UNSW Course Outline

# COMP2041 Software Construction: Techniques and Tools - 2024

Published on the 12 Feb 2024

## General Course Information

**Course Code :** COMP2041
**Year :** 2024
**Term :** Term 1
**Teaching Period :** T1
**Is a multi-term course? :** No
**Faculty :** Faculty of Engineering
**Academic Unit :** School of Computer Science and Engineering
**Delivery Mode :** In Person
**Delivery Format :** Standard
**Delivery Location :** Kensington
**Campus :** Sydney
**Study Level :** Undergraduate
**Units of Credit :** 6

**Useful Links**
Handbook Class Timetable

## Course Details & Outcomes

### Course Description

This course is designed for students who have mastered the basics of programming. It aims to broaden your knowledge of techniques and tools for software construction. It covers: Unix filters, shell scripting and Python (for programming), git (for version control), docker (for portable deployment), package managers (for configuration and deployment). At the end of this course, you should be able to build moderate-sized software systems and configure them so that others can download and deploy your work.

## Course Aims

This course aims to expand your programming and system configuration ability to the point where you can put your own software on github and have others make use of it. It is core for the Bioinformatics and Software Engineering programs, and is a popular elective in other CSE programs. It is also valuable in giving you exposure to common industry tools.

## Course Learning Outcomes

| Course Learning Outcomes |
|---|
| CLO1 : Solve problems by writing Unix shell scripts and Python programs |
| CLO2 : Build software systems using a range of techniques and strategies |
| CLO3 : Understand when to use of specific technologies and strategies during software development |
| CLO4 : Use tools for version control, system configuration, debugging, and performance improvement |
| CLO5 : Articulate and communicate effectively concepts related to programming and systems |

| Course Learning Outcomes | Assessment Item |
|---|---|
| CLO1 : Solve problems by writing Unix shell scripts and Python programs | • Tests<br>• Labs<br>• Assignments 1 and 2<br>• Exam |
| CLO2 : Build software systems using a range of techniques and strategies | • Tests<br>• Labs<br>• Assignments 1 and 2<br>• Exam |
| CLO3 : Understand when to use of specific technologies and strategies during software development | • Tests<br>• Labs<br>• Assignments 1 and 2<br>• Exam |
| CLO4 : Use tools for version control, system configuration, debugging, and performance improvement | • Tests<br>• Labs<br>• Assignments 1 and 2<br>• Exam |
| CLO5 : Articulate and communicate effectively concepts related to programming and systems | • Tests<br>• Exam |

## Learning and Teaching Technologies

WebCMS3

# Assessments

## Assessment Structure

| Assessment Item | Weight | Relevant Dates |
|---|---|---|
| Tests<br>Assessment Format: Individual | 10% | Due Date: Week 3 - Week 10 |
| Labs<br>Assessment Format: Individual | 15% | Due Date: Most weeks |
| Assignments 1 and 2<br>Assessment Format: Individual | 30% | Due Date: Week 7 and Week 10 |
| Exam<br>Assessment Format: Individual | 45% | Due Date: During Exam Period |

## Assessment Details

### Tests

<u>Assessment Overview</u>

There will be weekly tests from week 3 onwards, designed to give you timely and realistic feedback of your understanding of the course material. Tests may be programming exercises, multiple choice questions, or both.

These will be conducted in your own time under self-enforced exam-like conditions. Each test will specify the conditions, but typically these will include:

1. no assistance permitted from any person;
2. a time limit of one hour;
3. no access to materials (written or online) except specified language documentation or man pages.

Each test is worth 1.7 marks, and will be automarked. Your total mark for the tests component is computed as a sum of your best 6 of 8 test marks. Any violation of the test conditions will result in a mark of zero for the entire test component.

<u>Course Learning Outcomes</u>

- CLO1 : Solve problems by writing Unix shell scripts and Python programs

- CLO2 : Build software systems using a range of techniques and strategies
- CLO3 : Understand when to use of specific technologies and strategies during software development
- CLO4 : Use tools for version control, system configuration, debugging, and performance improvement
- CLO5 : Articulate and communicate effectively concepts related to programming and systems

## Labs

### Assessment Overview

Following the tutorial class each week, there will be a two-hour laboratory 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, laboratory classes are **important**: if you do not put a great deal of effort into the lab classes, you risk failing the final exam.

Each week, there will be one or more exercises to work on. These exercises will be released in the week preceding the lab class.

Completed exercises need to be submitted. You must submit exercises before the deadline using **give** to obtain a mark for a lab exercise, The usual lab exercise submission deadline will be Monday 12:00 noon but some lab exercises may have an extended deadline.

Most lab exercises will be automarked. There will be partial marks for attempts which do not pass autotests. Tutors will separately provide feedback on your approach to the problem and on the style of your solution. Some labs may contain exercises which will be assessed during the lab.

Submission of any work that it is not your own will result in an automatic mark of zero for the entire lab component.

The lab exercises for each week are worth in total 2 marks. All of your lab marks will be summed to give you a mark out of 18; if their sum exceeds 15, your total mark will be capped at 15.

Challenge exercises may be specified for some labs. Challenge exercises may be involve concepts not covered in lectures and range in difficulty from not-very-hard to almost-impossible.

The contribution of challenge exercises to lab marks will be limited to 20%

Hence you can obtain almost full marks (95+%) for the lab component without completing challenge exercises.

<u>**Course Learning Outcomes**</u>

- CLO1 : Solve problems by writing Unix shell scripts and Python programs
- CLO2 : Build software systems using a range of techniques and strategies
- CLO3 : Understand when to use of specific technologies and strategies during software development
- CLO4 : Use tools for version control, system configuration, debugging, and performance improvement

## Assignments 1 and 2

**Assessment Overview**

There are two assessable programming assignments. Assignments give you the chance to practice what you have learnt on relatively large problems (compared to the small exercises in the labs). Assignments are a *very important* part of this course, therefore it is essential that you attempt them yourself.

- Assignment 1, on shell programming; worth 15%
- Assignment 2, on Python programming; worth 15%

The assignment weighting and deadlines may be slightly varied when the assignment designs are complete.

Assignments are automarked, and also have feedback from tutors on programming style and software structure.

**Course Learning Outcomes**

- CLO1 : Solve problems by writing Unix shell scripts and Python programs
- CLO2 : Build software systems using a range of techniques and strategies
- CLO3 : Understand when to use of specific technologies and strategies during software development
- CLO4 : Use tools for version control, system configuration, debugging, and performance improvement

## Exam

**Assessment Overview**

There will be a three-hour final exam, held in-person, in CSE computer labs, during the exam period. This will be centrally timetabled and appear in your UNSW exam timetable. This is a closed-book exam; we provide some reference material.

The exam will contain short-answer questions, which may require you to read code, and implementation tasks, where you will be required to write code.

During this exam you will be able to execute, debug and test your answers.

The implementation tasks will be similar in nature to those encountered in lab exercises.

You will not be expected to remember all the details of programming languages used in the course; reference information will be provided along with the exam paper, giving a summary of any language that we expect you to use. You also have access to the Unix programmers manual.

There is a hurdle requirement on the final exam. If you do not score at least 40% (18.0/45) on the exam (after scaling), 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.

**Course Learning Outcomes**

- CLO1 : Solve problems by writing Unix shell scripts and Python programs
- CLO2 : Build software systems using a range of techniques and strategies
- CLO3 : Understand when to use of specific technologies and strategies during software development
- CLO4 : Use tools for version control, system configuration, debugging, and performance improvement
- CLO5 : Articulate and communicate effectively concepts related to programming and systems

## General Assessment Information

**Grading Basis**

Standard

# Course Schedule

| Teaching Week/Module | Activity Type | Content |
|---|---|---|
| Week 1 : 12 February - 18 February | Lecture | Course introduction; Unix filters. |
| Week 2 : 19 February - 25 February | Lecture | Unix filters (cont'd); Shell programming. |
| Week 3 : 26 February - 3 March | Lecture | Shell programming (cont'd). |
| Week 4 : 4 March - 10 March | Lecture | Introduction to Version Control with Git. |
| Week 5 : 11 March - 17 March | Lecture | Practical Python programming. |
| Week 7 : 25 March - 31 March | Lecture | Python programming. (cont'd). |
| Week 8 : 1 April - 7 April | Lecture | Git in more detail |
| Week 9 : 8 April - 14 April | Lecture | Tools for System Configuration & Deployment |
| Week 10 : 15 April - 21 April | Lecture | Revision; Exam Information. |

## Attendance Requirements

Students are strongly encouraged to attend all classes and review lecture recordings.

# Course Resources

## Recommended Resources

There is no required textbook for the course. Useful (both for this course and beyond) reference books include:

- Brian W. Kernighan and Rob Pike. 1998. *The Practice of Programming*. Addison-Wesley.
  (The inspiration for 2041, both in overall philosophy and in tool details.)
- Steve McConnell. 2004. *Code Complete* (2nd ed.). Microsoft Press.
  (Many interesting case studies and practical ideas.)
- Stephen G. Kochgan and Patrick Wood. 2013. *Unix® Shell Programming*. Sams Publishing.
  (A careful introduction to shell programming.)
- Carl Albing and JP Vossen. 2017. *bash Cookbook* (2nd ed.). O'Reilly Media.
  (An example-based introduction to shell programming.)
- Shelley Powers, Jerry Peek, Tim O'Reilly, and Mike Loukides. 2009. *Unix Power Tools* (3rd ed.). O'Reilly Media.
  (A comprehensive guide to common Unix tools.)
- Mike Loukides and Andy Oram. 1996. *Programming with GNU Software*. O'Reilly Media.
  (A tutorial on the GNU programming tools: gcc, gdb, etc.)
- Arnold Robbins. 2009. *Unix in a Nutshell* (4th ed.). O'Reilly Media.
  (A concise guide to Unix and its toolset.)
- Brian W. Kernighan and Rob Pike. 1984. *The Unix Programming Environment*. Prentice-Hall.
  (Predecessor to *The Practice of Programming*; a good introduction to Unix tools.)

Pointers to other useful reading material, including documentation for all of the tools used in the practical work, are provided from the course web pages.

## Course Evaluation and Development

The course will be evaluated at the end of the session using the myExperience system; feedback will be used to make continual improvements to the course. Students are also encouraged to provide informal feedback during the session, and to let course staff know of any problems as soon as they arise. Suggestions will be listened to openly, positively, constructively, and thankfully, and every reasonable effort will be made to address them.

The myExperience feedback from the last course offerings was very positive.

Based on it we are looking to add more explanation to sample solutions, and make course exercises more relevant to working on containers and virtual machines/cloud infrastructure.

## Staff Details

| Position | Name | Email | Location | Phone | Availability | Equitable Learning Services Contact | Primary Contact |
|---|---|---|---|---|---|---|---|
| Convenor | Andrew Taylor | | | | | Yes | No |
| Administrator | Dylan Brotherston | | | | | No | Yes |

## Other Useful Information

### Academic Information

#### I. Special consideration and supplementary assessment

If you have experienced an illness or misadventure beyond your control that will interfere with your assessment performance, you are eligible to apply for Special Consideration prior to, or within 3 working days of, submitting an assessment or sitting an exam.

Please note that UNSW has a Fit to Sit rule, which means that if you sit an exam, you are declaring yourself fit enough to do so and cannot later apply for Special Consideration.

For details of applying for Special Consideration and conditions for the award of supplementary assessment, please see the information on UNSW's Special Consideration page.

#### II. Administrative matters and links

All students are expected to read and be familiar with UNSW guidelines and polices. In particular, students should be familiar with the following:

- Attendance
- UNSW Email Address
- Special Consideration
- Exams
- Approved Calculators
- Academic Honesty and Plagiarism
- Equitable Learning Services

#### III. Equity and diversity

Those students who have a disability that requires some adjustment in their teaching or learning environment are encouraged to discuss their study needs with the course convener prior to, or at the commencement of, their course, or with the Equity Officer (Disability) in the Equitable Learning Services. Issues to be discussed may include access to materials, signers or note-takers, the provision of services and additional exam and assessment arrangements. Early notification is essential to enable any necessary adjustments to be made.

**IV. Professional Outcomes and Program Design**

Students are able to review the relevant professional outcomes and program designs for their streams by going to the following link: https://www.unsw.edu.au/engineering/student-life/student-resources/program-design.

*Note: This course outline sets out the description of classes at the date the Course Outline is published. The nature of classes may change during the Term after the Course Outline is published. Moodle or your primary learning management system (LMS) should be consulted for the up-to-date class descriptions. If there is any inconsistency in the description of activities between the University timetable and the Course Outline/Moodle/LMS, the description in the Course Outline/Moodle/LMS applies.*

## Academic Honesty and Plagarism

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW 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.*

Plagiarism is a type of intellectual theft. It can take many forms, from deliberate cheating to accidentally copying from a source without acknowledgement. UNSW has produced a website with a wealth of resources to support students to understand and avoid plagiarism, visit: student.unsw.edu.au/plagiarism. The Learning Centre assists students with understanding academic integrity and how not to plagiarise. They also hold workshops and can help students one-on-one.

You are also reminded that careful time management is an important part of study and one of the identified causes of plagiarism is poor time management. Students should allow sufficient time for research, drafting and the proper referencing of sources in preparing all assessment tasks.

Repeated plagiarism (even in first year), plagiarism after first year, or serious instances, may also be investigated under the Student Misconduct Procedures. The penalties under the procedures can include a reduction in marks, failing a course or for the most serious matters (like plagiarism in an honours thesis or contract cheating) even suspension from the university. The Student Misconduct Procedures are available here:

www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf

## Submission of Assessment Tasks

Work submitted late without an approved extension by the course coordinator or delegated authority is subject to a late penalty of five percent (5%) of the maximum mark possible for that assessment item, per calendar day.

The late penalty is applied per calendar day (including weekends and public holidays) that the assessment is overdue. There is no pro-rata of the late penalty for submissions made part way through a day. This is for all assessments where a penalty applies.

Work submitted after five days (120 hours) will not be accepted and a mark of zero will be awarded for that assessment item.

For some assessment items, a late penalty may not be appropriate. These will be clearly indicated in the course outline, and such assessments will receive a mark of zero if not completed by the specified date. Examples include:

- Weekly online tests or laboratory work worth a small proportion of the subject mark;
- Exams, peer feedback and team evaluation surveys;
- Online quizzes where answers are released to students on completion;
- Professional assessment tasks, where the intention is to create an authentic assessment that has an absolute submission date; and,
- Pass/Fail assessment tasks.

## Faculty-specific Information

Engineering Student Support Services – The Nucleus - enrolment, progression checks, clash requests, course issues or program-related queries

Engineering Industrial Training – Industrial training questions

UNSW Study Abroad – study abroad student enquiries (for inbound students)

UNSW Exchange – student exchange enquiries (for inbound students)

UNSW Future Students – potential student enquiries e.g. admissions, fees, programs, credit transfer

**Phone**

(+61 2) 9385 8500 – Nucleus Student Hub

(+61 2) 9385 7661 – Engineering Industrial Training

(+61 2) 9385 3179 – UNSW Study Abroad and UNSW Exchange (for inbound students)

## School Contact Information

**CSE Help! - on the Ground Floor of K17**

- For assistance with coursework assessments.

**The Nucleus Student Hub** - https://nucleus.unsw.edu.au/en/contact-us

- Course enrolment queries.

**Grievance Officer** - grievance-officer@cse.unsw.edu.au

- If the course convenor gives an inadequate response to a query or when the course convenor does not respond to a query about assessment.

**Student Reps** - stureps@cse.unsw.edu.au

- If some aspect of a course needs urgent improvement. (e.g. Nobody responding to forum queries, cannot understand the lecturer)