The University of W.A.
School of Computer Science & Software Engineering

First Semester 2003

Software Quality and Measurement
670.400

26 lectures & 12 tutorial/workshops by

Terry Woodings
Senior Teaching Fellow

Room 4.18 9380 2618
terry@ee.uwa.edu.au

Introduction

This course will look at the application of the theory of software engineering to industry. Commercial systems are generally the work of teams of specialists and are of sufficient scale that it is not realistically possible for a single person to comprehend all their inherent complexity. The course considers software development and maintenance as a process which can be defined, measured, modelled and optimised. In particular, it will consider the assessment of quality, reliability and usability. There will be some discussion of safety critical issues as well as standard techniques for the production of systems for the private or public sector.

Course Outcomes:

At the conclusion of the course, students should understand software development as an industrial-strength process: its management, planning, control, measurement and improvement, as well as specific issues such as estimation, planning and quality assurance. Students should develop skills in VV&T (validation, verification and testing), the use of ISO standards, user contact, configuration management, the design and use of metrics, reliability modelling and risk management. Students will gain an awareness of the human and professional issues (such as privacy), regarding the impact of new systems on organisations and society at large. SQM400 is NOT a programming unit. It indicates that programming is only one aspect of software engineering.

Prerequisites:

This course is built on (and replaces) Software Quality and Reliability 407 (taught up to 2002). Students need to have a strong knowledge of programming theory and practice, such as that indicated by completing Software Requirements and Project Management 300 *or* equivalent.

Course style

There will be two lectures and a tutorial/workshop per week with lab classes as needed for particular exercises. The class is small enough to be able to run lectures in a fairly interactive mode. This course is neither a programming unit nor about learning specific tools - it is concerned with software development techniques and the underlying theories and concepts. Students are expected to do some reading of the text (Pressman or Sommerville) and of supplementary material each week.

Timetable

Two lectures per week at 10.00 and 11.00 on Wednesdays in the Ross LT.
There will also be a Tutorial/workshop at a time to be arranged at the first lecture. Attendance at the workshop is essential to cover all the practical elements of the course.

Text books

Roger Pressman: "Software Engineering - A Practitioner's Approach", 5th edition (European Adaptation), McGraw Hill, 2000. The 4th edition (1997) is also acceptable.

An excellent (optional) alternative and source of supplementary material is:
Ian Sommerville: "Software Engineering", 6th edition, Addison Wesley, 2001. The 5th edition (1995) is also acceptable.

Each student will also need to obtain a copy of **HB90.9-2000** "Software Development - Guide to ISO 9001:2000" - available online from the University of WA library

Syllabus

Week	**Lecture topics**
1	The concept of a software process and quality principles
2	Basic techniques for assuring software systems
3	Estimation (planning and control depend on estimation skills)
4	Standards and procedures
5	Validation of requirements
6	Verification and testing as a controlled process
7	Configuration management and change control
8	Data integrity and measurement theory
9	Measurement and testing of usability
10	Measurement and certification of software products
11	Measurement of the software process
12	Human factors – the client, the manager, the software team
13	Assessment of an organisation’s capability and strategic management

Assessment

There are two specific assignments. The first, worth 25%, is a group project on estimation, designed to demonstrate skills in development scheduling and measurement for project control. The second, worth 20% and in two parts, assesses documenting procedures together with designing and implementing software quality metrics.

As most of the students will be employed as professional software engineers at the end of the year, the standard of presentation of assignments and projects will be at a level suitable for putting before a Managing Director. Also, in industry, there is usually a preference for an adequate job done early than a perfect job done late. Accordingly, projects may be marked on a sliding scale allowing extra marks for early submissions. That is, there will be a small adjustment of marks (up to 1% per day) for early or (down by 5% per day) for late submission of projects. The marking algorithm will be discussed in class and will be clearly indicated on relevant project sheets.

Students are expected to maintain a Practical Work Folder in which they will keep notes of all tutorial work and a log of software engineering papers read. This Folder will be audited twice during the semester and is worth 5% of the final mark.

There will be a three hour (open book) exam at the end of the semester worth the other 50% of the final mark. Items permitted in the exam room will be a file of lecture notes, up to three text books, assignments and the Practical Work Folder.

To pass this course a student is expected to achieve a satisfactory performance in both the Continuous (Assignments + Practice folder) and Examination components of the course.

Practical Work Folder

As with other professionals, engineers are expected to maintain a casebook or Practical Work Folder containing:

- notes of all practical/tutorial work;
- a log of research papers read.

This Folder should be available for inspection at the tutorial and will be audited during the semester. It should be handed in for assessment at the end of the course and is worth 5% of the final mark.

1. Practical Work. Each week, students will be doing some practical work associated with the topics of the course, normally in preparation for the tutorial. They should record what they did, discussions, answers to questions etc., from each week's tutorial in their Practical Work Folder. Students will also have a few exercises to complete at non-timetabled times. Worksheets will be provided for these exercises and students are expected to formally write-up their results/experiences/conclusions.

2. Reading Log. It is important that students develop the habit of maintaining a current knowledge of their areas of expertise. Each student is required to give some time to browsing in the library and reading at least one journal article or new book chapter (on some aspect of software engineering of the student's choosing) per week. Evidence of this will be by means of a reading log. Information in the log should include author, article title, journal or book title, date published, date read, a note on the author's theme or thesis and the student's evaluation of the item with regard to its relevance in the context of software engineering. The reading log should be maintained as a word-processed document with a printed version of each entry on a fresh page. The twelve pages may not include chapters of the text book or articles from trade magazines.

As a result of the readings, at the end of the semester, each student will append to the reading log a two page appreciation of the state of the art in the topic chosen. This will indicate the current state of knowledge, what is within the capability of software engineers and what is at present infeasible. It should list a number of currently open research questions.

Some (40) useful authors are: Vic Basili, Barry Boehm, Lionel Briand, Fred Brooks, Michael Cusumano, Tom DeMarco, Edsger Dijkstra, Khaled El-Emam, Michael Fagan, Richard Fairley, Norman Fenton, Dan Freedman, Tom Gilb, Robert Glass, Robert Grady, Maurice Halstead, Bill Hetzel, Tony Hoare, Watts Humphrey, Darrel Ince, Ross Jeffery, Capers Jones, Barbara Kitchenham, Nancy Leveson, Michael Lyu, Harlan Mills, John Musa, Jakob Nielsen, David Parnas, Shari Pfleeger, Walker Royce, Norman Schneidewind, Gordon Schulmeyer, Martin Sheppard, Ben Shneiderman, Rob Thomsett, Gerald Weinberg, Niklaus Wirth, Ed Yourdon, Horst Zuse.

For sets of major original papers, students could check the tutorial collections produced by Richard Thayer (editor), such as: "Software Engineering Project Management" published by the IEEE in 1990 and 1997.

Plagiarism and Collusion

Group activities are an effective form of learning and are encouraged. Several of the assignments and some tutorial work - where indicated on project assignment sheets - may be done as a group. Always give credit to other peoples' work that has been included in any assignment. Plagiarism, cheating and unauthorised collusion are regarded seriously and the student's attention is drawn to the University's policies on these issues.

End of document