Higher National Unit specification

General information

Unit title: Software Development: Analysis and Design (SCQF level 8)

Unit code: HA4D 35

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 use of **object-oriented** techniques in the software analysis and design process. The Unit reviews conventional and contemporary analysis and design techniques and covers the use of object-oriented analysis techniques to define user requirements, the use of object-oriented design techniques to design software solutions, and the use of object-oriented modelling techniques to model solutions.

Learners will gain practical experience in the use of object-oriented techniques to analyse user requirements and design and model software solutions. The Unit is suitable for learners who have previous experience in the use of analysis and design techniques in software development and wish to further develop their skills in using object-oriented techniques.

On completion of the Unit, learners will be competent in using object-oriented analysis, design and modelling techniques in the software analysis and design process and should be ready to undertake the Unit *Software Development: Analysis and Design* (SCQF level 9) or equivalent.

Having completed this Unit, along with the Unit *Software Development: Implementation and Testing* (SCQF level 8), learners should be able to progress to the Unit *Software Development: Project* (SCQF level 8) or equivalent.

Higher National Unit Specification: General information (cont)

Unit title: Software Development: Analysis and Design (SCQF level 8)

Outcomes

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

1 Describe the use of analysis and design techniques in the software development process.
2 Define software requirements using object-oriented analysis techniques.
3 Design software solutions using object-oriented techniques.
4 Model software solutions using object-oriented techniques.

Credit points and level

2 Higher National Unit credits at SCQF level 8: (16 SCQF credit points at SCQF level 8)

Recommended entry to the Unit

Entry to this Unit is at the discretion of the centre. However, it would be useful if the learner had prior knowledge/skills in the analysis and design of computer software. This could be evidenced by possession of the Higher National Unit HA4C 34 *Software Development: Analysis and Design* (SCQF level 7) 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: Analysis and Design (SCQF level 8)

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 the use of analysis and design techniques in the software development process.

Knowledge and/or Skills

- Describe conventional and contemporary approaches to software development
- Describe analysis and design tools and models
- Describe the Waterfall Development approach
- Describe the Agile Development approach

Outcome 2

Define software requirements using object-oriented analysis techniques.

Knowledge and/or Skills

- Identify objects and group them into classes
- Specify object interaction
- Specify object behaviour
- Specify object internals (attributes)
- Define requirements using common models (use cases, object models)

Outcome 3

Design software solutions using object-oriented techniques.

Knowledge and/or Skills

- Produce a functional design solution using object-oriented modelling techniques
- Map technology-independent concepts onto implementing classes and interfaces to produce a model of the solution domain
- Take account of implementation constraints (hardware and software platforms, performance requirements, persistent storage and transactions, usability, budgets and time limitations)

Higher National Unit specification: Statement of standards

Unit title: Software Development: Analysis and Design (SCQF level 8)

Outcome 4

Model software solutions using object-oriented techniques.

Knowledge and/or Skills

- Model dynamic behaviours (business processes, use cases, sequence, activity, statechart diagrams)
- Model static structures (classes, class diagrams, attributes, operations, visibility, association, aggregation, inheritance, relationships between classes components)
- Use an object-oriented modelling language to model solutions

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).

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 Outcomes 1, 2, 3 and 4. The evidence of practical competence will be the use of object-oriented techniques to analyse requirements and design and model software solutions as 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 make use of object-oriented techniques to analyse software requirements and design and model solutions. The evidence would consist of the requirements analysis and the software design and model. 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 for the candidate to define the software requirements for **one** problem using object-oriented techniques. For Outcome 3, it is sufficient to design a software solution for **one** problem. For Outcome 4 it is sufficient to model a software solution for **one** problem. The same problem could be used for all three Outcomes or a different problem could be used for each Outcome.

Higher National Unit specification: Statement of standards (cont)

Unit title: Software Development: Analysis and Design (SCQF level 8)

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.

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: Analysis and Design (SCQF level 8)

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 be aware of the main characteristics of a range of conventional and contemporary Software Development approaches including: Waterfall, Prototyping, Incremental, Spiral, Rapid Application Development (RAD) and Agile. These characteristics should include the advantages and disadvantages and phases of each approach, the iterative nature of the development process and the factors affecting the choice of approach.

They should be able to describe major analysis and design tools and models, including User Interface Design, Design Principles, Data Design and Data Dictionary. They should be aware of the reasons for using object-oriented techniques (reuse of code, inheritance, etc), characteristics of good program design and know the relevant tools. In particular, they should be aware of the Agile approach to software development and the characteristics of Agile planning practices.

Learners should be able to describe the advantages and disadvantages of the Waterfall Development approach to analysis and design, and use appropriate information gathering methods. They should be able to describe the production of a functional and non-functional requirements specification.

They should also be able to describe the Agile Development approach to analysis and design, in conjunction with an object-oriented methodology, by selecting an Agile approach such as Scrum or Kanban, producing an Agile Product Roadmap, using User Stories to capture requirements, producing a Product Backlog, managing and track progress, presenting results, seeking feedback and reflecting on future improvements.

It should be made clear that Agile and object-oriented design are different. Agile software development encompasses a group of software development methods based on iterative and incremental development; whereas, object-oriented design is the process of planning a system of interacting objects for the purpose of solving a software problem. The main thing to note in these definitions is the fact that one involves planning and the other uses incremental and emergent development strategies.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Analysis and Design (SCQF level 8)

Outcome 2

Learners should know that the purpose of analysis phase of the software life-cycle is to create a model of the system's functional requirements that is independent of implementation constraints. They should realise that the main difference between object-oriented analysis and other forms of analysis is that, when using the object-oriented approach, we organise requirements around objects, which integrate both behaviours (processes) and states (data) modelled after the real world objects that the system interacts with.

In conventional analysis methodologies, processes and data are considered separately. For example, data may be modelled by entity relationship diagrams and behaviours by flow charts or structure charts.

Learners should be able to carry out the primary tasks in object-oriented analysis (OOA):

- ***finding and organising the objects:*** the best way of finding potential objects is to review each use case to find nouns that correspond to business entities or events. Class diagrams can then be used to graphically depict the identified objects and their associations and relationships.
- ***describing how the objects behave and interact*****:** each object has a state, which is the value of its attributes at a specific point in time. An object changes state when something happens or when the value of one of its attributes changes. A state diagram can be used to model the life cycle of a single object. It shows the different states that an object can have, the events that cause the object to change state, and the rules that govern the object's transition between states.
- ***defining the internals (or attributes) of the objects:*** an attribute is a named property of a class. It has a type and describes the range of values that that property may hold.

Outcome 3

Learners should be able to produce a functional design solution using object-oriented modelling techniques. They should be able to take account of implementation constraints such as hardware and software platforms, performance requirements, persistent storage and transactions, usability, budgets and time limitations.

They should be able to map technology-independent concepts onto implementing classes and interfaces to produce a model of the solution domain.

Outcome 4

Object-oriented modelling (OOM) is a common approach to modelling applications, systems, and business domains by using the object-oriented paradigm throughout the entire development life cycles. OOM is a main technique heavily used by both OOA and OOD activities in modern software engineering.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Analysis and Design (SCQF level 8)

Object-oriented modelling typically divides into two aspects of work: the modelling of dynamic behaviours like business processes and use cases, and the modelling of static structures like classes and components.

Learners should be able to model dynamic behaviours, such as business processes and use cases, and static structures such as classes and components, using object-oriented modelling languages, eg **Unified Modelling Language (UML),** a general-purpose, developmental, modelling language in the field of software engineering, intended to provide a standard way to visualize the design of a system or **Systems Modelling Language (SysML)**, a general-purpose modelling language for systems engineering which supports the specification, analysis, design, verification and validation of a broad range of systems.

Guidance on approaches to delivery of this Unit

This Unit is a component of the PDA Software Development (SCQF) level 8. It should be delivered before, or in parallel with the Unit *Software Development: Implementation and Testing* (SCQF level 8). Both of the Units should be completed before delivery of the Unit *Software Development: Project* (level 8).

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.

There is significant scope in this Unit to illustrate concepts and skills with case studies of analysis and design. 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.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Analysis and Design (SCQF level 8)

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 conventional 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 the three 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 8), 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 exercise could be used to cover all three Outcomes or a separate exercise may be used for each Outcome. The exercise(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 periodic 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 largely practical). This could be done by the blog demonstrating how analysis and design 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.

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.

Higher National Unit Support Notes (cont)

Unit title: Software Development: Analysis and Design (SCQF level 8)

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 will provide opportunities for developing Computational Thinking skills.

History of changes to Unit

Version	Description of change	Date

General information for learners

Unit title: Software Development: Analysis and Design (SCQF level 8)

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 object-oriented techniques in the software analysis and design Process. The Unit describes the analysis and design techniques used and covers the use of object-oriented analysis techniques to define user requirements, the use of object-oriented design techniques to design software solutions and the use of object-oriented modelling techniques to model solutions.

You will gain practical experience in the use of object-oriented techniques to analyse user requirements and design and model software solutions. The Unit is suitable for learners who have previous experience in the use of analysis and design techniques in software development and wish to further develop their skills in using object-oriented techniques.

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

On completion of the Unit, you will be competent in using object-oriented analysis, design and modelling and should be ready to undertake the Unit *Software Development: Analysis and Design* (SCQF level 9) or equivalent. Along with the Unit *Software Development: Implementation and Testing* (SCQF level 8), you should be able to progress to the Unit *Software Development: Project* (SCQF level 8) or equivalent.