Introduction

- This course is taught by Michael Oudshoorn and Ray Babcock.
- Ray’s consulting times are advertised on the web and outside his door.
- Michael is available by appointment through Jeannette Radcliffe in the Departmental Office.
  - It is difficult for the Head of Department to set fixed office hours due to various meetings.
  - Don’t let the need for an appointment stop you from seeking help.
- A Bulletin Board is being set up for this subject and you are encouraged to submit questions to it.
  - Questions and answers are visible to everyone.
  - You’re encouraged to read it regularly.
  - It will also be used for class-wide announcements.

What is our background?

- Ray has many years of experience as a software engineer working on various projects before becoming an academic.
- Michael has consulted to numerous companies in Australia, Asia and the US. This consulting includes teaching various aspects of Software Engineering to staff of these companies.
- Both have a detailed knowledge of what it takes to build large software systems.
  - Building the right system right!
- We plan to pass those skills onto you through the lectures etc. and by providing you with realistic experience in your assignment.

What is your background?

- Ideally, you should have completed:
  - CS351 Software Engineering
  - CS352 Software Engineering and Project
  - CS324 Programming Techniques
  - ENGL223 Technical Writing
- You will be proficient at coding and the use of data structures and algorithms.

What is the subject all about?

- CS351/352 examines software engineering in the small, while CS460/461 examines software engineering in the large.
- Software engineering in the large, is those aspects of software engineering needed to design, develop and maintain large complex systems.
- We will discuss the skills needed to do this in lectures and you will get the chance to put them in practice through your assignment.

Objectives

- The course aims to produce graduates that:
  - Are able to design, document, build and integrate complex software systems.
  - Understand the issues involved in software engineering in the large including:
    - Requirements analysis
    - Functional specification
    - Effort estimation
    - Test strategies
    - Verification and validation
    - Build and debug a large software system
  - Communicate and present thoughts and ideas in a clear, precise manner
  - Follow relevant standards
  - Work in a team environment
  - Develop efficient and correct code
Our hopes

- By the time students finish this course they should be highly employable.
- Students should have a portfolio to show prospective employers to illustrate their abilities.

Course structure

- The course is paired with CS461 and students must complete both courses in the same year.
- The course will consist of lectures, discussions and a large project with several deliverables – more on this soon.
- The course requires verbal presentations and written documentation to be produced.
- The project is open-ended and offers the opportunity to utilize knowledge you have gained in other subjects – networks, compilers, operating systems, artificial intelligence, graphics, …
  - Indeed your general knowledge of computer science will typically shape the design and implementation of the solution you produce.

Indicative lecture topics

- Lifecycle and development models
- Requirements gathering
- Functional specifications
- Design
- Formal methods
- Safety critical software
- Project planning and management
- Cost and effort estimation/models
- Fault tolerance
- Metrics
- Verification and validation; test plans
- Standards
- CASE tools
- Capability Maturity Model
- Risk analysis

Corequisite

- CS460 Senior Design Project I must be taken with CS461 Senior Design Project II.
- This allows the lecture material to be delivered over Fall and the project to span Fall and Spring giving you a realistic experience.
- The courses are a lot of work (more on this shortly), but it is a lot of fun, and you gain many valuable skills which will assist you in any large software project you get involved in.

Tutorials

- We aim to have a small number of tutorials/discussions. This will be the opportunity for you to review another group’s work and pass constructive criticisms.
  - It is also an opportunity to see what others are doing that is better than what you are doing and to “borrow” ideas.

Indicative project

- The project is challenging and large.
- It is probably larger than you should tackle over 2 semesters.
  - You need to determine what is important and what you can reasonably accomplish.
- The project varies each year.
- It is open-ended.
- We expect each solution will be different.
- The actual project details will be provided in a few weeks.
- It is a group-based project and you will need to cooperate with your group members.
Group-based

- The project is group based. You will not get to choose your own team.
- Everybody has to contribute to all aspects of the project.
- You should not let your team down through poor performance on your behalf.
- At the end of the project, each group member will need to tell the instructors what the effort distribution was across the group. The group project mark will be distributed across individuals in a manner consistent with the effort each member put into the project.
  - If you let your group down through lack of performance during the year, it will be reflected in your grade!

Project deliverables

- Over CS460 and CS461 you will need to deliver many things, including:
  - Requirements document
  - Functional specification
  - Several talks/presentations
  - Prototype demonstration
  - Final demonstration
  - User manual
  - Minutes of meetings
  - Web page for the project
  - CV’s
  - Bid for the job including cost estimation
- Complete details and deadlines will be provided at a later date.

Assessment

- Assessment is based on:
  - Project (40%)
  - Quizzes (10%)
  - Midterm exam (20%)
  - Final exam (30%)
- Note that the project extends over 2 semesters. The percentages above relate to the work completed this semester only.
- A serious attempt must be made at each component.
  - In particular, a 50% average across each of the following is required to pass the subject:
    - Quizzes
    - Project/assignments
    - Exams

Resources available

- Web pages found at www.cs.montana.edu/courses/460.
- Visit the web pages often and check for updates.
- You will find:
  - A bulletin board
  - Lecture plan with links to the lecture slides
  - Past exam papers (Fall 2003) and an exam produced by Michael at the University of Adelaide so you can see what kinds of questions he might ask. NOTE that you may not be able to answer all of the questions since it is an exam for a different course!

Consulting help

- There is no lab with a TA to assist you in this course!
- You are regarded as almost ready to graduate and will be treated as such.
- We expect you to solve most of your own problems!
  - Read the textbook
  - Use the library
  - Use the internet
  - Talk to your peers
- A consultant will be available in a consulting room to offer limited assistance with coding problems.
  - Visits to the consultant should be recorded in your timesheets.

Textbooks

- There are no fixed textbooks for this course. Like real-life, you need to decide if you need one, and if so, which book best suits you.
- We recommend the following as good books that cover the majority (but not all) of the lecture material.
  - Ian Sommerville, “Software Engineering”, Addison-Wesley
**Exam**

- The mid-term exam will be held approximately half way through the course. It will cover material seen up until that point.
- The final exam will be held during the exam week. It will cover the whole of the course.
- All exams may cover all material covered in this course up to that stage:
  - Lectures
  - Assignments
  - Tutorials
  - Readings
  - Discussions
- Lecture attendance is strongly recommended. Note that the lecture slides made available on the web are merely the instructors notes – they are not a substitute for you taking your own notes during class, or reading textbooks, etc.

**Our expectations of you**

- Ask questions
- Interact with the instructors
- Use each other as a resource
- Exchange ideas
- Experiment
- Contribute to your group
- You should take notes in lectures – the slides are merely our notes and may sometime be terse!

**Workload**

- This is a 3 credit course. It should therefore occupy about 8-9 hours of your time each week.
- Lectures occupy 3 hours/week, which leaves about 5-6 hours/week on your project.
- You should expect to spend about 130 hours on this course over the Fall semester.

**Client/manager meetings**

- There are several “meetings” scheduled during the course between the instructors and the groups.
- During these meetings the first half of the lecture timeslot, the meeting will be devoted to a meeting between the group and the “client” – it is your chance to find out about the requirements the client has etc.
- The second half of the lecture timeslot, the meeting will be between the group and the “manager”.
- The instructors will wear different “hats” at each meeting and will appear ignorant of what took place in earlier meetings.

**Deadlines**

- The project has a number of deadlines.
- These are firm deadlines and significant grades are lost for being late – 20% of the available marks each day or part thereof.
  - Very much like a penalty clause in a contract.
- Deadlines will be at a specific time and late penalties applies from 1 second past the deadline. The system clock on esus will be treated as the official clock for the course.
- No excuses will be accepted – including machine failure, busy labs, …

**Hint**

- “Design for change” should be your mantra.
- Expect it, plan for it, design for it.
- We may change the requirements during the project without extending the deadline – just like a real customer.
- When talking to Michael and Ray please make it clear which hat you want them to wear – the customer or the instructor.
  - Depending on which hat they wear, you may get different advice, reactions or behaviour!
- Don’t expect sympathy – expect the unexpected and prepare and plan for it.
What is software engineering?

- Not just programming!
  - Covers all aspects of the systematic construction of software with a specific purpose. "Multi-person construction of multi-version software".
- Large software projects
  - Usually a collection of programs used over a long period.
  - 25,000 – 10,000,000+ lines of code.
  - Development teams of 5-200 people.
  - Maintained by many generations of staff.
- Never completely understood by one person.
- Components of bigger systems.

Problems for Software Engineers

- Dealing with large systems.
- Software complexity
- Project complexity
  - Interactions with clients.
  - Coding and management.
  - Meeting deadlines
- Organizational problems
- Cost overruns
- Deficiencies of current software engineering practice:
  - Lack of good metrics and measurement tools.
  - Lack of standards.

Why Software Engineering?

- Software is pervasive
  - Banks, shops, cars, institutions, government, homes, ...
- Software is no no means "perfect"
  - Malfunctions are common. E.g.,
    - Spreadsheet program destroys database.
    - Computer controlled jail doors won’t lock.
  - Malfunctions are sometimes dangerous. E.g.,
    - Mercedes with “graceful-stop no-skid” brake computer:
      - 120m (390ft) skid mark, 1 passenger dead.
    - Airbus crash into a forest at an air show outside of Paris.
- Software projects are frequently out of control:
  - US Navy engages GE for software for frigates in 19983.
  - Frigates built in 1986/7 but no software until 1988.
  - GE sacked, Nobel-Tech gets contract and delivers software in 18 months.

Software lifecycle

- Requirements Analysis
- Design & Specification
- Coding & Module Testing
- Integration & System Testing
- Delivery & Maintenance

Software lifecycle “student view”