SQA Advanced Unit Specification

General information for centres

Unit title: Software Development: Programming Paradigms (SCQF level 8)

Unit code: HT91 48

Superclass:	CB
Publication date:	August 2017
Source:	Scottish Qualifications Authority
Version:	01

Unit purpose

This unit is intended to introduce learners to the different programming paradigms currently in use in the software industry. It is a specialist unit, and as such builds on units that have already been undertaken. It is particularly useful to those who wish to continue their study at higher levels.

The unit examines, in turn, the main paradigms of System/Imperative programming, which may be the main point of familiarity with existing knowledge and shows that there are areas where these types of language may not be the most productive.

The next paradigm considered is that of Imperative/Low level. Although there has been a trend away from studying assembler level languages, the resurgence of Single Board Computers (SBCs) and the emergence of the 'Internet of Things' (IoT) suggests that this is a skill set that may have practical as well as academic usefulness.

Declarative/Functional languages are widely used in both academic and practical settings, and provide an alternative way of developing solutions.

An introduction to Declarative/Logic systems provides an insight into the programming languages used for example in artificial intelligence and allied systems.

While it is anticipated that most learners will have some experience in Imperative languages, this unit seeks to draw the broader picture of programming solutions in general and develop the skills and confidence to provide a solution in the most appropriate paradigm for the problem to hand.

Outcomes

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

1 Identify programming paradigms.
2 Use programming and hardware constructs in different paradigms.
3 Implement solutions using different paradigms.

Credit points and level

1 SQA Unit credit at SCQF level 8: (8 SCQF credit points at SCQF level 8)

Recommended entry to the unit

Learners should have some experience in producing programs — this could be evidenced by having achieved relevant qualifications such as the SQA Advanced Unit HP2P 47 *Software Development: Programming Foundations*.

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.

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**.

SQA Advanced Unit Specification: Statement of standards

Unit title: Software Development: Programming Paradigms (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

Identify programming paradigms.

Knowledge and/or skills

- Overview of different programming paradigms
- Imperative/System
- Imperative/Low-Level
- Declarative/Functional
- Declarative/Logic

Outcome 2

Use programming and hardware constructs in different paradigms.

Knowledge and/or skills

- Sequence
- Selection
- Iteration
- Recursion
- Branching
- Assemblers
- Compilers
- Interpreters
- Sensors
- Actuators

Outcome 3

Implement solutions using different paradigms.

Knowledge and/or skills

- Solutions in an Imperative/System language
- Solutions in an Imperative/Low-Level language
- Solutions in a Declarative/Functional language
- Solutions in a Declarative/Logic language

Evidence requirements for this unit

Learners will need to provide evidence to demonstrate their knowledge and/or skills across all outcomes.

All the outcomes will be assessed holistically by a single assignment. Learners will be required to produce solutions to a number of given problems. At least one solution must be produced for each of the different paradigms noted in Outcome 1 (Imperative/System, Imperative/Low-Level, Declarative/Functional and Declarative/Logic) and each solution must be implemented in an appropriate language/paradigm for that problem.

Solutions must be accompanied by internal documentation and results of testing.

Learners should be given a number of suitable problems to select from.

Evidence may be captured, stored and presented in a range of media (including audio and video) and formats (analogue and digital).

Assessment should be conducted under open book conditions with access to on-line resources. Assessors should take reasonable steps to assure the authenticity of learners' submissions.

SQA Advanced Unit: Support Notes

Unit title: Software Development: Programming Paradigms (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 40 hours.

Guidance on the content and context for this unit

This unit provides an opportunity for learners to experience different programming paradigms. It is anticipated that most learners will have some experience of imperative programming languages, but less experience with other paradigms.

This unit may be of most use to learners who are planning to articulate on to further courses of study.

While the choice of implementation languages is at the discretion of the centre, this document assumes the following:

Paradigm	*Implementation Language*
Imperative/System	C
Imperative/Low-Level	Assembler (ARM)
Declarative/Functional	SML
Declarative/Logic	Prolog

Guidance on approaches to delivery of this unit

This unit requires the use of software to implement solutions in each of the four different paradigms introduced. A general-purpose (desktop/laptop) computer with appropriate software would be a suitable resource, but centres may wish to consider provision of a small system such as a Single Board Computer (SBC) for the assembly language section. It is anticipated that a C compiler and an ARM assembler will be required, along with Prolog and SML interpreters.

Centres choosing to select other platforms must ensure suitability of their choice, and confirm availability of toolchains.

There are at least two different delivery modes that can be used for this unit. The first is to examine each of the paradigms in turn. This would allow a notional ten hours per paradigm.

Alternatively, the topics could be delivered in the sequence suggested by the order of the outcomes. If delivering against this sequence then the following represents a logical progression.

It may well be that some topics are more familiar to learners, and delivery time could be adjusted.

Outcome 1 (Identify programming paradigms) (8 hours)

Learners should be introduced to the fundamentals of the different programming paradigms. Depending on the amount of syntax examples use to illustrate key points of differences in paradigms some time from Outcome 3 could be transferred here.

While tutor-led delivery may be used to introduce the concepts, it may be appropriate to task learners with investigating the history of the paradigms and typical application domains.

While by no means the only choice, centres may wish to consider the use of the programming language C for the Imperative/System section of the course. C compilers are readily available.

The use of a processor with a modern instruction set, such as the ARM range is suggested as an example of the Imperative/Low-Level paradigm. Inexpensive ARM based platforms are relatively cheap and easy to source, and open source toolsets are available. (Centres may also wish to investigate the use of emulators for these devices) One starting point for resources can be found at http://infocenter.arm.com/help/index.jsp . It may also be useful to use a resource such as that at http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/ok01.html.

SML is an example of Declarative/Functional language suitable for this unit. At the time of writing, SML resources can be found at http://www.smlnj.org/.

Prolog is a suitable example language for the Declarative/Logic component. At the time of writing resources can be found at http://www.swi-prolog.org/ and http://www.learnprolognow.org/lpnpage.php?pageid=implementations.

Outcome 2 (Use programming constructs in different paradigms) (8 hours)

In terms of sequence/branch/selection iteration and recursion, it may be useful to start at the machine level (branch/jump) and develop how higher level constructs are implemented at the machine level.

Similarly, the use of functions and recursive calls can be explained in the light of machine level subroutine calls and stack frames.

A brief overview of interpreter/compilation technologies may be useful for this outcome.

The topic of Sensors is relatively large, and as such this unit should provide an introduction to some of the sensors available, and an overview of interfacing. For simple sensors such as switches, Light Dependant Resistors (LDRs) or basic Ultrasonic sensors there may be only basic issues such as pull-up resistors involved.

Similarly, the introduction to actuators should be an overview. Simple actuators such as Light Emitting Diodes (LEDs) and simple sounders along with current limiting precautions may be covered.

Outcome 3 (Implement solutions using different paradigms) (24 hours)

This outcome builds on the concepts introduced in Outcomes 1 and 2, and provides the delivery to allow learners to develop their programming skills in the various systems.

Unless significant amounts of language syntax is covered in Outcome 1, then this is the logical place to cover syntax features.

Guidance on approaches to assessment of this 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.

Assessment should be conducted under open book conditions with access to on-line resources. Assessors should take reasonable steps to assure the authenticity of learner submissions. If the authenticity of a submission is in doubt then this may be clarified by questioning such as viva voce.

Given the time-scales involved, it is not envisaged that learners are required to write solutions for large, complicated problems, but problems somewhat beyond a simple ‘hello world’ are needed. Complexity is a subjective issue within software development, so it is suggested that a system with a Cyclomatic complexity (or comparable metric) of no lower than four is selected. A Cyclomatic complexity of ten or more is probably beyond the scope of this unit.

Outcome 1 is overtaken by the successful matching of paradigm to problems, Outcome 2 is overtaken by using language constructs in the production of solutions and Outcome 3 is achieved by the production of the required solutions. All the outcomes will be assessed holistically by a single assignment. Learners will be required to produce a portfolio of solutions to a number of given problems.

While not exhaustive, ideas for problems to solve include:

- Controlling a simple sensor/actuator system, such as a Light Dependant Resistor (LDR) as a proximity sensor and a Light Emitting Diode (LED), indicating distance by frequency of flashing
- A simple password problem (match three characters)
- A Small Knowledge base, such as a small genealogy system with about 10 facts
- A small sorting routine such as Insertion sort

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.

The unit is written with the expectation that evidence be collected using an on-line approach. This does not preclude the use of more traditional methods.

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

Knowledge of a wide selection of data structures and the algorithms that operate on them is a fundamental component of the software developer's skill set and a computational thinking mindset developed while undertaking the unit will be broadly transferrable. Operating on these structures may well involve utilisation and development of Core Skills such as *Numeracy*, *Communications, ICT* and *Problem Solving*. Opportunities exist, especially in formative sections, to develop team-working skills.

History of changes to unit

Version	Description of change	Date

General information for learners

Unit title: Software Development: Programming Paradigms (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.

This unit introduces you to a number of different programming paradigms currently used in the software industry. You will gain the knowledge of different programming languages and have practical experience in using programming constructs in different paradigms and implementing solutions using different paradigms.

You will probably have some experience in writing computer programs before starting on this unit, but probably in one or more of the imperative programming languages. There are a number of different ways of constructing computer programs to solve problems. While a large number of these ways are the imperative programming languages such as the procedural languages C/C++ and Java, there are a number of approaches that do not follow these approaches. These different paradigms include the declarative programming languages such as Prolog and SML.

Declarative languages are used across the programming industry and a command of these languages can be considered an important set of skills for the professional software developer.

The assessment for this unit is the production of a small portfolio of solutions, with at least one solution being developed in each of the programming language/paradigms contained in the unit. Part of what you are being assessed on is your ability to select the appropriate language/paradigm for the solution being developed.