School of Electrical and Information Engineering
University of the Witwatersrand, Johannesburg
ELEN3009A — Software Development II

Course Brief and Outline — 2023

Academic Staff

Dr S.P. Levitt (course co-ordinator)
Room: CM3.237
Tel: 011 717 7209
Email: stephen.levitt@wits.ac.za

1 Course Background

This course forms part of the "Software Development" line of courses. It follows on from Software Development I and is a pre-requisite for Software Development III. It deepens and broadens students' technical knowledge and skills with respect to software engineering and development.

2 Course Objectives

This course has three main objectives. It introduces the student to object-oriented modelling and design, programming using modern C++, and key software engineering technical practices such as unit testing and version control. More specialized software engineering courses flow from, and build on, the conceptual knowledge and practical skills developed in this course.

3 Course Outcomes

On successful completion of this course, the student is able to:

1. describe how C++ fits into the software development ecosystem and the trade-offs that it makes with respect to other programming languages;
2. perform object-oriented analyses of moderately complex software problems;
3. design and implement object-oriented solutions in C++ for such problems;
4. program in C++ in a modern, idiomatic fashion;
5. create a graphical user interface (GUI) using a game programming library;
6. utilise a test framework for software testing and verification;
7. use version control in a simple manner for tracking changes and sharing code;
8. use a formal notation to document a software design as well as an automated documentation tool.

4 Course Content

C++'s Place in the World
Compiled versus interpreted languages; strongly-typed versus weakly-typed languages; language popularity; programming paradigms

The Development Environment
The Integrated Development Environment (IDE); the build toolchain: pre-processor, compiler, linker; debugging

C++ Fundamentals
Primitive types, pointers and references; parameter passing; `const`; `static`, smart pointers, `auto`, range-based for loops, tuples

The C++ Standard Template Library
Containers; iterators; algorithms; vector and string

Programming Principles and Practices
Readable code, DRY principle, defensive coding, separation of concerns

Object-Oriented Analysis and Design
Levels of abstraction; interface and implementation; object-oriented analysis and design process; Tell, Don't Ask principle; Liskov substitution principle; code smells; modelling pitfalls

Object-Oriented Programming in C++
Classes and objects; constructors and destructors; copy construction and assignment; inheritance and polymorphism; aggregation.

Unit Testing
Automated unit testing, unit-test frameworks, test-driven development; writing good tests

Version Control
Git and GitHub, commits, fast-forward merges, pull requests, collaboration

Software Documentation
Unified Modelling Language (UML); class and sequence diagrams; technical documentation; extracting documentation from code

Error Handling
Exceptions; assertions; pre- and post-conditions; error handling schemes

Graphical User Interfaces
Game development library; graphics; event handling

5 Prior Knowledge Assumed

This course depends on content covered in Software Development I. In particular it assumes that students have a working knowledge of programming fundamentals (in C++). This includes variables; scope; flow control; raw pointers; functions, standalone classes, and the ability to write small programs to solve problems.

The prerequisites and co-requisites to register for this course are defined in the current *Rules & Syllabuses: Faculty of Engineering and the Built Environment*.

6 Assessment

6.1 Formative Assessment

Although marks are allocated to the assessment of laboratories (see Section 6.2), this is regarded as formative in that the emphasis is on student participation and learning rather than whether the solutions are correct or not. Feedback on laboratories is provided through comments on each group's source code submissions as well as discussions held during dedicated lab review sessions.

6.2 Summative Assessment

Assessment Contributor	Duration (hours)	Component	Method & Weight	Calculator Type	Permitted Supporting Material
Laboratories and Engagement	33	No	Marks, 8%	–	–
Project	30	No	Rubric, 30%	–	–
Test	1	No	Marks, 22%	1	Reference sheets provided
Examination*	3	No	Marks, 40%	1	Reference sheets provided

*Note that the end-of-year examination requires a minimum of 35% in order to pass the course (termed a subminimum). This means that even if the average final mark for the course is 50% or above, a final examination mark of less than 35% does not qualify one to pass the course, as per the University Rules and Syllabuses.

6.3 Assessment Methods

Students will be assessed working *alone* (test and exam) and in *groups* (labs and project).

The project will require the student to work as part of a team to creatively identify, assess, formulate and solve a software development problem by using concepts, methods, tools and techniques introduced in this course. Additionally, the student must communicate the results critically and effectively by means of a set of group-effort reports using appropriate structure, style and graphical support. A marking rubric will accompany the project brief and this rubric will present the criteria for assessing the project outcomes.

The test is a written test which will take place during the semester. The material to be covered will be indicated by the lecturer.

The final exam will be time-tabled within the Oct/Nov examination period. All the material covered during the course (including lectures, laboratories, additional material, and the course project) is examinable.

The test and examination are the only *mandatory in-person* requirements for this course. Students will be required to write the test and all examinations (final exam/deferred/supplementary) on the Wits campus at the venue, and scheduled date and time, that is published by the School or the Examination and Graduations Office.

7 Satisfactory Performance (SP) Requirements

For the purpose of Rule G.13 *satisfactory performance in the work of the class* means the completion of prescribed laboratory activities, submission of assignments, and writing of scheduled tests unless excused in terms of due procedure.

8 Teaching and Learning Process

8.1 Teaching and Learning Approach

The basic material of this course is covered in lectures. From time to time, additional material will be referred to. In many lectures questions will be posed and students should answer these in the discussion forums. The forums are also available for the general discussion of course material.

Lecture will be used to introduce and explain key concepts, but these will need to be reinforced through practical laboratory work.

The software development tools used during this course are freely available enabling students to use their own computing facilities.

8.2 Information to Support the Course

Supporting information for this course can be found on the course website.

There is no prescribed textbook for this course, although the textbook that is used in Software Development I covers some of the course material. Many good introductory texts on C++ programming are available in the Engineering and Geo/Maths libraries and at major bookstores. Additionally, there is an enormous amount of material that is accessible on the web. These resources can be used to supplement the course material.

Each of the following books are highly recommended if you wish to delve deeper into the language. *C++ Crash Course* is a concise and modern introduction to C++. *Effective Modern C++* is a very good "morality" guide — focusing on what you should do rather than presenting everything that you can do with C++. *C++ Coding Standards* and the *Core Guidelines* present important guidelines and considerations for C++ developers. Finally, *The C++ Programming Language* is by the creator of C++, Bjarne Stroustrup, and is the best reference for the language.

1. Josh Lospinoso. *C++ Crash Course: A Fast-Paced introduction*. No Starch Press, San Francisco, first edition, 2019
2. Scott Meyers. *Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14*. O'Reilly Media, first edition, 2014
3. Herb Sutter and Andrei Alexandrescu. *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices*. C++ In-Depth Series. Addison-Wesley Professional, Berkeley, California, first edition, 2005; along with the C++ Core Guidelines
4. Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley Professional, fourth edition, 2013

8.3 Learning Activities and Arrangements

Lectures

These will generally be made available in the form of videos which can be viewed from the course website. Students are expected to watch the recordings and it will be taken that all material covered in this fashion has been viewed by students. Lecture slots will be used to work through exercises in order to reinforce the course material.

Laboratories

There are a number of laboratories associated with this course. *Doing the laboratories is indispensable for understanding the course material.* Teaching assistants will be on hand to provide assistance in the forums and during the laboratory sessions.

Project

The project brief will handed out to the students during the course. The project will be carried out over a number of weeks. Students are required to work in groups of two for the project. Note that the School's policy on the timely submission of projects, as described in the *Red Book*, will be strictly enforced.

Consultation

Students are expected to consult with the lecturer or teaching assistants during the laboratory sessions. Students should contact the lecturer via email with regard to personal matters relating to the course.

9 Course Home Page

Further information and announcements regarding the course are posted on Ulwazi and/or the course home page: https://witseie.github.io/software-dev-2/

All students are expected to consult the course home page, and their email, at regular intervals.