This qualification is regulated by one or more of the following:
Ofqual, Qualifications Wales, CCEA Regulation or SQA.

BCS LEVEL 5 DIPLOMA IN IT
SOFTWARE ENGINEERING

SYLLABUS
THIS QUALIFICATION WILL BE RETIRING IN 2026

CONTENTS

3. Introduction
4. Qualification Suitability and Overview
5. SFIA Levels
6. Learning Outcomes
7. Syllabus
18. Examination Format
18. Question Weighting
19. Recommended Reading
21. Document Change History

September 2023 v3.1
This qualification is regulated by one or more of the following: Ofqual, Qualifications Wales, CCEA Regulation or SQA.
Introduction

Level 5 Diploma in IT

The second stage within the BCS three-stage Higher Education Qualification programme, the Level 5 Diploma enables candidates who have already achieved the Level 4 Certificate in IT to progress to higher levels of knowledge and competency.

This internationally-recognised qualification introduces you to the business-related aspects of the IT industry, developing your technological expertise while also considering the potential challenges of the day-to-day running of an organisation, such as legal obligations and intellectual property.

Our modules have been created in-line with the latest developments in the industry, giving you a competitive edge in the IT job market. You will have the opportunity to learn about object-oriented programming, user experience, systems analysis and design, as well as to build upon knowledge and skills developed during the Level 4 Certificate.

To successfully achieve the qualification, candidates need to complete:

- One core module
- Three optional modules
- One Professional Project in IT

Candidates who wish to progress onto the next stage will need to complete the Project at end of the Level 6 Professional Graduate Diploma in IT.

Software Engineering 1 Optional Module

The Software Engineering 1 module is an optional module that forms part of the Level 5 Diploma in IT – the second stage within the BCS three-stage Higher Education Qualification programme. Candidates will be introduced to software engineering and its theoretical models, software design principles and will learn about the software development process, including project planning and product risk management.

Qualification Suitability and Overview

Candidates must have achieved the Certificate in IT or have an appropriate exemption to be entered for the Diploma in IT. Candidates can study for this diploma by attending a training course provided by a BCS accredited Training Provider or through self-study, although it is strongly recommended that all candidates register with an approved centre. Studying with an approved centre will deliver significant benefits.

Candidates are required to become a member of BCS, The Chartered Institute for IT, to sit and be awarded the qualifications. Candidates may apply for a four-year student membership that will support them throughout their studies.

The Level 5 Diploma is suitable for professionals wishing to gain a formal IT qualification, and this module may be particularly relevant for candidates interested in career opportunities such as intelligent systems, forensic computing, or computer security.

<table>
<thead>
<tr>
<th>Total Qualification Time</th>
<th>Guided Learning Hours</th>
<th>Assessment Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1886 hours</td>
<td>225 hours</td>
<td>2 hours</td>
</tr>
</tbody>
</table>
SFIA Levels

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

<table>
<thead>
<tr>
<th>Level</th>
<th>Levels of Knowledge</th>
<th>Levels of Skill and Responsibility (SFIA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>K7</td>
<td>Evaluate</td>
<td>Set strategy, inspire and mobilise</td>
</tr>
<tr>
<td>K6</td>
<td>Synthesise</td>
<td>Initiate and influence</td>
</tr>
<tr>
<td>K4</td>
<td>Analyse</td>
<td>Ensure and advise</td>
</tr>
<tr>
<td>K3</td>
<td>Apply</td>
<td>Enable</td>
</tr>
<tr>
<td>K2</td>
<td>Understand</td>
<td>Apply</td>
</tr>
<tr>
<td>K1</td>
<td>Remember</td>
<td>Assist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Follow</td>
</tr>
</tbody>
</table>

SFIA Plus

This syllabus has been linked to the SFIA knowledge skills and behaviours required at Level 5.

ASUP3

Identifies and resolves issues with applications, following agreed procedures. Uses application management software and tools to collect agreed performance statistics. Carries out agreed applications maintenance tasks.

DESN4

Designs components using appropriate modelling techniques following agreed architectures, design standards, patterns and methodology. Identifies and evaluates alternative design options and trade-offs. Creates multiple design views to address the concerns of the different stakeholders of the architecture and to handle both functional and non-functional requirements. Models, simulates or prototypes the behaviour of proposed systems components to enable approval by stakeholders. Produces detailed design specification to form the basis for construction of systems. Reviews, verifies and improves own designs against specifications.

TEST3

Reviews requirements and specifications, and defines test conditions. Designs test cases and test scripts under own direction, mapping back to pre-determined criteria, recording and reporting outcomes. Analyses and reports test activities and results. Identifies and reports issues and risks associated with own work.

HCEV3

Applies tools and methods to design and develop users’ digital and off-line tasks, interactions and interfaces to meet agreed usability and accessibility requirements for selected system, product or service components. Creates workable prototypes. Assists, as part of a team, on overall user experience design. Assists in the evaluation of design options and trade-offs. Consistently applies visual design and branding guidelines.

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

Learning Outcomes

Upon completion of this module, candidates will be able to:

- Explain the background of the software crisis and the need for an engineering approach.
- Appreciate the distinction between software programming and an engineering approach to the development of a software product.
- Create models of software data and processes using object oriented modelling approaches such as the UML.
- Describe and evaluate software tools and technology to enhance productivity and quality of software development.
- Demonstrate skills of software documentation, quality assurance and evaluation, and testing as part of software development.
Syllabus

1. The Nature of Software

Learners will be able to:

1.1 Discuss the nature of software.

Indicative content

a. Defining software
b. Software application domains
   i. Systems software
   ii. Application software
   iii. Engineering/scientific software
   iv. Embedded software
   v. Artificial intelligence software
   c. Legacy software
d. Changing nature of software
   i. Web apps
   ii. Mobile apps
   iii. Cloud computing
   iv. Product line software

Guidance
Candidates should be able to understand that producing software is an engineering task and that engineering disciplines are often applicable to it. They should also appreciate that producing software is more than just programming.

1.2 Discuss theoretical models.

Indicative content

a. Prescriptive models:
   i. Waterfall models, e.g. classic lifecycle or V model
   ii. Incremental process models
   iii. Evolutionary process models, e.g. prototyping or spiral model
   iv. Concurrent models

Guidance
Candidates should understand how models have brought useful structure to the discipline of software engineering, as well as how modern software is characterised by constant change, tight schedules and the need to meet users’ expectations. They should also understand and be able to discuss both strengths and weaknesses of evolutionary process models, which were originally conceived to address these issues.

1.3 Explain the motivation for development of software engineering.

Indicative content

a. Projects running over budget
b. Projects running over time
c. Inefficient software
d. Software not meeting requirements
e. Unmanageable projects
f. Code difficult to maintain
g. Failure to deliver

Guidance
Candidates need to be able to understand the motivations for the development of software engineering as a discipline, as well as to describe issues that have led to its development.

1.4 Describe the cost of maintenance.

Indicative content

a. Characterising maintenance, e.g. ISO/IEC 14764
b. Corrective maintenance
c. Adaptive maintenance
d. Perfective maintenance
e. Preventative maintenance

Guidance
Candidates need to be able to understand a variety of reasons for which maintenance is necessary, and appreciate that the maintenance costs may often be larger than initial development costs.
2. Software Engineering key practices

**Learners will be able to:**

2.1 Describe and analyse the multidisciplinary nature of software design.

**Indicative content**
- Characteristics of a software engineer
- Psychology of software engineering
- Cohesiveness of the software team
- Team structures
- Agile and global teams
- Collaboration tools

**Guidance**
Candidates are expected to be able to identify distinct sub-disciplines of software engineering and to be able to identify their characteristics.

2.2 Explain team work in software engineering.

**Indicative content**
- Characteristics of a software engineer
- Psychology of software engineering
- Cohesiveness of the software team
- Team structures
- Agile and global teams
- Collaboration tools

**Guidance**
It is important for candidates to appreciate the overall concept that software is never developed in isolation. Paired programming would be one example. Candidates should also understand how having a sense of purpose, involvement, trust and improvement will all contribute to a software team’s overall effectiveness.

2.3 Describe productivity in software engineering.

**Indicative content**
- Lines of code
- Function points
- Constructive cost model (CoCoMo)
- Cyclomatic complexity

**Guidance**
In general, candidates should be able to measure the effectiveness of software engineering processes for this section, possibly in lines of code, but other measures may be applicable.

1.5 Explain software quality.

**Indicative content**
- Software functional quality
- Software structural quality
- The cost of quality
- Quality of design
- Quality of conformance
- Performance quality
- Feature quality
- Reliability
- Durability
- Serviceability
- Aesthetics

**Guidance**
Candidates need to have an appreciation of the wide variety of factors which contribute to a measure of software quality. They should also be able to appreciate that given factors will be differently emphasised in a variety of contexts.

Explain team work in software engineering.

**Indicative content**
- Characteristics of a software engineer
- Psychology of software engineering
- Cohesiveness of the software team
- Team structures
- Agile and global teams
- Collaboration tools

**Guidance**
It is important for candidates to appreciate the overall concept that software is never developed in isolation. Paired programming would be one example. Candidates should also understand how having a sense of purpose, involvement, trust and improvement will all contribute to a software team’s overall effectiveness.

Describe productivity in software engineering.

**Indicative content**
- Lines of code
- Function points
- Constructive cost model (CoCoMo)
- Cyclomatic complexity

**Guidance**
In general, candidates should be able to measure the effectiveness of software engineering processes for this section, possibly in lines of code, but other measures may be applicable.
Describe testing in software engineering.

**Indicative content**

a. Black-box testing  
b. White-box testing  
c. Grey-box testing  
d. Unit testing  
e. Integration testing  
f. System testing  
g. Acceptance testing  
h. Smoke and sanity testing  
i. Regression testing  
j. Functional and non-functional testing  
k. Usability testing  
l. Security testing  
m. Traceability testing

**Guidance**

Candidates are expected to be able to describe a range of valid range of testing techniques, rather than one single technique. Candidates should also understand that one testing approach is not necessarily an alternative to another, but can be a complementary approach which may uncover a different class of errors to other methods.

Explain product maintenance.

**Indicative content**

a. Maintenance debt  
b. Dependence on external factors  
c. Lifecycle

**Guidance**

Candidates need to be aware that delivery of a product is not the end of the cycle – it will be followed by maintenance. The idea of maintenance debt, which lies in more closely the cost of maintenance with decisions taken earlier in the maintenance lifecycle, is also important.

Describe the software product life cycle.

**Indicative content**

a. Preliminary analysis  
b. Systems analysis, requirements definition  
c. Systems design  
d. Development  
e. Integration and testing  
f. Acceptance, installation, deployment  
g. Maintenance  
h. Evaluation  
i. Disposal

**Guidance**

Candidates need to be able to explain and describe the phases that a typical software development exercise will go through.

3. Software development models and methods

**Learners will be able to:**

3.1 Explain design principles.

**Indicative content**

a. Transparency  
b. Separation of concerns  
c. Abstraction  
d. Modularity  
e. Development by incremental methods

**Guidance**

Candidates should understand the role of software design and its place within the modelling activity, as it sets the stage for the construction phase – its goal is to create a software model which will take into account (and meet) all user requirements.

3.2 Utilise and demonstrate notations for software components.

**Indicative content**

a. Syntax  
b. Semantics

**Guidance**

Candidates need to be familiar with techniques for modelling software systems. They should be able to identify the essential features of modelling notations and distinguish between the allowable symbols within notations and the meaning of those symbols.

3.3 Demonstrate Unified Modelling Language (UML) modelling.

**Indicative content**

a. UML modelling of use cases for a logical/end-user view (e.g. use case diagram)  
b. Class diagram  
c. Object diagram  
d. Activity diagram  
e. Sequence diagram

**Guidance**

Candidates must be able to discuss and demonstrate the use of UML techniques in order to visualise the design of a system.
4. Validation, verification and testing

Learners will be able to:

4.1 Describe product and process visibility.

**Indicative content**
- Product visibility
- Process visibility

**Guidance**
Candidates will need to be able to explain the concept of visibility and outline ways in which visibility can be achieved for both software products and software processes.

4.2 Explain traceability in software systems and describe the processes.

**Indicative content**
- Mapping from requirements to specifications
- Mapping from specification to design
- Mapping from design to implementation
- Testing to ensure traceability
- Derivation paths
- Flowdown paths

**Guidance**
Candidates should be able to identify the steps a software engineer will take to ensure that user requirements identified in a specification are realised in an implementation.

5. Software engineering tools and environments

Learners will be able to:

5.1 Demonstrate and explain Computer Aided Software Engineering (CASE) tools.

**Indicative content**
- Automated support
- Semi-automated support
- Upper CASE
- Lower CASE

**Guidance**
Candidates should be able to discuss the ways in which automated tools can aid the software engineer.

5.2 Describe the role of repositories.

**Indicative content**
- Package management systems
- Source control software
- Software dependencies

**Guidance**
Candidates should understand the role of repositories in supporting incremental development.

5.3 Explain software reuse and evolution.

**Indicative content**
- The reuse landscape
- Design patterns
- Generator-based reuse
- Application frameworks
- Application system reuse

**Guidance**
Candidates should be able to identify the way in which software reuse can contribute to the development of new applications. Software should be designed to be reusable in the first instance, as well as making sure to reuse it when the opportunity arises.
6. Project management

**Learners will be able to:**

6.1 Explain how to use project estimating and project planning tools.

**Indicative content**

- Algorithmic cost modelling
- Expert judgement
- Estimation by analogy
- Parkinson's Law
- Pricing to win

**Guidance**

Candidates need to be able to discuss a range of techniques they might need to employ in order to estimate the amount of effort necessary to produce a software component.

6.2 Describe the management and maintenance of software products.

**Indicative content**

- Variety of mechanisms for updating software deployed in customer environments
- Risks associated with software updates

**Guidance**

Candidates should be able to describe a range of strategies for updating software products which have been deployed and are operating in user environments. They should be able to discuss the different levels of risks of said strategies and the context in which they should be used.

6.3 Explain the total cost of system ownership.

**Indicative content**

- Computer hardware and programs
  - Network hardware and software
  - Server hardware and software
  - Workstation hardware and software
- Installation and integration of hardware and software
- Purchasing research
- Warranties and licenses
- License tracking compliance
- Migration expenses
- Risk
- Operation expenses
  - Infrastructure
  - Power
  - Testing costs
  - Downtime, outage and failure expenses
  - Diminished performance
  - Security
  - Backup and recovery
  - User training
  - Audit
  - Insurance
  - Information technology personnel costs
  - Management costs
- Long-term expenses
  - Replacement
  - Upgrade
  - Decommissioning

**Guidance**

Candidates need to be able to identify all of the issues contributing to the cost of a software product and be able to use these to evaluate different software development strategies.
### Examination Format

This module is assessed through completion of an invigilated written exam.

- **Type**: Four written questions from a choice of six, each with equal marks
- **Duration**: Two hours
- **Supervised**: Yes
- **Open Book**: No (no materials can be taken into the examination room)
- **Passmark**: 10/25 (40%)
- **Delivery**: Paper format only

Adjustments and/or additional time can be requested in line with the [BCS reasonable adjustments policy](#) for candidates with a disability or other special considerations.

### Question Weighting

Candidates will choose four questions from a choice of six. All questions are equally weighted and worth 25 marks.

---

### 6.4 Analyse and explain the software life cycle cost modelling.

#### Indicative content

- Prevention costs
- Appraisal costs
- Failure costs
  - Internal failure costs
  - External failure costs

#### Guidance

Candidates should understand and have an appreciation for the cost of quality, as well as the cost of a lack of quality, both for users who must try to contend with glitches in software and for the team that has built and now has to maintain it.

### 6.5 Describe project and product risk management.

#### Indicative content

- Schedule flaws
- Requirements inflation
- Employee turnover
- Specification breakdown
- Poor productivity

#### Guidance

Candidates should be able to list the main risk factors in producing software products and be able to describe techniques to mitigate them.
Recommended Reading

Primary texts

<table>
<thead>
<tr>
<th>Title</th>
<th>Author</th>
<th>Publisher</th>
<th>Date</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Engineering</td>
<td>I. Somerville</td>
<td>Pearson</td>
<td>2015</td>
<td>978-1292096131</td>
</tr>
</tbody>
</table>

Additional texts and resources

<table>
<thead>
<tr>
<th>Title</th>
<th>Author</th>
<th>Publisher</th>
<th>Date</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Mythical Man-Month</td>
<td>F. P. Brooks</td>
<td>Addison-Wesley</td>
<td>1995</td>
<td>978-0201835953</td>
</tr>
<tr>
<td>Clean Architecture</td>
<td>R. C. Martin</td>
<td>Prentice Hall</td>
<td>2017</td>
<td>978-0134494164</td>
</tr>
<tr>
<td>Beginning Software Engineering</td>
<td>R. Stephens</td>
<td>Sybex</td>
<td>2015</td>
<td>978-8126555376</td>
</tr>
<tr>
<td>Effective Project Management: Traditional, Agile, Extreme</td>
<td>R. K. Wysocki</td>
<td>Wiley India</td>
<td>2014</td>
<td>978-8126552207</td>
</tr>
<tr>
<td>Peopleware: Productive Projects and Teams</td>
<td>T. Demarco and T. Lister</td>
<td>Addison Wesley</td>
<td>2016</td>
<td>978-0321934113</td>
</tr>
<tr>
<td>Managing Software Debt: Building for Inevitable Change</td>
<td>C. Sterling</td>
<td>Addison Wesley</td>
<td>2020</td>
<td>978-0321948618</td>
</tr>
</tbody>
</table>
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 its intended use.

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.

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1.0</td>
<td>Document created</td>
</tr>
<tr>
<td>August 2021</td>
<td></td>
</tr>
</tbody>
</table>