Higher National Unit specification

General information

Unit title: Software Development: Project (SCQF level 9)

Unit code: HA4L 36

Superclass: CB

Publication date: January 2016

Source: Scottish Qualifications Authority

Version: 01

Unit purpose

The purpose of this Unit is to apply skills and knowledge of software analysis, design, implementation and testing to produce a software product composed of multiple sub-programs in a "real world" teamwork scenario. The Unit is aimed at learners looking to extend their existing qualifications to enable them to take a role as an entry-level software developer.

Learners will work in a team to determine the scope, plan the development of a complex software project, design the operation and interaction of its components, produce working code to meet the requirements, and test a completed solution to prove functional operation, while following methodical project management practices and incorporating software design patterns. The project will include interaction with third-party APIs, and management of source code with version control software.

The Unit is suitable for learners who have completed the Units *Software Development: Analysis and Design* (SCQF level 9) and *Software Development: Implementation and Testing* (level 9) or a prior learning or experience of software development. The Unit involves the practical application of skills, and will give learners exposure to the whole software development life-cycle while working with others to produce working software.

On completion of the Unit, learners will be competent in team working with others in the production of complex software products or move onto a degree level course in Software or progress to an entry-level software development position.

Higher National Unit Specification: General information (cont)

Unit title: Software Development: Project (SCQF level 9)

Outcomes

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

1 Plan the development of a complex software product as part of a team.
2 Design the structure of a complex software product using object-oriented programming techniques as part of a team.
3 Develop a complex software product using an object-oriented programming language as part of a team.
4 Test the operation and acceptance of a complex software product as part of a team.

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

The Unit is suitable for learners who have completed the Units *Software Development: Analysis and Design* (SCQF level 9) and *Software Development: Implementation and Testing* (level 9) or have prior learning or experience of software development. The Unit involves the practical application of skills, and will give learners exposure to the whole software development life-cycle as part of a team.

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: Project (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.

Outcome 1

Plan the development of a complex software product as part of a team.

Knowledge and/or Skills

- Apply a methodical approach to team project management
- Apply contemporary development approach
- Gather requirements information
- Prioritise requirements
- Validate acceptance criteria for product
- Formulate test plan

Outcome 2

Design the structure of a complex software product using object-oriented programming techniques as part of a team.

Knowledge and/or Skills

- Produce wireframe designs
- Produce system interaction diagrams
- Produce object diagrams
- Identify appropriate design patterns
- Write pseudocode
- Select algorithms

Higher National Unit specification: Statement of standards (cont)

Unit title: Software Development: Project (SCQF level 9)

Outcome 3

Develop a complex software product using an object-oriented programming language as part of a team.

Knowledge and/or Skills

- Use version control software to manage version history
- Build working software
- Structure code idiomatically
- Apply object-oriented programming to meet design
- Apply appropriate software design patterns
- Process input according to design requirements
- Integrate external APIs for data or service provision
- Interact with data persistence
- Output results and feedback to user

Outcome 4

Test the operation and acceptance of a complex software product as part of a team.

Knowledge and/or Skills

- Check operation of code using a range of techniques
- Ensure acceptance criteria are met
- Measure coverage of tests
- Diagnose causes of errors
- Correct identified errors

Evidence Requirements for this Unit

For the purposes of definition in this Unit a 'complex software product' will have multiple and configurable functionality that demonstrates a broad range of object-oriented implementation such as polymorphism, inheritance and/or modularity. The product may interface with multiple APIs and sources of data persistence.

Learners will need to provide evidence to demonstrate their Knowledge and/or Skills across all Outcomes. All evidence will be in the form of practical competence.

Evidence of practical competence will take the form of **at least one component of a complex software product**. There is no requirement for separate evidence for cognitive and practical competence.

Higher National Unit specification: Statement of standards (cont)

Unit title: Software Development: Project (SCQF level 9)

The product must be sufficient to demonstrate the Knowledge and Skills in each Outcome. Candidates must develop **at least one component of a complex software product**. The product should be error-free. The following evidence is the **minimum** required:

1 Software development plan
2 Test plan
3 System design
4 Component designs
5 Source code
6 Object code
7 Test logs

This evidence may be supplied on paper or digitally or a combination of these.

The standard of the evidence should be consistent with the SCQF level of this Unit. At this level, the evidence should collectively or individually demonstrate:

- an understanding of the scope and defining features of the software development process and an integrated knowledge of its main areas and boundaries.
- advanced computational thinking.
- knowledge of one or more software development specialisms that is informed by forefront developments.
- a critical understanding of core theories, concepts, principles and terminology.
- using a few skills, techniques, practices and/or materials that are specialised and/or advanced.
- convey complex program designs in well-structured and coherent form.
- practise in a range of professional level contexts that include a degree of unpredictability.
- seeking guidance where appropriate, manage ethical and professional issues in accordance with current professional and/or ethical codes or practices.

There are no **time limitations** on the production of evidence. The evidence may be produced at any time during the life of the Unit. Learners may use reference materials when undertaking assessment.

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.

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: Project (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

This Unit may be delivered as a stand-alone Unit or following completion of the two other Units making up the Professional Development Award (PDA) in Software Development at SCQF level 9.

Learners should be encouraged to actively explore various hard-copy and internet resources including blog posts, tutorial guides and question and answer sites in order to familiarise themselves with a wide range of methodologies and tools as appropriate to each Learning Outcome. In addition, they must be aware of the internet safety, security, confidentiality and health and safety procedures of the organisation. It is also important to maintain the security and confidentiality of data and information and therefore leaners should be encouraged to back up and check for viruses on a regular basis in order to reduce the risk of loss of evidence of their achievement.

The overall aim of this Unit is for learners to apply skills and knowledge of software analysis, design, implementation and testing to produce a complex software product composed of multiple sub-programs, as part of a team. The focus of the Unit is on practical competencies and learners will analyse a problem, design the operation and interaction of program components, produce working code to meet requirements, and test a completed solution to prove functional operation, while working as part of a team throughout the whole development life-cycle.

It is anticipated that substantial amount of time will be spent on practical tasks, and although learners may use tools that they have sourced themselves; appropriate tools to complete all of the Outcomes should be made available through their centre.

Learners must work as part of a team; each individual evidencing the key Knowledge and Skills to achieve this Outcome. The team will work with tools and processes commensurate with the contemporary software industry. Learners will prepare for entry into the workplace with experience from their qualification that maps to ‘real-world’ problems that are representative of common business practices.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Project (SCQF level 9)

Outcome 1

This Outcome allows learners to work as part of a team applying appropriate analysis and planning techniques to produce a plan for development of a complex software product. Planning should demonstrate understanding of an appropriate software development process, using a contemporary development methodology such as Agile, Scrum or Kanban. Learners are expected to produce planning documents for gathering information, prioritising requirements and creating acceptance test plans.

The overall intention is that work completed in Outcome 1 will be taken into design in Outcome 2.

Outcome 2

The main purpose of this Outcome is for learners to design the structure of their product and the operation of their code for a complex software product based on object-oriented programming techniques as part of a team.

Learners will demonstrate the visualisation of their product through design artefacts such as wireframes, system interaction and object diagrams. Learners should have gained an understanding of the importance of using recognised solutions to common problems such as incorporating design patterns into their planning. Learners will use pseudocode to demonstrate their choice of algorithms and code structure for the operation of their intended product

The overall intention is that their planning will be taken into implementation in Outcome 3.

Outcome 3

The purpose of this Outcome is for learners to develop a complex software product using an object-oriented programming language as part of a team

Learners will turn the design of their product into a working software product using an object-oriented language and object-oriented design practices. Pseudocode from the design phase should be turned into well-structured working code using a combination of appropriate objects, constructs, sub-programs and functions to meet design requirements.

The choice of object-oriented language is not specified, but learners should use idioms appropriate to their chosen language, and implement design patterns to reuse recognised solutions.

Learners will work in a team using source code control to track changes and distribute code amongst the team members.

The product should interact with users and data storage offering multiple and configurable functionality. The product should be integrated with a minimum of one third-party API and demonstrate use of code libraries. User inputs should be validated against agreed criteria, and appropriate results/feedback outputted to the users.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Project (SCQF level 9)

The overall intention is that the product implementation in Outcome 3 can be used as a starting point for Outcome 4.

Outcome 4

This Outcome allows learners to apply their Knowledge and Skills to test the operation and acceptance of a complex software product as part of a team.

Learners will demonstrate their ability to check the operation of code to ensure agreed criteria are met using a range of techniques. This may include testing against their own test plan, manual testing to a use-case plan, or automated testing using 'Unit' or 'integration' or other approaches to testing code.

Learners should demonstrate that they have completed a minimum of 50% testing of the core code structure of their product, including the integration with a third-party API. Learners need to be able to decipher error messages and identify what file/function/line is the source of the error and then to suggest alternative solutions and correct the code accordingly.

Learners could write their own test plan based on their planning and design in Outcomes 1 and 2 If learners cannot identify errors in their own code, eg in the product created to meet previous Outcomes, then they will need to be provided with errors in code of equal complexity to test against.

Guidance on approaches to delivery of this Unit

The Outcomes in this Unit should be delivered once learners have completed learning equivalent to the *Analysis and Design* and *Implementation and Testing* Units at level 9. It would be expected that the delivery of this Unit's Outcomes progresses sequentially and that assessment would be undertaken at the end of each Outcome, eg learners should be confident in their analysis and design skills before completing implementation and then testing.

It is recommended that learners be guided through the full life-cycle of a number of increasingly complex software development projects as part of learning and teaching prior to assessment. This will allow learners to gain confidence in the application of their skills with more independence.

Throughout the Unit, the focus of projects could be on software products that have a relevance to the learner and to their further learning. Products could be anything from a calendar-driven appointment scheduling application through to a 'price-comparison' website.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Project (SCQF level 9)

Guidance on approaches to assessment of this Unit

Evidence Requirements for this Unit will be for practical competence. 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.

The assessment for all Outcomes could comprise a series of practical tasks completed over a period of time. It is recommended that a holistic assessment covering all four Outcomes is used. Learners may have access to online resources, and the assessment need not be done under supervision, but all work should be authenticated as the learner's own. This could be done by oral questioning of the learner and/or observation of some aspects of the learner's work.

For each Outcome, learners should devise a product definition of an appropriately complex software product. Learners should use the same product for all Outcomes, but alternative evidence may be used where this is not possible.

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.

Outcome 1: Plan the development of a complex software product as part of a team.

Learners should devise their own complex product definition as part of a team.

- **Apply a methodical approach to team project development**

  Learners should agree protocols to facilitate communication between themselves, to organise their workloads and coordinate their schedules. This could be evidenced by screenshots of a range of communication channels, such as electronic communication, online project management tools, or written outlines of agreed processes.

  Each learner should demonstrate adherence to the agreed methodology by evidence drawn from various points throughout the project life-cycle.

- **Apply contemporary development approach**

  Learners should demonstrate use of a contemporary software development methodology, such as Agile, to outline the development process for a complex software product.

- **Gather and prioritise requirements information and produce acceptance criteria for product**

  Learners should describe how the product will function for different types of users and journeys through the system. Their findings should be broken down into individual requirements, prioritised to indicate a sequence of implementation. Each requirement should include information about its acceptance criteria.

- **Formulate test plan**

  Learners should prepare an acceptance test plan to meet agreed criteria for their product using appropriate software testing methodologies.

Assessment could take the form of a demonstration of a comprehensive project plan using a project management software tool, or a paper-based method such as index cards or post-it notes.

Outcome 2: Design the structure of a complex software product as part of a team.

Learners may develop the product scoped in Outcome 1, or develop a product of equal complexity.

- **Produce wireframe designs**

  Wireframe designs should indicate both the expected user interface and user interaction, output and feedback, and how configuration could be managed. There should be appropriate wireframes for the major cases of functionality and for edge-case scenarios.

  Assessment could take the form of an 'elevator-pitch' style presentation, a written report format or be presented in a diagrammatic style.

- **Produce system interaction diagrams**

  Learners should be able to demonstrate the complete operation or flow-through the input/outputs of their product. Assessors should select two interactions between code and user input or output and ensure learners have produced detailed structure charts or system interaction diagrams for them. At least one of these should incorporate functionality of a third-party API.

  Assessment tools for this might take the form of white-boarding in a discussion, or sampling from a complete set of system interaction diagrams.

- **Produce object diagrams**

  Learners should be able to demonstrate that they have identified the classes, attributes and operations required for the product. They should highlight the relationships between the objects and how polymorphism, encapsulation and/or modularity can be used in the design of their product.

  Assessment tools for this might take the form of CRC cards or Use-Case diagrams for the major components of their implementation.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Project (SCQF level 9)

- **Write pseudocode, select algorithms and Identify design patterns**

  Learners should describe a minimum of two sub-routines of their design, one interaction of multiple objects and the incorporation of a third-party API, by writing pseudocode outlining how the algorithms the have selected would function in the operational program. Learners' pseudocode should also demonstrate how it would be incorporated into a larger product if necessary and where following design patterns could minimise their workload.

  Assessment might take the form of observed discussion of the learner's pseudocode.

Outcome 3: Develop a complex software product using an object-oriented programming language as part of a team.

Learners may develop the product scoped in Outcome 1 and designed in Outcome 2, or develop a product of equal complexity if this is not possible

- **Use version control software to manage version history**

  Learners should be able to demonstrate on-going management of source code using a version control repository. They should show a regularity of commits throughout the project life-cycle. They should be able to demonstrate that they have used the repository to share development code across the team.

  This may be evidenced by screenshots of the management interface of their chosen version control software.

- **Build working software**

  This is the core activity of this Outcome, so the primary measure of success is to produce working code. This means that the product runs from start to finish without exceptional errors. A minimum of 50% of this Outcome should be based on achieving working code. Code should be well-structured using an object-oriented programming approach, and be readable with its intention clearly documented with comments or idioms.

  The product should interact with the user, process their input, and interact with third-party APIs according to design requirements. Users should receive feedback from the product and be able to continue interaction. Results should be stored to enable persistent use of the software over time, ie users should be able to come back to find results of previous operations. Storage could take the form of file system, database or an appropriate alternative.

  Assessment might take the form of observation by assessor of the working product, with access to source code to check the structure for coverage of relevant Knowledge and Skills.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Project (SCQF level 9)

Outcome 4: Test the operation and acceptance of a complex software product as **part of a team.**

Learners may test the operation of the product they have developed for Outcome 3, or a product of equal complexity.

- **Check operation of code using a range of techniques**

- **Ensure acceptance criteria on test plan are met with adequate test coverage**

  Learners should be able to run their code from start to finish and check interaction with third-party APIs and data storage using a range of techniques. They should test code against their test plan to ensure an appropriate level of acceptance for a range of users.

- **Diagnose and correct errors**

  Learners should diagnose any identified errors and be able to rectify the error(s) to return to working code. Each learner should be responsible for ensuring the code they wrote operates successfully with the code their team members wrote.

  This type of assessment may take the form of a professional discussion by Assessor with learner demonstrating the working code and explaining steps undertaken to diagnose and correct one or more errors in their own product.

  Alternatively, learners could be asked to write a 'bug report' on program code provided to them and asked to document the steps required to duplicate a bug in their product. This could be followed by learner making appropriate corrections.

  This type of assessment might take the form of a blog post, report or a personal demonstration to the assessor of what steps the learner took to achieve results.

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: Project (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 apply skills and knowledge of software analysis, design, implementation and testing to produce a software product composed of multiple sub-programs in a 'real world' teamwork scenario. The Unit is aimed at learners looking to extend their existing qualifications to enable them to take a role as an entry-level software developer.

While following methodical project management practices and incorporating software design patterns, you will work in a team to:

- determine the scope.
- plan the development of a complex software project.
- design the operation and interaction of its components.
- produce working code to meet the requirements.
- test a completed solution to prove functional operation.

The project will include interaction with third-party APIs, and management of source code with version control software.

The Unit is suitable for learners who have completed the Units *Software Development: Analysis and Design* (SCQF level 9) and *Software Development: Implementation and Testing* (level 9) or a prior learning or experience of software development. The Unit involves the practical application of skills, and will give you exposure to the whole software development life-cycle while working with others to produce working software.

You will need to provide evidence to demonstrate your Knowledge and/or Skills across all Outcomes. Evidence of your cognitive and practical competence will take the form of at least one component of a complex software product.

On completion of the Unit you will be competent in working with others in the production of complex software products, and be able to move onto a degree-level course in software development, or into an entry-level software development job.