COMP2041 Course Introduction

Software Construction: Techniques and Tools

Aims

This course aims to explore techniques and software tools that can be used in the production of high-quality software.

By the end of the course, you should:

- understand the software development process and appreciate the use of specific technologies and strategies during different phases of software development
- be able to implement software systems via the Unix shell and Perl scripting languages
- be able to implement components of software systems in the C, Java and Haskell programming languages
- be able to choose appropriate interfaces for users to interact with software systems and for software systems to interact with each other
- be able to experimentally analyse properties of existing programs, both in terms of correctness (by testing) and efficiency (by profiling),
- be able to improve the properties of existing programs, both in terms of correctness (by debugging) and performance (by tuning)
- be able to understand the configuration and installation of software systems under the Unix environment that use Make, CVS and Configure scripts

Pre-requisites

It would be preferable if you had passed COMP2011 Data Organisation before attempting this course, simply to have a greater level of programming experience. However, you should be able to cope ok as long as you have a good pass in COMP1021 Computing 1B. You can do COMP2011 concurrently with COMP2041 if you wish. COMP2021 Digital Systems Structures is not specifically relevant for this course, but, once again, if you have already done it, the extra experience with programming will help you.

Web Pages

The course web pages will provide links to all material related to this course. The pages can be accessed via:

From any CSE lab workstation:	file:/home/cs2041/web/index.html
From anywhere on the Internet:	http://www.cse.unsw.edu.au/~cs2041/

You should become familiar with the structure and content of the course web pages early on, and consult them frequently. In particular, all important announcements will be posted to the Course Noticeboard on the opening page.

Staff

Name	Room	Phone	Email	Role
John Shepherd	Elec Eng 325	9385 6494	jas	Lecturer-in-charge, administration

Syllabus

The following is a summary of the topics that will be covered in this course. For more information (e.g. level of detail for each topic) consult the course Web pages.

1. Qualities of software systems
   - Correctness, clarity, reliability, efficiency, portability, ...
2. Techniques for software construction
   - Analysis, design, coding, testing, debugging, tuning
   - Interface design, documentation, configuration
3. Tools for software construction
   - Programming languages (C, Java, Haskell)
   - Scripting languages (Perl, Shell, CGI)
   - Filters (awk, sed, grep, tr, ...)
   - Analysis tools (debuggers, profilers)
   - Configuration tools (make, rcs, cvs)

The focus for the practical work will be on C and Perl; however, we will still expect you to be able to produce Java and Haskell programs if needed.

Reading Material

The following book is the recommended text for this course:

- Kernighan & Pike, *The Practice of Programming*, Addison-Wesley, 1998.
  (Gives theory, philosophy and some tool details)

Another useful and interesting book that you might consider buying:

- McConnell, *Code Complete*, Microsoft Press, 1993.
  (Ok, ok, so it's Microsoft Press ... but it has many interesting case studies and good ideas)

Other useful reference books include the following:

- Peek, O'Reilly, Loukides, *Unix Power Tools*, O'Reilly, 1997.
  (Comprehensive guide to common Unix tools)
- Robbins, *Unix in a Nutshell: System Vr4 and Solaris 7*, O'Reilly, 1995.
  (Concise guide to Unix and its toolset)
- Loukides & Oram, *Programming with GNU Software*, O'Reilly, 1997.
  (Concise guide to the GNU toolset (gcc, gdb, make, gprof))
- Wall et al., *Programming Perl* (2nd ed), O'Reilly, 1996.
  (Invaluable Perl reference manual)
- Sebesta, *A Little Book on Perl*, Prentice Hall, 1999.
  (Neat little book on Perl, if you don't want a big heavy one)
- Christiansen et al., *Perl Cookbook*, O'Reilly, 1998.
  (Lots and lots and lots of interesting example Perl programs)
- Kernighan & Pike, *The Unix Programming Environment*, Prentice Hall, 1984.
  (Pre-cursor to the textbook, intro to Unix tools)

For pointers to other useful reading material, including documentation for all of the tools used in the practical work, see the course Web pages.

Classes

Each week there will be a one-hour lecture on Tuesday and a two-hour lecture on Thursday. Lectures will be used to present the theory and practice of the techniques and tools in this course. There will be extensive use of case studies and practical demonstrations during lectures. Lecture notes will be available on the course Web pages before each lecture; you should print them out and bring them along to annotate during the class.

Lecture timetable:

Day	**Time**	**Location**
Tuesday	4pm-5pm	EE LG1
Thursday	3pm-5pm	CLB 7

Each week you will also be expected to attend a one-hour tutorial session to clarify ideas from lectures and work through exercises based on the lecture material. Tutorial registration will occur via Sirius in week 2.

Following the tute class each week, there will be a two-hour lab class, during which you will work on a variety of small practical problems involving the tools introduced in lectures. Because this course is practical in nature, lab class are a very important component, and you should make every effort to attend the labs and complete the exercises diligently.

Assignments

In the assignments, you will work through the process of building and/or modifying software systems, using the tools and techniques described in lectures. The first assignment will involve Perl and CGI, while the second assignment will involve C programming, debugging and performance tuning. Each assignment is worth 10%, so the assignments together contribute 20% of the overall mark for the course.

Ass	**Description**	**Due**	**Marks**
1	Perl/CGI	Sept 8	10%
2	C programming	Oct 27	10%

Assignments are to be completed individually and without assistance from anyone except the lecturers and tutors. Plagiarism will not be tolerated; assignment submissions will be checked for similarity and anyone found guilty of submitting assignment work that they did not do themselves will be *awarded a mark of zero* for the course.

Exams

There will be a two-hour practical exam, to be held in the CSE labs on Saturday 4 November. During this exam, you will be required to solve three small programming problems, similar in nature to those completed in the lab classes. It is a

hurdle requirement for this course that you *correctly* solve at least two of the three programming problems. You will be provided with a testing system in the exam to allow you to check whether your program is correct or not. The prac exam contributes 30% of the overall mark for the course.

There will also be a two-hour closed-book written exam for this course in the November exam period. The exam will consist of a mixture of multiple-choice, descriptive questions, quantitative analysis, and small programing exercises, and will include material related to the assignment work. You will not be expected to remember the details of programming languages used in the course; reference information will be provided along with the exam paper, giveing a summary of any language that we expect you to use. Written exam programming questions will be marked leniently with respect to to precise syntax. The written exam contributes 40% of the overall mark for the course.

Assessment

The course assessment includes components from lab work, project work, a prac exam and a final written exam, in the following proportions:

Component	**Ident**	**Value**		**Component**	**Ident**	**Value**
Lab Work	`lab`	10%		Assignments	`ass`	20%
Prac Exam	`prac`	30%		Final Exam	`exam`	40%

Your final mark in this course will be based on a combination of the above components. Note: both the Prac Exam and the Final Exam are hurdle requirements. If you fail to achieve a satisfactory standard in either of them, you will fail the course. The required standard in the Prac Exam is a mark of at least 20/30 (i.e. two completely functioning programs). For the Final Exam, you must achieve a pass (20/40) on the scaled exam marks.

The following formula describes precisely how the mark will be computed and how the hurdle will be enforced.

```
labs   = mark for lab exercises  (out of 10)
ass    = mark for assignments    (out of 20)
prac   = mark for practical exam (out of 30) [scaled]
exam   = mark for written exam   (out of 40) [scaled]
okExam = prac >= 20/30 && exam >= 20/40
total  = labs + ass + prac + exam

final  = total,         if okExam
       = min(45,total), otherwise
```

John Shepherd, July 2000