Next Generation Higher National Unit Specification

Software Development (SCQF level 8)

Unit code: J7D9 48

SCQF level: 8 (24 SCQF credit points)

Valid from: session 2023–24

Prototype unit specification for use in pilot delivery only (version 1.0) June 2023

This unit specification provides detailed information about the unit to ensure consistent and transparent assessment year on year.

This unit specification is for teachers and lecturers and contains all the mandatory information required to deliver and assess the unit.

This edition: June 2023 (version 1.0)

Unit purpose

This is a specialist unit designed for learners with an interest in taking an object-oriented approach to software development that includes systems analysis, development, and testing.

It is a mandatory unit for the HND Software Development and is particularly suitable for learners with a vocational interest in programming, or who wish to progress to higher education and/or vendor qualification-based computing subjects.

Learners should have previous experience in computer programming at SCQF level 7 or above before starting the unit, along with good written communication, critical thinking, and analytical skills. Previous experience of a software development setting is desirable but not essential.

The unit provides a project-based learning context for applying object-oriented software development principles to the design, implementation, and testing of a software solution. Learners understand the importance of each development stage and demonstrate an appropriate use of tools to produce a software application. The unit should provide a hands-on approach to building a software application, where learners create the design and then implement that design by using an appropriate software development lifecycle methodology.

The unit introduces a range of testing strategies and highlights the importance of testing while developing software applications and their deployment. Learners understand the value of planning and time management during software development projects, and experience the challenges of working as a software developer.

Learners develop knowledge of the theoretical concepts, underlying principles, scope and role of systems analysis and design. They also develop problem solving and object-oriented technical skills through implementation and testing. Learners must demonstrate their proficiency in these skills in implementing object-oriented software solutions to a defined problem.

Learners gain competence in using programming constructs for selection, iteration, data structures and collections in an appropriate programming language. They consider programming language features such as type systems, constructs, execution, and memory management.

On completion of the unit, learners may progress to more specialised topics such as software engineering, data engineering, data science or machine learning at an appropriate level. Learners may also progress to a computing-related vendor qualification or certification exam.

Unit outcomes

Learners who complete this unit can:

1 create a software design from a given software scenario
2 code a software application from a design
3 test a software application
4 deploy a software application
5 prepare an application user guide
6 evaluate the software development processes

Evidence requirements

Learners must provide product evidence relating to the outcomes. Their knowledge and skills are inferred from the product evidence.

Learners produce product evidence in the context of a software development project. The project must involve a system based on a real-life problem. You must provide the problem scenario. The project must be an individual project, and learners must provide evidence for every part of the software development process in their evaluation report.

Each learner must provide the following evidence:

1 a user requirements specification with natural language analysis and clarifications of any assumptions or problem statements
2 a rewrite of the software specification with visualisation modelling performed using any appropriate Unified Modeling Language (UML) diagrams and technique
3 a testing plan and documentation of test logs, error logs and runs
4 source code for the project, along with an executable
5 technical documentation for the software solution
6 evidence of software deployment
7 an application user guide
8 an evaluation report of the full software development process

The learner's evaluation report must describe:

1 the extent to which the implemented solution met the proposed specification
2 a self-reflection on the conduct of the software development project (to include all the stages: design, development and testing)
3 a self-evaluation of knowledge and skills gained during the software development project
4 the use of feedback to inform the modifications made during the project

Learners can produce evidence over an extended period in lightly-controlled conditions or generate it holistically in conjunction with other units in a group award. Evidence produced in lightly-controlled conditions must be authenticated. The Guide to Assessment provides further advice on methods of authentication.

We recommend that the problem scenario that you provide is changed regularly. The standard of evidence should be consistent with the SCQF level of the unit.

Knowledge and skills

The following table shows the knowledge and skills covered by the unit outcomes:

Knowledge	Skills
Learners should understand: ♦ the software development lifecycle ♦ software development methodologies and frameworks, with emphasis on agile development ♦ continuous development ♦ software documentation ♦ software design tools ♦ user requirements analysis, including natural language analysis (NLA) ♦ software functional and non-functional requirements ♦ Unified Modeling Language for visual modelling ♦ software design patterns ♦ integrated development environments (IDEs) and programme version control ♦ how to document code ♦ programming concepts and constructs ♦ classes and objects ♦ attributes and methods ♦ parameter passing ♦ exception handling ♦ data validation ♦ standard object libraries ♦ information hiding and encapsulation ♦ abstraction ♦ polymorphism ♦ inheritance ♦ association ♦ aggregation and composition ♦ dependency ♦ multiplicity ♦ coupling and cohesion	Learners can: ♦ manage time to meet milestones for each stage of the project design, implementation, testing and deployment ♦ develop user requirements documentation ♦ create software specification documentation ♦ evaluate alternative software methodologies ♦ create documentation for selected software design ♦ use analysis and design visualisation tools ♦ use an integrated development environment (IDE) to develop code ♦ build code that is source-controlled and secure ♦ use standard libraries in code ♦ create readable code that is well-designed, modular, and maintainable ♦ use a range of data structures ♦ use input and output file operations ♦ use a range of error handling techniques ♦ use data validation techniques ♦ create a test plan using a defined strategy and methodology ♦ schedule and perform software testing ♦ create test logs ♦ create error logs ♦ create re-run tests ♦ write internal documentation ♦ write user documentation

Knowledge	**Skills**
Learners should understand: ♦ accessing and manipulating data structures ♦ arrays, linked lists, binary trees, and hash tables ♦ searching, sorting, insertion and deletion algorithms ♦ algorithms and libraries ♦ input and output file operations ♦ test documentation ♦ testing strategies and methods ♦ error handling ♦ how to debug code ♦ software security ♦ software deployment processes	Learners can: ♦ deploy a software application ♦ evaluate the final product and process

Meta-skills

Throughout this unit, learners develop meta-skills to enhance their employability in the software development sector.

Self-management

This meta-skill includes:

- focusing: demonstrating the attention to detail that developing program code and syntax demands and which is crucial to successful coding practices
- adapting: critical reflection on the processes of the project and the knowledge and skills gained; self-learning, encouraged by the project-based nature of the unit
- initiative: displaying independent thinking; demonstrating the self-motivation, responsibility and decision making required to reach project milestones with objectives and deliverables at each stage

Social intelligence

This meta-skill includes:

- communicating: receiving information through various means and interpreting it; storytelling
- collaborating: listening and conveying information
- leading: being a change catalyst; understanding the role of leadership

Innovation

This meta-skill includes:

- curiosity: information sourcing, recognising problems and devising solutions (decomposition)
- creativity: maker mentality; imagination; visualising; contributing positively to the creative design of a software solution; application of design principles and careful analysis of requirements
- sense-making: pattern recognition; holistic thinking; analysis; evaluation
- critical thinking: logical and computational thinking; deconstruction; judgement

Delivery of unit

The time required varies depending on the previous experience of individual learners. Based on 120 hours delivery and assessment time, we suggest the following distribution:

Outcome 1 — Create a software design from a given software scenario
(40 hours)

Outcome 2 — Code a software application from a design
(50 hours)

Outcome 3 — Test a software application
(10 hours)

Outcome 4 — Deploy a software application
(5 hours)

Outcome 5 — Prepare an application user guide
(5 hours)

Outcome 6 — Evaluate the software development processes
(10 hours)

Additional guidance

The guidance in this section is not mandatory.

You can carry out the unit on its own or as part of a group award. If it is part of a group award, assessment may be combined with other units in the award.

The unit provides detailed underpinning knowledge that learners need to carry out practical activities when progressing to advanced computing subjects.

You should allow for a range of scenarios and practical activities whenever possible, to allow learners to develop skills in:

- designing an object-oriented solution to a computing problem
- selecting a programming language
- implementing and testing code

You should give learners access to a range of programming languages and associated development environments. This can include a user interface library that learners can use to create a graphical interface.

Learners apply software development principles to the design and implementation of a software solution (to a significant set of requirements) and demonstrate competence in working on a software development project.

The core learning activity is a software development project that is split into four direct stages:

- design
- development
- testing
- deployment

The project should be of medium complexity. As learning becomes learner-centred, your role is to be a coach or facilitator, to provide guidance, resources and advice, and to teach the knowledge and practical skills required in a software development project.

You should set milestones for the project's duration. After each milestone, learners should meet with you to discuss the corresponding deliverables and provide an opportunity for you to ask questions and give feedback. Assessment of deliverables focuses on the consistency between the stated software design and actual program code. Learners can make changes to the design if they provide adequate justification and update documents. You should allow an acceptable level of errors in the codebase, if the process has been documented and does not affect the execution of the program.

Since the evidence created during the project represents learning, you must provide feedback that is authentic and constructive to the objectives of the problem.

Each learner must write an evaluation and reflection on their own understanding of the problem domain, any knowledge and skills gained during the project, and updated modifications. This develops their understanding of what constitutes good practice in a software development project and allows learners to move forward with these skills in future study.

To accommodate project-based learning, the learning experiences should make good use of scheduled class time and combine formal presentation of theory or technical knowledge and extended periods of practical tasks, to contribute to the project-based assessment.

You should:

1 provide a software development brief for learners
2 clarify any assumptions and problem statements from the brief; this may include a transcript of a client interview
3 run the defined test plan of the application provided by learners

We recommended that during the unit, you should:

- benchmark key dates during different stages of the project
- support learners by providing feedback at each of the different stages

Equality and inclusion

This unit is designed to be as fair and as accessible as possible with no unnecessary barriers to learning or assessment.

You should take into account the needs of individual learners when planning learning experiences, selecting assessment methods or considering alternative evidence.

Guidance on assessment arrangements for disabled learners and/or those with additional support needs is available on the assessment arrangements web page: www.sqa.org.uk/assessmentarrangements.

Information for learners

Software Development (SCQF level 8)

This section explains:

- what the unit is about
- what you should know or be able to do before you start
- what you need to do during the unit
- opportunities for further learning and employment

Unit information

This unit aims to provide you with an opportunity to apply software development principles to the design, implementation, testing and deployment of a software solution to a defined problem, and demonstrate competence in delivering a software solution.

The specialist unit is intended for learners interested in contemporary practices in software development. You should have previous experience in computer programming at SCQF level 7 or above before starting this unit. Previous experience in software development is desirable but not essential.

Most of your learning comes from working through the stages of a given software development project. This project represents a real-world situation that allows you to become actively engaged in constructing the software solution, and the process allows you to learn an industry-related skill. You understand the benefits and drawbacks to several common software development methodologies, and then manage your project through the stages of your chosen methodology for the project.

You gain knowledge of a range of tools and techniques that are used in the design, development, testing and deployment of a software development project and the production of documentation and visualisations for your design. You build source-controlled and secure code that follows a suitable design solution. You achieve this by employing a robust testing strategy and methods to get your application ready for deployment.

You understand the importance of creating and using internal and external software development documentation. You learn how to use an external repository to keep software secure and exercise version control.

The assessment for this unit focuses on the extent to which your built software application aligns with your design. At each milestone in your project there is an opportunity to receive feedback from your assessor. The evidence for assessment consists of the working software as deployed, along with the required documentation and evidence of your design, test plans and test results, and your evaluations.

Throughout the unit, you develop meta-skills covering self-management, social intelligence and innovation.

The unit is mandatory in HND Software Development and is particularly suitable if you have a vocational interest in computing, articulating to university or progressing to vendor qualification or certification exams.

Administrative information

Published: June 2023 (version 1.0)

Superclass: CB

History of changes

Version	Description of change	Date

Note: please check SQA's website to ensure you are using the most up-to-date version of this document.