UNIVERSITY OF LONDON

BSc Computer Science

# Module Specification

| Key Information | | | |
|---|---|---|---|
| Module title | Software Design and Development | | |
| Level | 5 | Credit value | 15 |
| Member Institution | Goldsmiths | Notional study hours and duration of course | 150 |
| Module lead author/ Subject matter expert | Matthew Yee-King | | |
| Module co-author | | | |

## Rationale for the module

Professional software engineers use many techniques and tools to manage the development and improve the quality of the software they produce. Version control systems are used throughout the software industry to handle and store software as it is developed, including the integration of contributions from teams of people. Test-driven development and unit testing is a standard method used to test and verify software as you produce it. Software engineers use exception handling and defensive coding techniques to make more robust, fault-tolerant software. By learning how to use these tools and techniques, you will be ready to work on larger software projects with other people, which is what you can expect to do in the workplace as well as in other modules in the programme.

## Aims of the module

This module aims to advance your software development skills so that you can write more robust and complicated programs. You will learn how to use a range of programming techniques that will allow you to deal with unwanted or unexpected events that might happen when your application is running. You will use defensive coding to check data before processing it, and exception handling to gracefully manage unforeseen or unwanted occurrences. You will learn how to discuss program structure concerning cohesion (how to meaningfully organise code into modules) and coupling (how to define the interactions between different parts of the program). You will learn about test-driven development, where you write tests for your code, and write the code itself, in parallel. You will also learn how to use software versioning tools to manage a software project as it develops.

## Topics covered in this module:

The topics listed here are an approximation of what will be covered. The topics presented may be slightly revised to ensure currency and relevance. Students will be advised of any changes in advance of their study.

1. Language primer 1: variables and conditionals
2. Language primer 2: control flow
3. Language primer 3: functions
4. Version control
5. Collaboration using version control
6. Module coupling and cohesion
7. Unit testing
8. Test driven development
9. Defensive coding
10. Exception handling

Approximately 10-12 hours of study will be required per topic. The remaining study time is intended for coursework and examination preparation.

## Learning outcomes for the module

Students who successfully complete this module will be able to:

1. Write programs using variables, control flow and functions
2. Use defensive coding and exception handling techniques to prevent processing of invalid data and to handle unexpected events
3. Use version control tools to manage a codebase individually and collaboratively
4. Define test driven development and write unit tests
5. Assign different categories of module coupling and cohesion to a given program
6. Describe how user testing can be carried out and evaluated

## Assessment strategy, assessment methods

**Summative and Formative Assessments**
The module will contain a range of summative and formative assessments. Summative assessments are assessments which contribute directly towards your final grade. Formative assessments do not count directly towards your final grade. Instead, they provide you with opportunities for low stakes practice, and will often provide some sort of feedback about your progress. For example, a practice quiz might provide you with feedback about why a particular answer was wrong.

**Assessment Activities**
The table below lists the assessment activity types you might encounter taking the module. It also states if that type of assessment can be automatically graded. For example, multiple choice quizzes can be automatically graded, and so can some programming assignments. It also states if that type of assessment will be found in the summative coursework and the summative examination. More details about the summative assessments are provided below.

| Assessment activity type | Can it be automatically graded with feedback in some cases? | Coursework | Examination |
|---|---|---|---|
| Quiz | X | X | X |
| Writing task | | X | X |
| Programming task | X | X | X |
| Peer review task | | x | |

**Pass Mark**

In order to pass this module, you must achieve at least 35% in each element of summative assessment and an overall weighted average of 40%, subject to the application of rules for compensation. Please refer to the programme regulations for more information.

**Summative Assessment Elements**

As this is a module that has a significant amount of theory it is assessed as a theory-based module. This means that the summative assessment is composed of two elements, whose weightings are listed in the table below.

| Summative Assessment Component | Percentage of final credit | Deadline |
|---|---|---|
| **Coursework** | 50% | Mid session |
| **Examination** | 50% | End of session |

The coursework comprises a variety of practical exercises and quizzes which in total will take up to 25 hours of study time to complete. The examination will be two hours long, and consist of written answer and multiple choice questions.

## Learning resources

The module will draw on a number of different, largely web-based, public resources as well as the resources produced as bespoke material for this module. The standard text book(s) for the module will be:

> Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14, Scott Meyers, O'Reilly Media, ISBN-10: 1491903996, 2014

The programming language used is C++.