Software Design and Development Syllabus

Version 20201008

Module description

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.

The course is organised into five high level topics, each of which contains four sub-topics. The sub-topics are detailed in the table towards the end of this document. The list of the high level topics is as follows:

Topic 1: Modules and module complexity

Topic 2: Test-driven development

Topic 3: Robust and secure programming

Topic 4: User testing

Topic 5: Version control

Module goals and objectives

Upon successful completion of this module, you 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

Textbook and Readings

There are a variety of readings in the course, many taken from the IEEE library which you can access with your University online library account. There is no core textbook.

Module outline

The module consists of 5 high level topics, each of which spans four weeks.

Topic 1: Modules and module complexity	**Key concepts:** • Modules and module complexity • Module cohesion: theory and analysis • Module coupling: theory and analysis • Module coupling and cohesion in practice **Learning outcomes:** • Assign different categories of module coupling and cohesion to a given program • Write programs using variables, control flow and functions
Topic 2: Test-driven development	**Key concepts:** • Test-driven development • Unit testing in Python • Unit testing in C++ • Unit testing a web API **Learning outcomes:** • Define test driven development and write unit tests • Write programs using variables, control flow and functions
Topic 3: Robust and secure programming	**Key concepts:** • Assertion and parameter checking • Secure programming • Exception handling

	• Using a debugger **Learning outcomes:** • Use defensive coding and exception handling techniques to prevent processing of invalid data and to handle unexpected events • Write programs using variables, control flow and functions
Topic 4: User testing	**Key concepts:** • User stories: what should they be able to do? • Black box and white box testing • Usability testing • Accessiblity testing **Learning outcomes:** • Describe how user testing can be carried out and evaluated
Topic 5: Version control	**Key concepts:** • Clone/ create, add, commit • Branch, merge, conflicts, gitignore (what not to commit) • Best practice bug reporting for public/ private repositories • Optimising your public github profile **Learning outcomes:** • Use version control tools to manage a codebase individually and collaboratively

Activities of this module

The module is comprised of the following elements:

- Lecture videos. In each topic, you will find a sequence of videos in which the example programs for the course are coded up. Further videos review the key programming techniques seen in the coding videos.
- Readings. Each topic may include several suggested readings. These are a core part of your learning and, together with the videos, will cover all of the concepts you need for this module.
- Practice Quizzes. Each topic will include practice quizzes, intended for you to assess your understanding of the topics. You will be allowed unlimited attempts at each practice quiz. There is no time limit on how long you take to complete each attempt at the quiz. These quizzes do not contribute toward your final mark in the class.
- Programming Activities. Some topics include programming activity worksheets.
- Code. Some topics include source code for you to work with.
- Discussion Prompts. Each topic includes discussion prompts. You will see the discussion prompt alongside other items in the lesson. Each prompt provides a space for you to respond. After responding, you can see and comment on your peers' responses. All prompts and responses are also accessible from the general discussion forum and the module discussion forum.
- Assessed coursework. There is one assessed coursework, in the middle of the module.
- Exam. There is an exam at the end of the module.

How to pass this module

The module has two major assessments each worth 50% of your grade:

- Midterm coursework. This consists of writing and programming tasks
- End of term exam. This consists of multiple choice questions and long answer questions.

Activity	Required?	Deadline week	Estimated time per course	% of final grade
Midterm coursework	Yes	1-10	25 hours	50%
End of term examination	Yes	22	25 hours	50%