout the life of the Unit, some form of authentication would be necessary. There would be no time limitation on the completion of the blog since it would be done on an on-going basis throughout the life of the Unit.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Implementation and Testing (SCQF level 9)

Centres are reminded that prior verification of centre-devised assessments would help to ensure that the national standard is being met. Where learners experience a range of assessment methods, this helps them to develop different skills that should be transferable to work or further and higher education.

Opportunities for e-assessment

E-assessment may be appropriate for some assessments in this Unit. By e-assessment we mean assessment which is supported by Information and Communication Technology (ICT), such as e-testing or the use of e-portfolios or social software. Centres which wish to use e-assessment must ensure that the national standard is applied to all learner evidence and that conditions of assessment as specified in the Evidence Requirements are met, regardless of the mode of gathering evidence. The most up-to-date guidance on the use of e-assessment to support SQA's qualifications is available at **www.sqa.org.uk/e-assessment**.

Opportunities for developing Core and other essential skills

There is no automatic certification of Core Skills or Core Skill components in this Unit. The Unit provides opportunities for the development of Computational Thinking skills,

History of changes to Unit

Version	Description of change	Date

General information for learners

Unit title: Software Development: Implementation and Testing (SCQF level 9)

This section will help you decide whether this is the Unit for you by explaining what the Unit is about, what you should know or be able to do before you start, what you will need to do during the Unit and opportunities for further learning and employment.

The purpose of this Unit is to introduce you to the use of libraries and Application Programming Interfaces (APIs) in software development, the use of code repositories and the management of the software testing process.

Libraries and frameworks covered will include standard and add-on libraries for the relevant platform. Version control will be accomplished by the use of code repositories.

The management of testing includes test planning and estimation, test progress monitoring and control, configuration management as it relates to testing the relationship between risk and testing.

You will develop their programming skills by planning, designing, implementing and testing practical solutions using an appropriate software development environment.

You may be assessed in various ways, including multiple-choice questions relating to the theoretical knowledge covered in the Unit, and practical exercises applying the implementation and testing skills learned.

On completion of the Unit, you will have gained knowledge and experience of implementing complex programs using an object-oriented programming language, making use of APIs, libraries, frameworks and code repositories and managing software testing.

Once you have completed both this Unit and *Software Development: Analysis and Design* (SCQF level 9) you can progress to the Unit *Software Development: Project* (SCQF level 9).