Course Outline

Contents

- Course Details
- Course Summary
- Assumed Knowledge
- Learning Outcomes
- Teaching Strategies and Rationale
- Student Conduct
- Assessment
- Course Schedule
- Resources for Students
- Course Evaluation and Development

Course Details

Course Code	COMP1531
Course Title	Software Engineering Fundamentals
Convenor	Aarthi Natarajan (/users/z2136899)
Admin	Aarthi Natarajan (/users/z2136899)
Classes	**Lectures** : Tue 14:00 - 16:00 Science Theatre (K-F13-G09) Wed 15:00 - 16:00 Science Theatre (K-F13-G09) Timetable for all classes (/COMP1531/17s2/timetable)
Consultations	Aarthi Natarajan (to be advised)
Units of Credit	6
Workload	In class: (Lectures: 3 hours/week, Tute/lab: 3 hours/week) Outside class: (Labs, Group Project, Quizzes, ...)
Course Website	http://cse.unsw.edu.au/~cs1531/ (http://cse.unsw.edu.au/~cs1531/)
Handbook Entry	http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP1531.html (http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP1531.html)

Course Summary

This course provides an introduction to software engineering principles: software life-cycle concepts, modern development methodologies including XP, Scrum etc., conceptual modeling and how these activities relate to programming. The students are exposed to agile software practices, team collaboration and effective communication through implementing a group project based on agile software methodologies that requires them to analyse, design, build and deploy a web-based application. This course is typically taken in the semester after completing COMP 1511, but could be delayed and taken later. It provides essential background for the teamwork and project management required in many later courses.

The goal of this course is to expose the students to:

- basic elements of software engineering - derived from the life-cycle of a software system, including requirements elicitation, analysis and specification; design; construction; verification and validation; deployment; and operation and maintenance
- software engineering methodologies, processes, measurements, tools and techniques
- agile software, collaboration and communication practices
- web-based system architecture and development practices on web platforms

Assumed Knowledge

We assume that students have taken a first programming course, which has included:

- ability to write simple programs in a programming language e.g. C and are familiar with loop structures, defining and invoking functions and returning results
- exposure to a moderate-sized, team-based project and some testing/debugging ideas

These are assumed to have been acquired in COMP 1511.

Learning Outcomes

After completing this course, students will be able to ...

1. describe the phases of software development and life-cycle of software - and illustrate them from experience
2. describe common behavior that contribute to the effective functioning of a team and identify necessary roles in a software development team
3. understand conceptual data modelling and develop simple data models
4. get some exposure to project management and software development tools
5. analyse a problem and elicit user requirements
6. articulate software design principles and use a design paradigm to design a software system (e.g., simple Web application based on the MVC framework)
7. demonstrate robust coding practices (e.g., handling exceptions, following coding standards, refactoring)
8. describe effective coding validation and verification techniques (e.g., code reviewing, fault logging, a range of test measures)
9. demonstrate effective usage of testing fundamentals (e.g., unit tests, integration tests, test plan/cases, test automation)4
10. understand agile software development practices and analysis, design, implementation and testing in the context of an agile framework

Teaching Strategies and Rationale

This course uses the standard set of practice-focused teaching strategies employed by most CSE foundational courses:

- Lectures ... introduce concepts, show examples
- Tutorials ... reinforce concepts and provide additional examples
- Lab Work ... provide examples of using various technologies
- Assignments .. will be in the form of a group project

This course aims to provide the students with a strong foundation in the fundamental principles and practices of software engineering that will prepare them for the advanced software engineering workshops. As such, a broad range of key software engineering topics will be taught and reinforced through a group project, that will enable students to apply the theoretical concepts acquired to solve a practical software engineering problem. An agile software delivery style has been chosen for the implementation of the group project, to make students familiar with modern agile development methodologies.

Lectures

Lectures will be used to present the theory and practice of the techniques in this course. Although the lectures will primarily focus on the key concepts of software engineering, some lectures will also include practical demonstrations of various key technologies required for the implementation of the group project. Lecture notes will be available on the course web pages before each lecture.

Tutorials

Tutorials will commence from week 2, where you will be expected to attend a one-hour tutorial session. These sessions will serve two purposes: (1) clarify ideas from lectures and work through exercises based on the lecture material (2) provide practical demonstrations in the tools and languages that students would need to learn to implement their group project. You should make sure that you use them effectively by examining in advance the material to be covered in each week's tutorial, by asking questions, by offering suggestions and by generally participating. The tutorial questions will be posted on the Web in the week before each tutorial. Attendance at the tutorial is compulsory and will constitute 2% of the overall course mark.

Laboratory Classes

Following the tutorial class each week, there will be a two-hour laboratory class. The nature of the activities carried out in the lab sessions will vary each week.

Some sessions will comprise a variety of small practical exercises involving the tools and frameworks introduced during the lectures. These exercises will be released in the week preceding the lab class. These lab sessions will be done in pairs, where you and partner will be "pair programming". Pair programming is an activity in agile software development activity, where one person is coding and the other person reviews if the software delivers the required goals. Each practical programming session will be scheduled in blocks of two weeks, to enable you and your partner to swap the roles of the "coder" and "reviewer". Hence, you will be required to pair with the same partner for the two-week block. You and you partner should discuss the exercises before going to the lab, to maximise the usefulness of the class and decide on the role you will be taking. The pair members can be changed for the next programming block.

To obtain a mark for a lab exercise you should (as a pair) demonstrate the completed lab exercise to your tutor during a lab session and your tutor will provide feedback on your approach to the problem and on the style of your solution. You should **normally** get your lab work assessed during the week for which it is scheduled (i.e. you must complete the week 2 lab exercise during the week 2 lab). Your tutor may permit you to demonstrate your completed exercise if you and your partner experience genuine difficulty in complete the practical exercise.

Summary: to obtain any lab marks for the Week X lab, you must do 2 things:

- *demonstrate your work to your tutor in the week X lab class*
- OR *demonstrate your work at the start of the lab in week X+1 (only if deemed necessary by the tutor)*

You **cannot** obtain marks by e-mailing lab work to tutors.

Periodically, the lab sessions will be used to schedule demonstrations of your ongoing group project. As the implementation of the group project will be based on an **agile software development methodology** , you will required to demonstrate (as a team) progress of your project in iterations.

As this course is has a significant practical component, laboratory classes are important to help you acquire the necessary skills in the relevant tools and frameworks that will be used in the implementation of the group project and also prepare you for the final exam. If you do not put a good amount of effort into the lab classes you risk failing the group project and the final exam.

Lab exercises will be assessed using the following grade system:

Grade	**Criteria**
A	Complete, correct, and clear solution to lab exercises (worth full marks)
B	Most of the standard lab exercises completed, or all completed but with one or more major bugs
C	Partial solution only, most of lab not completed
.	Not attempted

The practical programming sessions will contribute to 15% of your overall course mark. The mark for iteration demonstration of the group project will go towards the mark allocated for your group project.

Group Project

There will one main group project which will run through the whole teaching period and contributes to 30% of the overall course mark. The specification of the group project will be released in week 2. You are required to form groups of 4 (no more than 4) for the group project. Due to lab size numbers, one or two groups of 3 may be permitted in the lab, if necessary.

The group project will be implemented using an agile software delivery mode. As such, your team will be required to build and deliver the project in iterations. Each iteration will deliver a part of the requirements of the project and will encompass all the SDLC activities, namely analysis, design, coding and testing. At the end of the iteration, you (as a team) will demonstrate to your lab class the functionality implemented during that iteration cycle. Changes to project requirements are a natural and unavoidable part of any software project life-cycle. Hence, students will need to bear in mind that project requirements may be subject to change and enhancements to functionalities may be made at the end of each iteration. You will need to carefully design the solution for your current iteration, such that the solution is extensible to accommodate these changes. The lectures will guide you through good design principles to enable you to build modular, extensible solutions.

Each iteration must be demonstrated on the scheduled date. Late demonstrations will not be accepted. A mark will be awarded at the end of each iteration, which will go towards the overall mark for the group project. The final demo of the group project will be conducted in week 13.

Final Exam

There will be a two-hour exam, which will be centrally timetabled and appear in your UNSW exam timetable. The exam will contain a mixture of "theory" questions and multiple choice questions. More specific details of the exam will be provided through the course.

There is a hurdle requirement on the final exam. If you do not score at least 45% on the exam, you cannot pass this course. If your overall course score exceeds 50%, despite scoring very poorly (<40%) on the exam, the hurdle will be enforced via a grade of UF. Of course, if your overall course score is less than 50%, then your grade will be FL.

If you cannot attend the Final Exam because of illness or misadventure, then you must submit a Special Consideration request, with documentation, through MyUNSW within 48 hours of the exam. If your request is reasonable, then you will be awarded a Supplementary Exam. If your overall course mark is between 47 and 50, then you will also be awarded a Supplementary Exam.

Student Conduct

!! New and Important !!

The **Student Code of Conduct** (Information (https://student.unsw.edu.au/conduct) , Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)) sets out what the University expects from students as members of the UNSW community. As well as the learning, teaching and research environment, the University aims to provide an environment that enables students to achieve their full potential and to provide an experience consistent with the University's values and guiding principles. A condition of enrolment is that students *inform themselves* of the University's rules and policies affecting them, and conduct themselves accordingly.

In particular, students have the responsibility to observe standards of equity and respect in dealing with every member of the University community. This applies to all activities on UNSW premises and all external activities related to study and research. This includes behaviour in person as well as behaviour on social media, for example, Facebook groups set up for the purpose of discussing UNSW courses or course work. Behaviour that is considered in breach of the Student Code Policy as discriminatory, sexually inappropriate, bullying, harassing, invading another's privacy or causing any person to fear for their personal safety is serious misconduct and can lead to severe penalties, including suspension or exclusion from UNSW.

If you have any concerns, you may raise them with your lecturer, or approach the School Ethics Officer (mailto:ethics-officer@cse.unsw.edu.au) , Grievance Officer (mailto:grievance-officer@cse.unsw.edu.au) , or one of the student representatives (mailto:stureps.ug@cse.unsw.edu.au) .

Plagiarism is defined as (https://student.unsw.edu.au/plagiarism) using the words or ideas of others and presenting them as your own. UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Plagiarism and Academic Integrity (https://student.unsw.edu.au/plagiarism)
- UNSW Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism. In particular, you are also responsible that your assignment files are not accessible by anyone but you by setting the correct permissions in your CSE directory and code repository, if using. Note also that plagiarism includes paying or asking another person to do a piece of work for you and then submitting it as your own work.

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW staff and students have a responsibility to adhere to this principle of academic integrity. Plagiarism undermines academic integrity and is not tolerated at UNSW. Plagiarism at UNSW is defined as using the words or ideas of others and passing them off as your own.

If you haven't done so yet, please take the time to read the full text of

- UNSW's policy regarding academic honesty and plagiarism (https://student.unsw.edu.au/plagiarism)

The pages below describe the policies and procedures in more detail:

- Student Code Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)
- Student Misconduct Procedure (https://www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf)
- Plagiarism Policy Statement (https://www.gs.unsw.edu.au/policy/documents/plagiarismpolicy.pdf)
- Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

You should also read the following page which describes your rights and responsibilities in the CSE context:

- Essential Advice for CSE Students (https://www.engineering.unsw.edu.au/computer-science-engineering/about-us/organisational-structure/student-services/policies/essential-advice-for-cse-students)

Special Consideration

If your work in this course is affected by unforeseen adverse circumstances, you should apply for Special Consideration through MyUNSW, including documentation on how you have been affected. If your request is reasonable and your work has clearly been impacted, then

- for a practical lab session, you may be granted an extension
- for the final exam, you may be offered a Supplementary Exam

Note the use of the word "may". None of the above is guaranteed. It depends on you making a convincing case that the circumstances have clearly impacted your ability to work.

If you are registered with Disability Services, please forward your documentation to Aarthi Natarajan within the first two weeks of semester.

Assessment

Item	**Marks**
Tutorial Attendance	2%
Quizzes	3%
Lab Exercises	15%
Group Project	25%
Final Exam	55%

Each quiz is worth 2 marks. Your total Quiz mark is capped at 5%. Quizzes are done individually online.

Lab marks vary. Your total Lab mark is computed by summing all of your lab marks and mapping into a mark out of 15. Labs are done in pairs.

Total marks for the group project will be the sum of the marks obtained at each iteration demo and the final demonstration.

Final Mark

Your final mark for this course will be computed using the above assessments as follows:

CourseWorkMark	=	tut mark + quiz mark + lab mark + group project mark	out of 45
ExamMark	=	Final Exam Mark	out of 55
ExamOK	=	ExamMark ≥ 25/55	true/false
FinalMark	=	CourseWorkMark + ExamMark	out of 100
FinalGrade	=	UF, if !ExamOK FL, if FinalMark < 50/100 PS, if 50/100 ≤ FinalMark < 65/100 CR, if 65/100 ≤ FinalMark < 75/100 DN, if 75/100 ≤ FinalMark < 85/100 HD, if FinalMark ≥ 85/100	

Students are eligible for a Supplementary Exam if and only if:

- they cannot attend the final exam due to illness or misadventure
- their final mark is in the range 47 ≤ FinalMark < 50 (in this case, FinalMark is limited to 50)

A Supplementary Exam will not be awarded for any other reason.

Course Schedule

The following is a **tentative** schedule of when course topics will be covered. As this is the first offering of the course, this will most likely vary as we develop the course material.

Week	Lectures
1	Course Overview, Introduction to Software Engineering, Software Development Life Cycle (SDLC), SDLC Development Methodologies Introduction to Python programming
2	Agile Software Model (Agile Manifesto, 12 principles of Agile development, Overview of agile frameworks - Scrum, XP etc) Core Values and Practices of XP, XP Planning Python Programming
3	Team Collaboration, GitHub Guest Lecture by Director of GitHub Introduction to Flask/Jinja 2
4	Requirements Engineering (Requirements analysis/Feasibility Study, Traditional approach to analysis - functional specifications, use-cases, prototypes, agile analysis - user-stories
5	Software Architecture and Design (Overview, Architectural styles, Types of Design (Function-oriented, Object Oriented (OO)), Lean, Agile Design techniques MVC Framework
6	Object Oriented Design Concepts (Classes, Objects, Encapsulation, Abstraction) Python as an OO programming language
7	OO Design Principles - SOLID principles, Modularity, Cohesion etc
8	Database Design and Modelling - Overview of database models, Relational databases, Tables, Primary Keys, Referential Integrity Introduction to SQLite3
9	Database Design and Modelling - Conceptual design, creating a data-model, object relational mapping (ORM) Introduction to SQLAlchemy (an ORM framework)
-	Non-teaching week (mid-semester break)
10	Development and Testing (Refactoring, Test-Driven development, Unit Tests, Integration Tests, User Acceptance Tests), Test frameworks
11	Software Deployment & Maintenance
12	Advanced Software Engineering Concepts Course review, Final Exam Guidelines

Resources for Students

There is no single text book that covers all of the material in this course at the right level of detail and using the same technology base as we are. The lecture notes should provide sufficient detail to introduce topics, and you will then study them in further depth in the tutorials, labs and group project.

There are also many online resources available, and we will provide links to the most useful ones. Some are listed below. If you find others, please post links in the Comments section on the Course Outline page.

Some suggestions for books that cover at least some of the topics in this course

- *Foundations of Software Engineering,* by Ashfaque Ahmed and Bhanu Prasad, CRC Press
- *Software Engineerings* , by Ivan Marsic, Rutgers, The State University of New Jersey
- *Agile Software Development: Principles, Patterns and Practice* , by Robert C Martin, Pearson

Course Evaluation and Development

This course has never run before. It will be evaluated at the end of the semester using the myExperience system. However, you are encouraged to provide feedback during the semester so that we can address any problems ASAP.

End of Course Outline

Resource created 4 years ago (Saturday 22 July 2017, 07:12:20 PM), last modified 4 years ago (Saturday 02 September 2017, 08:58:59 AM).

Comments (/COMP1531/17s2/forums/search?forum_choice=resource/11567) (/COMP1531/17s2/forums/resource/11567)

There are no comments yet.