Course Outline

SENG2011
Software Engineering Workshop 2A
Semester 1 2014

Contents

- Course Details
- Communication with students
- Course Staff
- Course Summary
- Course Aims
- Student Learning Outcomes
- Assumed Knowledge
- Teaching Rationale
- Teaching Strategies
- Assessment
- Academic Honesty and Plagiarism
- Course Schedule
- Resources for Students
- Course Evaluation and Development

Course Details

Important: See this summary for information about the structure of this year's course: it is different from last year.

Check the course *notices* for information about forming groups.

Course Code:	SENG2011
Course Title:	Software Engineering Workshop 2A
Units of Credit:	6
Course WebSite:	http://www.cse.unsw.edu.au/~se2011/
Handbook Entry:	http://www.handbook.unsw.edu.au/undergraduate/courses/2014/SENG2011.html
Hours per week:	3 plus a mentor meeting
Prerequisite:	SENG1031
Corequisite:	COMP2111
Lectures and location:	Consult the course timetable.

Communication with students

The course *WebCMS* web page will be the main source of announcements and information: in particular, there will not usually be "bulk emails" to the whole class, even if details change.Thus you *must* check the course *WebCMS* web page regularly, in particular the notices section, for updates and announcements – if you miss an important announcement because you did not check the notices web page, no special dispensation will be granted.

In addition to the UNSW email address, the School of CSE itself provides students with a login account and email. If you have a different email account that you prefer, you should redirect your School account to there: otherwise you might miss important announcements. (Use the mlalias command for that.)

Course Staff

Contact Details

Staff Name	Role	Email	Phone
Carroll Morgan	Course Convenor	carrollm at cse.unsw.edu.au	55317

Project management

Vincent Pang from the Australian School of Business will be teaching the project management component

Course Summary

This course teaches practical design and structuring techniques for computer programs that help us to proceed from informal but precise *requirements* through to *implementations* that are easy to understand and maintain. Unusually, we do not take the traditional route of proposing special notations or computer-based tools for this; instead we concentrate on concepts and ways of thinking.

Rather than having conventional tutorials, tutors will operate in the style of mentors, with fixed groups of students assigned to each mentor. Assignments however will be marked individually (unlike previous years). Students will be expected to justify their programs to other students, in a presentation style during lectures. There is no exam.

In the first 3/4 of the semester the lectures, presentations and mentor-meetings will be organised around a sequence of from-requirements-to-program examples that will become increasingly demanding. All but the last assignment will be based on similar examples.

Only in the last assignment will we look briefly at a program-verification tool Dafny; and some lectures will be devoted to setting up and becoming familiar with that tool.

There will also be lectures and an assignment on Project Management: see the course calendar.

Parallel instruction in the underlying methods used informally in SENG2011 will come from COMP2111.

Course Aims

- Engineering: learn how to construct an abstract specification/model of a software system, applying both informal and formal approaches, and how to take it towards an implementation.
- Project Management: to learn the basics of working in a software-project team so that you can deploy effectively the engineering techniques above while working in collaboration with others.

Student Learning Outcomes

After completing this course, students should:

- be able to build a rigorous specification of a program or program-component's required behaviour;
- have an appreciation of the role of software engineering in software-system development;
- be able to reason abstractly about requirements and be able to model them using various informal/formal methods; and
- have learned that both behavioural (human based) and technical views (computer based) of design and implementation are important issues.

Assumed Knowledge

- elementary discrete mathematics
- programming to an introductory level in a simple procedural language (e.g. C, Python or similar)
- competence with Unix (e.g. Linux) commands, and the operating-system structure as presented to its users.

Teaching Rationale

See Section 6 in the description of this year's course.

Teaching Strategies

SENG2011 is a workshop in which the teaching strategy is primarily to communicate techniques and to give experience. There are fewer than usual formal lectures in the style of explaining knowledge and facts.

The lectures that *are* in this course are to provide students with support and guidance in developing their practical skills. Students will be expected and encouraged to stand before the class to explain and justify the approach they have taken to developing particular programs from their requirements.

The lecture schedule is tentative and may vary during the semester as the need arises. Changes to the lecture schedule will be announced on the course website.

The remaining contact with teachers will primarily be with group *mentors*, who will give detailed guidance on understanding how to describe rigorously what a program should do and then to guarantee that the program does it. Students will be organised into groups of 6–8, and as groups are *required* to spend an average of 30 minutes with their group mentor per week, at a time suitable to the mentor but arranged in constellation with her/his students.

It is recommended that these mentor sessions be held in one of the small seminar rooms in the CSE building, but this will depend on availability. It is essential to find a time suitable for all members as attendance at (mentor)

group meetings is compulsory.

Assessment

The total course mark will be combined from marks for these components:

- Specification/programming assignments (60% of total mark): There will be a number of these assignments, no more than 4 but possibly fewer. Their due dates will be announced during the semester, and their marks are expected to be as follows:
  - Assignment 1: 10.
  - Assignment 2: 10.
  - Assignment 3: 15.
  - Assignment 4: 25.

  That gives a total of 60%.
- Project management (20% of total mark): More detail here shortly.
- Participation (20% of total mark): These marks will be awarded based on mentors' reports from their meetings during the term, and from the lecturer's observation of students during lectures and especially presentations.

  Note that, as a special case, if a student attends fewer than half of his/her mentor sessions, then the mark for the *whole course* can be capped at 64%, that is just less than a credit.

- Submissions handed-in late by up to one day (i.e. 24 hours) will be scaled by 85%; for two days it is 75% (that is total scaling, not in addition to the 85%); for three it is 65%; for four it is 50%; and after four days the submission will not be accepted at all (i.e. is scaled by 0%).

Academic Honesty and Plagiarism

Plagiarism is defined as "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:

- Learning Centre: Plagiarism and Academic Integrity
- MyUNSW: Plagiarism and Academic Misconduct
- CSE: Addendum to UNSW Plagiarism Guidelines
- CSE: Yellow Form (whose terms you have agreed to)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism.

Course Schedule

- Week 1 — Allocation of groups; outline of the course structure and aims. Introduction to static reasoning for straight-line programs.
- Week2 2–12 —TBA.

Resources for Students

The main resources for out-of-lecture study will be lecture summaries, and literature (often online) that will be identified as the course progresses.

In-depth references are as below, but they adopt in general a much more formal approach than the one we will be using:

- Edsger Dijkstra: A discipline of programming
- David Gries: The science of programming
- Edward Cohen: Programming in the 1990Â's
- Roland Backhouse: Program construction and verification
- Edsger Dijkstra and Wim Feijen: A method of programming
- Anne Kaldewaij: Programming, the derivation of algorithms

See the above as an indication of what is possible in later years, if you should specialise in programming for its own sake; do not take them as what is required this year.

Course Evaluation and Development

This year (2014) is the first time the course has been run in this form. It is however loosely based on the course COMP 6721: Informal methods that ran in the years 2010–12.