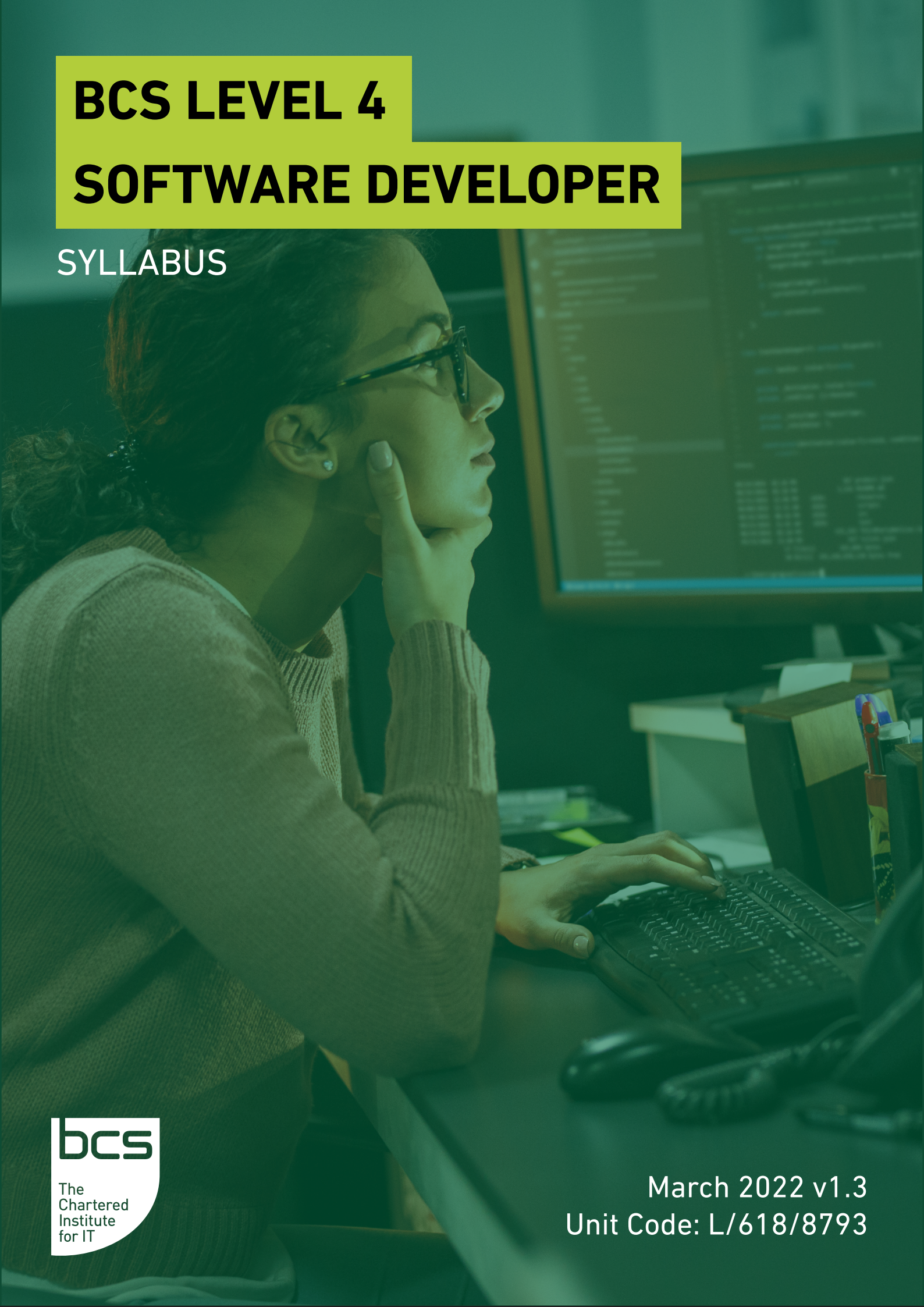
BCS LEVEL 4 SOFTWARE DEVELOPER

SYLLABUS



bcs
The Chartered Institute for IT

March 2022 v1.3
Unit Code: L/618/8793

CONTENTS

Introduction

Software touches every organisation, in every sector either directly or indirectly through bespoke development, the use of off-the-shelf business solutions or the realisation of an idea. Software developers turn requirements into reality, through a cycle of analysis, planning, building and testing using a range of tools and concepts.

This Level 4 module covers the key concepts, skills and tools required of anyone working in a Software Developer role, to be able to successfully undertake the tasks required for the development of quality software solutions.

Find out more about the BCS Level 4 Digital Modular Programme qualification in the Qualification Guide.

Qualification Suitability and Overview

This occupational module should be taken as part of the BCS Level 4 Diploma - Digital Modular Programme in Software Development, and cannot be taken as a standalone qualification. Learners must have successfully completed the exam for the BCS Level 4 Module in Digital Core within the last 12 months in order to undertake this module.

They will also need a good standard of written English and Maths. Centres must ensure that learners have the potential and opportunity to gain the qualification successfully. Learners must be aged 16+ to take this module.

This qualification is suitable for learners who are looking to progress their career within a software developer role. It can be taken as a standalone qualification, or in combination with other units and modules as part of a wider programme, such as an apprenticeship.

This is an occupationally focused qualification which will:

- Test a learner's applied knowledge, skills and behaviours to a range of scenarios
- Demonstrate a practical understanding of key concepts across the topic areas
- Enable a learner to progress in their career

Learners can study this module by attending a training course provided by a BCS accredited Training Provider or through self-study.

Total Qualification Time	Guided Learning Hours	Independent Learning	Assessment Time
340 hours	**237 hours**	**102 hours**	**1.5 hours**

Trainer Criteria

It is recommended that to effectively deliver this certification, trainers should possess:

- 10 days training experience or have a train the trainer qualification.
- A minimum of 3 years practical experience in the subject area.

SFIA Levels

This module provides learners with the level of knowledge highlighted within the table, enabling them to develop the skills to operate successfully at the levels of responsibility indicated.

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**

SFIA Plus

This syllabus has been linked to the SFIA knowledge skills and behaviours required at level 4 for an individual working in Programming and Software Development.

KSB15:

Taking innovative approaches to problem solving and/or devising inventive and creative solutions.

KSB19:

Understanding the needs, objectives and constraints of those in other disciplines and functions.

KSB24:

Working collaboratively with others to achieve a common goal.

KSC19:

Applying standards, practices, codes, and assessment and certification programmes relevant to the IT industry and the specific organisation or business domain.

KSC22:

Methods and techniques for structured reviews, including reviews of technical work products, test plans, business cases, architectures and any other key deliverables.

KSC23:

Testing techniques used to plan and execute software tests of all application components (functional and non-functional) to verify that the software satisfies specified requirements and to detect errors. Examples, but not limited to: dynamic testing techniques and test automation techniques.

KSC84:

Understanding and application of different development approaches e.g. iterative/ incremental methodologies (Agile, XP, TDD, SCRUM) or traditional sequential methodologies (Waterfall or V-Model) and their energy and resource footprints. Irrespective of development methodology a DevOps approach may also be taken where development and operational staff work collaboratively.

KSC01:

Applying automated systems to support specific business functions or processes and operational support systems.

KSC02:

Organised and documented sets of techniques, and proven methods, intended to facilitate the structured and open development of applications.

Further detail regarding the SFIA Levels can be found at www.bcs.org/levels.

Learning Outcomes

Upon completion of the module Learners will be able to demonstrate a practical understanding of:

- How to use the software development lifecycle and the roles of individuals within it.
- How to use the project lifecycle and the roles of individuals within it.
- How to select and apply a suitable software development methodology.
- How to use a combination of software development approaches and concepts to develop software in line with organisational and policies and procedures.
- How to build and use algorithms and databases.
- How and when to apply different types of testing.

Syllabus

1. The Software Development Lifecycle (SDLC) (10%) (K2)

Learners will be able to:

1.1 Implement the Software Development Lifecycle in a business context.

Indicative content

a. Software Development Lifecycle.
b. Systems Development Lifecycle.
c. Facilitate The Creation of Software.

Guidance

Learners shall be able to apply the Software Development Lifecycle to solve a range of complex business problems and opportunities. The Software Development Lifecycle (also known as the systems development lifecycle) provides structure to the management of software, from initial feasibility and requirements identification to ongoing maintenance.

1.2 Apply the seven stages of the Software Development Lifecycle to a business situation.

Indicative content

a. Feasibility Study.
b. Requirements Analysis.
c. Design.
d. Code Development.
e. Testing.
f. Deployment/Implementation
g. Maintenance.
h. Supporting legacy code.

Guidance

Learners shall be able to breakdown and apply every stage of the SDLC. Learners shall appreciate that this lifecycle, although appearing Linear/Waterfall in nature, can also be applied to Agile environments, where stages may be revisited/looped back to in a more iterative manner.

1.3 Implement the main activities expected of a software developer role at each stage of the Software Development Lifecycle.

Indicative content

a. Role of the software developer.
b. Software Developer Kit (SDK).
c. Solution Design (UML).

Guidance

Learners shall be able to demonstrate their ability to undertake the key tasks required of an individual in a software developer role throughout each stage of the SDLC interpreting and using UML, development using the SDK (IDE, Compiler, Debugger, Git).

1.4 Produce the high-level deliverables from each stage of the Software Development Lifecycle.

Indicative content

a. Return on investment.
b. Outcomes.
c. Costs and benefits.
d. Minimum viable product.

Guidance

Learners shall create the high level deliverables required from each stage of the SDLC, in order to solve a business problem. Such deliverables from each stage of the SDLC, may include for example the feasibility report which outlines the project plan and costings created during the feasibility study, or the full documented analysis of the problem and system created during requirements analysis.

2. The roles and responsibilities within the Software Development Lifecycle (SDLC) (10%) (K3)

Learners will be able to:

2.1 Analyse the roles and duties of others and relate them to the software development lifecycle.

Indicative content	Guidance
a. Requirements engineer. b. Business analyst. c. Software designer. d. Programmer/coder. e. Software tester. f. Software release engineer. g. Technical architect. h. Domain expert. i. Independent tester. j. Product owner.	Learners shall clearly define the various roles and responsibilities required throughout the SDLC, identifying relationships and dependences between roles. Learners will be able to identify and explain the duties commonly associated with these roles, including identifying the most suitable role to undertake a particular task or responsibility. and the relationships/dependancies between roles. Note that the individual job titles may vary in workplaces, and some titles used in an Agile development environment may encapsulate other roles.

2.2 Relate software development roles to the expected involvement in each stage of the SDLC .

Indicative content	Guidance
a. Roles from 2.1.	Learners shall understand the need for each of the roles outlined above and be able to describe their expected contribution and output throughout each stage of the SDLC, in a given business context. From the list provided in 2.1, learners should be able to identify the roles which are required at each stage of SDLC, what their expected contribution is and their reasoning for investment at that stage.

2.3 Compare and contrast the skills required to fulfil each role within the SDLC .

Indicative content	Guidance
a. Leadership. b. Creativity. c. Communication. d. Analytical skills. e. Team-work.	Learners shall clearly explain the differences and similarities between the skill sets required of each of the roles outlined above, and how they contribute to successful development.

3. The project lifecycle (5%) (K2)

Learners will be able to:

3.1 Employ the phases of the project lifecycle.

Indicative content	Guidance
a. Initiation Phase. b. Planning Phase. c. Execution Phase. d. Termination Phase.	Learners shall be able to practically implement each stage of the project lifecycle in a business context. Learners shall describe the purpose and scope of each stage of the lifecycle and detail the activities undertaken at each stage.

3.2 Explain the characteristics of the project lifecycle.

Indicative content	Guidance
a. Cost. b. Staffing. c. Risk and uncertainty. d. Ability to accommodate change.	Learners should be aware of how the costs, risks and demands of a project can vary throughout the PLC and factors which could influence these.

3.3 Compare and contrast the duties associated with each of the roles in the project lifecycle.

Indicative content	Guidance
a. Project Manager. b. Project Team Member. c. Project Sponsor. d. Executive Sponsor. e. Business Analyst.	Learners shall be able to identify the duties commonly associated with these roles at each stage of the project lifecycle, explaining the differences and similarities between roles, while identifying the most suitable role to undertake a particular task or responsibility.

3.4 Explain how the principles of the project life cycle management were applied in a familiar software development project.

Indicative content	Guidance
a. Project life cycle stages and principles.	Learners should be able to relate the project lifecycle to a given situation or task, detailing the roles and activities at each stage in that project.

4. Software development methodologies (10% K4)

4.1 Implement the primary characteristics of software development methodologies.

Indicative content

a. Waterfall Development.
b. Agile Development.
c. DevOps Deployment.
d. Rapid Application Development.
e. Behaviour driven development.
f. Test driven development.

Guidance

Learners should be able to understand and apply the key characteristics of the given methodologies. Learners shall be able to explain the common uses, strengths and weaknesses of each approach.

4.2 Compare and contrast the respective strengths and weaknesses of each of the software development methodologies listed in 4.1.

Indicative content

a. Methodologies from 4.1.

Guidance

Referring to the methodologies listed above, learners should be able to discuss the strengths, weaknesses, differences and similarities in these methodologies and how they may be used to compliment or support one another.

4.3 Describe the circumstances under which the use of a particular software development methodology would be appropriate.

Indicative content

a. Methodologies from 4.1.

Guidance

Referring to the methodologies listed above, learners should be able to explain and justify why a particular methodology would be selected, and the expected outcome and output of using this approach.

5. Software design approaches and solutions (10% K4)

5.1 Explain the importance of the following software design concepts in a business context.

Indicative content

a. Abstraction.
b. Control Hierarchy.
c. Data Structure.
d. Information Hiding.
e. Modularity.
f. Software Architecture.
g. Structural Partitioning.
h. Object oriented, Functional, Procedural.

Guidance

Learners shall recognise that the understanding and application of these concepts are fundamental to the process of software design. Learners should be able to explain the purpose and practical application of each concept.

5.2 Assess the importance of the following software characteristics to a given software product.

Indicative content

a. Compatibility.
b. Extensibility.
c. Fault Tolerance.
d. Maintainability.
e. Modularity.
f. Performance.
g. Portability.
h. Reliability.
i. Reusability.
j. Robustness.
k. Scalability.
l. Usability.

Guidance

By comparing the characteristics of the concepts listed, learners should be able to assess the suitability of a given concept in line with business and solution priorities and value. i.e. what is of the greatest importance and value, versus the greatest risks.

5.3 Choose the most appropriate software design pattern and framework.

Indicative content

a. Adapter.
b. Decorator.
c. Iterator.
d. Observer.
e. Singleton.

Guidance

Learners should be able to draw comparisons between design patterns and frameworks in order to select the most suitable tools for a particular solution. Learners shall be able to identify these patterns in use, including providing examples of when their use would be required/appropriate.

6. Organisational policies and procedures relating to the tasks being under taken (10% K2)

6.1 Describe the relationship between policies and procedures and explain how different procedures can implement the same policy.

Indicative content	Guidance
a. Difference between policy and procedure.	Learners should be able to understand how policy drives procedures. For example, a policy may state that any code developed should be "clean and maintainable", and therefore a procedure should exist to detail the acceptable standard of code, any specific organisational requirements and the steps required to build it. Multiple procedures may exist to ensure compliance with a single policy.

6.2 Apply well-defined policies and procedures to ensure the effectiveness of an organisation's operations.

Indicative content	Guidance
a. Governance. b. Legal requirements. c. Culture. d. Customer satisfaction. e. Working standards.	The need for policy and procedure in an organisation may be driven by a legal need, an industry standard or an organisational preference or style. Each policy or procedure will impact the way a task or role is undertaken and the expected outcome. Learners should be able to explain how to apply each of these and recognise examples of them in practise.

6.3 Discuss the range of policies and procedures that might be implemented in a software development environment.

Indicative content	Guidance
a. Naming conventions. b. Commenting of code. c. Source control (committing, pulling, pushing, merging etc). d. Secure development. e. Data protection.	Learners should be able to provide examples of the types of policies and procedures that are likely to exist in a software development environment and their purpose. This includes, but is not limited to, the examples listed. Learners should also understand the applied principles of Security By Design.

7. The principles of algorithms, logic and data structures relevant to software development (15% K4)

7.1 Analyse the role and purpose of different types of algorithms to meet a business need.

Indicative content	Guidance
a. Problem solving. b. Automation. c. Speed of processing.	Learners shall understand that an algorithm is a set of instructions provided in order to complete a given task. Learners shall be able to select an appropriate algorithm to assist with processing large volumes of data, solve complex problems and/or automate tasks.

7.2 Prepare examples of the use of Sequence, Selection, Iteration and Recursion in an algorithm.

Indicative content	Guidance
a. Sequence. b. Selection. c. Iteration. d. Recursion.	As the building blocks for algorithms, learners should be able to describe and create examples of these constructs.

7.3 Analyse the use of abstract data types in the design and analysis of algorithms.

Indicative content	Guidance
a. Queue. b. Stack. c. List.	Learners shall be able to identify and justify the use of abstract data types, showing understanding of how this data can be manipulated using different operations - for example, finding or adding to existing lists.

7.4 Calculate the space and time complexity of an algorithm.

Indicative content	Guidance
a. Big O and Little O notation.	Learners should be able to calculate the space and time complexity (memory required to complete, time taken to run) of a given algorithm using data provided.

7.5 Analyse the purpose and use of single and multidimensional arrays in programming.

Indicative content

a. Access individual elements.
b. Subscripts.
c. Searchable - find specific data.

Guidance

Learners shall be able to recognise the need for single and or multidimensional arrays and their suitability for use in a given solution or business context.

7.6 Discuss the advantages and disadvantages of using a list in place of an array and explain the way in which a list may be implemented as a linked structure.

Indicative content

a. Linked lists.
b. Making changes and additions.
c. Storage requirements.
d. Searching versus subscripts.

Guidance

Learners should be able to compare the differences in uses between a list and an array including how and when one may be more suitable than the other, considering their advantages and disadvantages.

7.7 Analyse the implementation of a stack and a queue using linked lists and/or arrays.

Indicative content

a. Stack - push, pop, empty.
b. Queue - add, remove, empty.

Guidance

Learners should understand the role of stacks and queues and how to successfully implement them using a range of lists/linked lists, arrays and pointers.

7.8 Analyse the implementation of a tree structure and discuss its use in software development.

Indicative content

a. Abstract data type.
b. Single element - roots, branches, leaves.
c. Binary trees or N-ARY.

Guidance

Learners should be able analyse the creation and role of a tree structure and explain its purpose of storing and finding data, including describing the individual roles of the roots, branches and leaves.

7.9 Show how a graph structure can be used to represent directed and undirected graphs and describe the basic operations provided by a graph structure.

Indicative content	Guidance
a. Permits loops. b. Nodes and relationships.	Learners should be able to develop graphs and highlight the differences between directed and undirected graphs. Learners should understand the role of the nodes and relationships in the graph structure and how they are represented.

7.10 Analyse the operation and implementation of common sorting algorithms.

Indicative content	Guidance
a. Bubble Sort. b. Insertion Sort. c. Quicksort.	With a number of sorting algorithms available, learners should be able to explain how each works and its suitability in a given business context, considering factors such as time, complexity and storage requirements.

7.11 Analyse the operation and implementation of a number of common searching algorithms.

Indicative content	Guidance
a. Linear Search. b. Binary Search. c. Tree Search. d. List size - volume of data.	Different types of search algorithms exist, and learners should be able to explain how each works and it's suitability and effectiveness in a given business context.

7.12 Compare and contrast the use of Hash Tables with a range of search algorithms.

Indicative content	Guidance
a. Speed. b. Collisions.	Learners should be able to consider the role of hash tables as a storage space and be able to assess the suitability of each type of algorithm to perform a search. Factors to be considered may include unique or duplicate values, collisions and time.

8. The principles and uses of relational and non-relational databases (10% K4)

8.1 Analyse the use of database software for storing data.

Indicative content	Guidance
a. Store and retrieve. b. Run queries.	Learners shall explain the role of a database as a storage location for data, to which queries may be applied to find, link or manipulate data. Learners should assess the suitability of a database in a business context.

8.2 Discuss the characteristics of a relational database management system, its role in a business context and the nature of Structured Query Language (SQL).

Indicative content	Guidance
a. Select/where from statements. b. Based on tables. c. Relational algebra.	Learners should be able to describe the key characteristics of a relational database and how it works, as listed. Learners will be expected to understand the role of SQL in managing data in a relational database.

8.3 Compare and contrast the use of relational databases with the use of Not Only SQL (NOSQL) systems.

Indicative content	Guidance
a. NOSQL - graphs, documents, key-value pairs. b. SQL - relational database. Scalability, ease of use, performance	Learners should understand and describe the differences and similarities in strengths, weaknesses and suitability of SQL relational databases and NOSQL systems in a business context, considering the factors listed.

8.4 Compare and contrast differing implementations of NOSQL databases.

Indicative content	Guidance
a. Document based. b. Key value. c. Graph based.	Learners should draw comparisons between the suitability, use, strengths and weaknesses of NOSQL databases, from the list provided. Learners should be able to explain the differences and similarities in the structure and purpose of these database types.

9. The nature of software designs and functional/technical specifications (10% K3)

9.1 Produce flowcharts and pseudocode to represent a software design.

Indicative content

a. UML Flowchart - flowline, terminal, process, decision, input/output, pre-defined process.
b. Pseudocode - sequence, selection and iteration.

Guidance

Learners should be able to build, complete and error check flowcharts and pseudocode to represent a given design or process. Learners should expect to be tested on items such as completing a missing element or highlighting errors in existing items.

9.2 Produce a functional specification for a given requirements document.

Indicative content

a. Functions that a component must perform.
b. Descriptions of system requirements, input/output.

Guidance

Learners should be able to build, complete and error check a functional specification, to represent the functional requirements elicited from business stakeholders. Learners can expect to be tested on completing partly built specifications and highlighting errors in existing items.

9.3 Produce a technical specification for a given requirements document.

Indicative content

a. Specific requirements of a project.
b. Language, hardware, software, libraries.

Guidance

Learners should be able to build, complete and error check a technical specification to represent the technical requirements elicited from business stakeholders. Learners can expect to be tested on completing partly built specifications and highlighting errors in existing items.

10. The nature of software testing frameworks and methodologies. (10% K4)

10.1 Compare and contrast functional testing methods.

Indicative content

a. Unit Testing including white box, black box, grey box and dry runs.
b. Integration Testing.
c. System Testing.
d. Acceptance Testing.

Guidance

Learners should apply their knowledge of various functional testing methods and compare their similarities, differences and suitability in a given context. Learners must understand how each form of testing is completed and its role in the development process.

10.2 Compare and contrast non-functional testing methods.

Indicative content

a. Performance Testing.
b. Security Testing.
c. Usability Testing.
d. Compatibility Testing.

Guidance

Learners should apply their knowledge of various non-functional testing methods and compare their similarities, differences and suitability in a given context. Learners must understand how each form of testing is completed and its' role in the development process.

10.3 Compare and contrast commonly used software testing frameworks.

Indicative content

a. Linear Automation Framework.
b. Modular Based Testing Framework.
c. Library Architecture Framework.
d. Data-driven Framework.
e. Keyword-driven Framework.
f. Hybrid Testing Framework.

Guidance

Learners shall apply their understanding of common testing frameworks to explain the differences, similarities and suitability for a given context.

Examination Format

This module is assessed through completion of an invigilated online assessment which learners will only be able to access at the date and time they are registered to attend.

Type	Digital assessment which includes: • a knowledge-based assessment consisting of 20 questions • a set of scenario-driven situational judgement assessments consisting of 4 scenarios each with a set of 5 questions, 20 in total
Duration	90 minutes
Supervised	Yes
Open Book	No (no materials can be taken into the examination room)
Passmark	Pass 26/40 (65%), Distinction 34/40 (85%)
Delivery	Digital format only

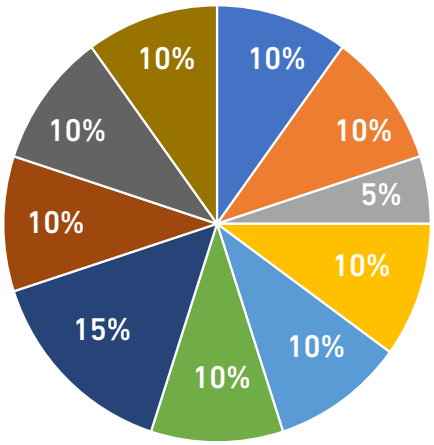
Adjustments and/or additional time can be requested in line with the BCS reasonable adjustments policy for Learners with a disability, or other special considerations including English as a second language.

Question Weighting

Each major subject heading in this syllabus is assigned a percentage weighting. The purpose of this is:

1. Guidance on the proportion of content allocated to each topic area of an accredited course.
2. Guidance on the proportion of questions in the exam.

Syllabus Area	Question type	
1. The Software Development Lifecycle (SDLC)	Multiple choice/ Scenario based/Written answer	10%
2. The roles and responsibilities within the Software Development Lifecycle (SDLC)		10%
3. The project lifecycle		5%
4. Software Development methodologies		10%
5. Software design approaches and solutions		10%
6. Organisational policies and procedures relating to the tasks being undertaken		10%
7. The principles of algorithms, logic and data structures relevant to software development		15%
8. The principles and uses of relational and non-relational databases		10%
9. The nature of software designs and functional/ technical specifications		10%
10. The nature of software testing frameworks and methodologies		10%
Total		**100%**



Recommended Reading

The following titles are suggested reading for anyone undertaking this award. Learners should be encouraged to explore other available sources.

Title:	Software Development in Practice
Author:	Bernie Fishpool, Mark Fishpool
Publiser:	BCS
Publisher Date:	August 2020
ISBN:	1780174977

Using BCS Books

Accredited Training Organisations may include excerpts from BCS books in the course materials. If you wish to use excerpts from the books you will need a license from BCS. To request a license, please contact the Head of Publishing at BCS outlining the material you wish to copy and the use to which it will be put.

Document 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	Document created.
Version 1.1	Syllabus updated.
Version 1.2	Updates to topic headings and passmark.

CONTACT

For further information please contact:

BCS
The Chartered Institute for IT
3 Newbridge Square
Swindon
SN1 1BY

T +44 (0)1793 417 445

www.bcs.org

The Chartered Institute for IT