BCS Level 3 Certificate in Programming
QAN 603/1192/7

Version 3.3
July 2020

This is a United Kingdom government regulated qualification which is administered and approved by one or more of the following: Ofqual, Qualification in Wales, CCEA Regulation or SQA

BCS Level 3 Certificate in Programming

Contents

Change History

Any changes made to the syllabus shall be clearly documented with a change history log. This shall include the latest version number, date of the amendment and changes made. The purpose is to identify quickly what changes have been made.

Version Number	Changes Made
Version 1.0 March 2016	Syllabus Created
Version 1.1 October 2016	Amendment made
Version 1.2 December 2016	Compliance statement added
Version 2.0 August 2017	Major amendments following full review
Version 2.1 October 2017	Minor amendments following review
Version 2.2 October 2017	Amendments made
Version 2.3 October 2017	Amendments made
Version 3.0 October 2017	Final document created
Version 3.1 November 2017	Amendment to the learning outcome numbering
Version 3.2 February 2018	Minor amend to format of QAN reference
Version 3.3 April 2020	Document amended to make it suitable for a range of different learners and not just apprentices

Introduction

This Certificate is the second module of the two knowledge modules required for the Level 3 Software Development Technician apprenticeship programme. It covers the range of concepts, approaches and techniques that are applicable to software development programming, for which learners are required to demonstrate their knowledge and understanding.

This certificate can also be delivered as a standalone programme for learners working to develop their knowledge and understanding of software development programming.

Objectives

Learners should be able to demonstrate knowledge and understanding of software development programming. Key areas are:

1. Understand how to implement code, following a logical approach.
2. Understand how code integrates into the wider project.
3. Understand how to follow a set of functional and non-functional requirements.
4. Understand the end-user context for the software development activity.
5. Appreciate the importance of seamlessly connecting applications to databases and understand types of data storage and their applications.
6. Demonstrate knowledge of database normalisation.
7. Understand why there is a need to follow good coding practices.
8. Understand the principles of good interface design.
9. Understand the importance of building security in to software at the development stage.

Learners who are completing this as part of the apprenticeship programme should collate evidence of lessons learnt in these key areas and these should be reflected upon when the learner is compiling the Summative Portfolio. This will provide the learner with the opportunity to identify how the task might be done better/differently with knowledge subsequently gained.

Target Audience

The certificate is relevant to learners who are either enrolled on the Level 3 Software Development Technician apprenticeship programme or want to have an introduction to software development.

Course Format and Duration

Candidates can study for this award by attending a training course provided by a BCS accredited training provider. The estimated total qualification time for this award is 169.5 hours.

Eligibility for the Examination

Apprenticeship learners:
Individual employers will set the selection criteria, but this is likely to include 5 GCSEs (especially English, mathematics and a science or technology subject); other relevant qualifications and experience; or an aptitude test with a focus on IT skills. Learners should

have experience in writing simple code and be able to make simple connections between code and defined data sources.
Level 2 English and Maths will need to be achieved, if not already, prior to taking the endpoint assessment.

Other learners:
It is recommended that learners have completed 5 GCSEs (especially English, mathematics and a science or technology subject); other relevant qualifications and experience; or an aptitude test with a focus on IT skills. Training providers may select criteria. Learners should have experience in writing simple code and be able to make simple connections between code and defined data sources.

Format and Duration of the Examination

The format for the examination is a 60-minute multiple-choice examination consisting of 40 questions. The examination is closed book (no materials can be taken into the examination room). The pass mark is 26/40 (65%).

Additional Time for Learners Requiring Reasonable Adjustments Due to a Disability

Learners may request additional time if they require reasonable adjustments. Please refer to the reasonable adjustments policy for detailed information on how and when to apply.

Additional Time for Learners Whose Language is Not the Language of the Examination

If the examination is taken in a language that is not the learner's native / official language, then they are entitled to 25% extra time.

If the examination is taken in a language that is not the learner's native / official language, then they are entitled to use their own **paper** language dictionary (whose purpose is translation between the examination language and another national language) during the examination. Electronic versions of dictionaries will **not** be allowed into the examination room.

Guidelines for Training Providers

Each major subject heading in this syllabus is assigned an allocated time. The purpose of this is two-fold: first, to give both guidance on the relative proportion of time to be allocated to each section of an accredited course and an approximate minimum time for the teaching of each section; second, to guide the proportion of questions in the exam. Training providers may spend more time than is indicated and learners may spend more time again in reading and research. Courses do not have to follow the same order as the syllabus. Courses may be run as a single module or broken down into two or three smaller modules.

This syllabus is structured into sections relating to major subject headings and numbered with a single digit section number. Each section is allocated a minimum contact time for

presentation. Learners should be encouraged to consider their Summative Portfolio throughout the modules.

Syllabus

For each top-level area of the syllabus a percentage and K level is identified. The percentage is the exam coverage of that area, and the K level identifies the maximum level of knowledge that may be examined for that area.

1 Implementing software code following a logical approach (17.5%, K3)

Understand how to implement code, following a logical approach.

1.1 Explain the fundamental concepts of programming.

- procedural vs. object-oriented vs. functional programming;
- compiled vs. interpreted.

1.2 Demonstrate the core constructs used when writing code.

- classes;
- objects;
- methods;
- variables;
- logic operators;
    - AND
    - OR
    - NOT
    - NAND
    - NOR
    - XOR
- control structures.
    - iteration
    - selection
    - sequence

1.3 Explain and demonstrate how algorithms are used.

- encryption;
- searching;
- sorting.

1.4 Explain and demonstrate how data structures are used and how data is represented in software code.

- types of data;
    - integer
    - floating
    - Boolean
    - character
    - string
- variables;
- lists, stacks, arrays.

1.5 Describe how to write software code in order to solve problems.

- describe how programs are structured;
  - instructions
  - sub-routines
  - pseudocode
  - data definitions and links
  - comments
- describe modularity and the rational re-use of code.
  - design patterns
  - library functions
  - frameworks

1.6 Understand the fundamental concept of Test Driven Development (TDD).

2 How code integrates into the wider project (10%, K2)

Understand how code integrates into the wider project.

2.1 Describe the activities undertaken in the following stages of software development:

- design;
- code development;
- testing.

2.2 Outline the activities undertaken in the following stages of software development:

- feasibility study;
- requirements analysis;
- deployment / implementation.

2.3 Understand software development activities for the following roles:

- requirements engineer;
- business analyst;
- software designer;
- software developer;
- software tester;
- software release engineer.

2.4 Describe the key business concepts and artefacts that must be considered during a software development project.

- processes and procedures;
  - business process management as it relates to business involvement in development
  - release management
- documentation;
- training;
- support;
- service levels.

2.5 Describe how software development is conducted within governance structures and the role of the project manager.

2.6 Understand how effective team-working contributes to the effective delivery of software projects.
- decision making;
- conflict resolution;
- collaboration;
- communication;
- peer review and retrospectives.

3 Developing software against a set of functional and non-functional requirements (15%, K2)

Understand how to follow a set of functional and non-functional requirements.

3.1 Understand how to follow a set of functional and non-functional requirements.

3.2 Understand the difference between functional and non-functional requirements and how these are used to drive software development activities.
- how to review requirements;
- how to assess their validity;
- how they are used as input to software design;
- how they are used during testing to ensure adequate test coverage.

3.3 Identify the different types of non-functional requirements, and the reasons they are important to the end-product of software development.
- availability;
- capacity;
- performance;
- scalability;
- reliability;
- maintainability;

3.4 Recognise common ways in which software requirements can be expressed.
- requirements documents – clear, unambiguous;
- user stories;
- use case diagrams;
- process models / flow diagrams;
- UML diagrams.

3.5 Describe the qualities of good requirements and the impact of poor requirements.

3.6 Explain how to determine the correct level of test coverage based on each requirement / type of requirement.

4 The end user context for software development (7.5%, K2)

Understand the end-user context for the software development activity.

4.1 Understand and recognise the relationship between the user and the environment in which the software will be used.

4.2 Understand the individual business and external constraints and dependencies that need to be taken into account when developing software.
- compliance;
- ethics;
- governance;
- legality.

4.3 Describe the methods used to identify end-user needs.
- questionnaires;
- user interviews;
- contextual enquiry;
- focus groups;
- personas;
- customer journey mapping.

5 Connecting code to data sources (5%, K2)

Appreciate the importance of seamlessly connecting applications to databases and understand types of data storage and their applications.

5.1 Explain the purpose of data storage for storing new information (orders or customer information).
- orders;
- customer information.

5.2 Explain the purpose of data storage for extracting and displaying data.
- products;
- pricing.

5.3 Explain the concept and key features of databases and data stores.
- relational databases;
- SQL and NoSQL;
- data files;
- data structures (tables, records, fields, definitions);
- document;
- key-value.

6 Database normalisation (5%, K3)

Demonstrate knowledge of database normalisation.

6.1 Explain the purpose and importance of effective data modelling and normalisation.

6.2 Demonstrate the principle of normalisation, that information or data should be stored only once.

7 Following good coding practices (12.5%, K2)

Understand why there is a need to follow good coding practices.

7.1 Explain the importance of good coding practice.

- quality of coding development.
    - design documentation
    - structure of code
    - consistent design and structure
    - secure code

7.2 Explain the purpose of good software coding principles and practices.

- the basic common principles;
    - DRY (don't repeat yourself)
- defensive programming;
- commenting;
- refactoring;
- patterns / anti-patterns.

7.3 Understand that there are a range of open and organisational coding standards and where to source them.

8 Principles of good interface design (10%, K2)

Understand the principles of good interface design.

8.1 Explain human computer interaction and understand the issues associated with interactive systems.

- usability / ease of use and intuitive design;
- graphical user interfaces (GUI) for different types of devices;
- ergonomic design.

8.2 Describe the key concepts and processes of good user interface design.

- design principles;
- design patterns;
- tools;
    - wireframes
    - prototypes
- techniques and methods.
    - A/B testing

8.3 Explain the importance of usability when developing interactive systems.

8.4 Describe the fundamental considerations for developing an accessible system and the purpose of the Web Accessibility Initiative (WAI).

9 Building in security software (17.5%, K2)

Understand the importance of building security in to software at the development stage.

9.1 Describe the following types of security issues and the scale and nature of threats that can impact software development.
- common security attacks;
- security versus resilience;
- social engineering.

9.2 Explain what is meant by 'building security in', in terms of secure software development and creating a secure end-product, and why it is important.
- the role coders play in determining a secure software end-product;
- the impact they can have on security by not building security in;
- why building security in at the start is better than trying to retrofit later.

9.3 Describe proactive security approaches during software design and development.
- security development lifecycle (SDLC);
- defensive design / defensive programming;
- test creation and execution;
- permission setting and role-based access;
- physical infrastructure and security.

9.4 Explain approaches to make software more secure.
- security scanning;
- penetration testing;
- fuzzing;
- load testing.

Levels of Knowledge / SFIA Levels

This syllabus will provide learners with the levels of difficulty / knowledge skill highlighted within the following table, enabling them to develop the skills to operate at the levels of responsibility indicated. The levels of knowledge and SFIA levels are explained on the website www.bcs.org/levels. The levels of knowledge above will enable learners to develop the following levels of skill to be able to operate at the following levels of responsibility (as defined within the SFIA framework) within their workplace:

Level	Levels of Knowledge	Levels of Skill and Responsibility (SFIA)
K7		Set strategy, inspire and mobilise
K6	Evaluate	Initiate and influence
K5	Synthesise	Ensure and advise
K4	Analyse	Enable
K3	Apply	Apply
K2	Understand	Assist
K1	Remember	Follow

Question Weighting

Syllabus Area	Target number of questions
1. Implementing software code following a logical approach	7
2. How code integrates into the wider project	7
3. Developing software against a set of functional and non-functional requirements	6
4. The end-user context for software development	3
5. Connecting code to data sources	3
6. Database normalisation	2
7. Following good coding practices	3
8. Principles of good interface design	4
9. Building in security software	5
Total	**40 Questions**

Format of Examination

Type	40 Question Multiple Choice.
Duration	60 minutes. An additional 25% will be allowed for learners sitting the examination in a language that is not their native / mother tongue.
Pre-requisites	Training from a BCS Accredited Training Provider is strongly recommended but is not a pre-requisite.
Supervised	Yes
Open Book	No
Pass Mark	26/40 (65%).
Calculators	Calculators cannot be used during this examination.
Total Qualification Time (TQT)	169.5 Hours
Delivery	Online.

Trainer Criteria

Criteria	▪ Have 10 days training experience or have a train the trainer qualification ▪ Have a minimum of 3 years practical experience in the subject area

Classroom Size

Trainer to learner ratio	1:16