SE Degree Review 2024 - Draft 4 (Teaching Commitee)

Version 2.0 11/09/2024

Appendix G: COMP613 structure

Introduction

Description

In the past, UNSW School of Computer Science and Engineering (CSE) has set up a working group on a regular basis to look at software engineering (SE) undergraduate degree and make recommendations for reviewing SE's existing structure and offerings in the light of developments in the field. In 2021, the last review made a number of recommendations based on evidence from difference perspectives as well as constraints in the university/school.

This draft report describes the preliminary findings of the 2024 review.

Review Participants

SE Cluster

Yuchao Jiang <yuchao.jiang@unsw.edu.au>;Fethi Rabhi <f.rabhi@unsw.edu.au>; Armin Chitizadeh <a.chitizadeh@unsw.edu.au>; Basem Suleiman <b.suleiman@unsw.edu.au>; Yuekang Li <yuekang.li@unsw.edu.au>; Yulei Sui <y.sui@unsw.edu.au>; Helen Paik <h.paik@unsw.edu.au>; Rachid Hamadi r.hamadi@unsw.edu.au; Ali Darejeh ali.darejeh@unsw.edu.au

Other CSE academics / Senior students

Wayne Wobcke, Carroll Morgan, Jake Renzella, Rani Jiang (rani.jiang@unsw.edu.au)

Industry Review Panel

- Daniel Wirjo (AWS), wirjo@amazon.com;
- Rob Pike robpike@gmail.com;
- Yacine Rabhi (ESS) yacine.softprodev@gmail.com;
- Nick Patrikeos (Atlassian) npatrikeos@atlassian.com;
- Chinmay Manchanda (Tyro) chinmay.manchanda08@gmail.com;
- George Wright (Nine Publishing) gwright@publishing.nine.com.au;
- Alan Hsiao (Cognitivo) Alan.Hsiao@cognitivo.com.au;
- Lawrence Yao (ROAR Software) lawrence.yao@unswalumni.com
- Andrew Gerrand andrewdg@gmail.com

Review Process

The adopted process will be similar to the process in previous reviews, get feedback from different perspectives and do a synthesis of the recommendations:

1. Different working groups considering different knowledge areas and the connection to the software engineering body of knowledge (SWEBOK) make some recommendations. For 2024 review, the different sub-groups are:
    a. Future of SE specialised courses: Yulei (Lead)
    b. Process & DevOps: Basem (Lead)
    c. Project management across the courses: Yuchao (Lead)
    d. Core course review (DESN2000, COMP1531): Fethi (Lead)
    e. Formal methods review: Carroll (Lead)
    f. COMP2511 Review: Alvin Cherk (Lead)

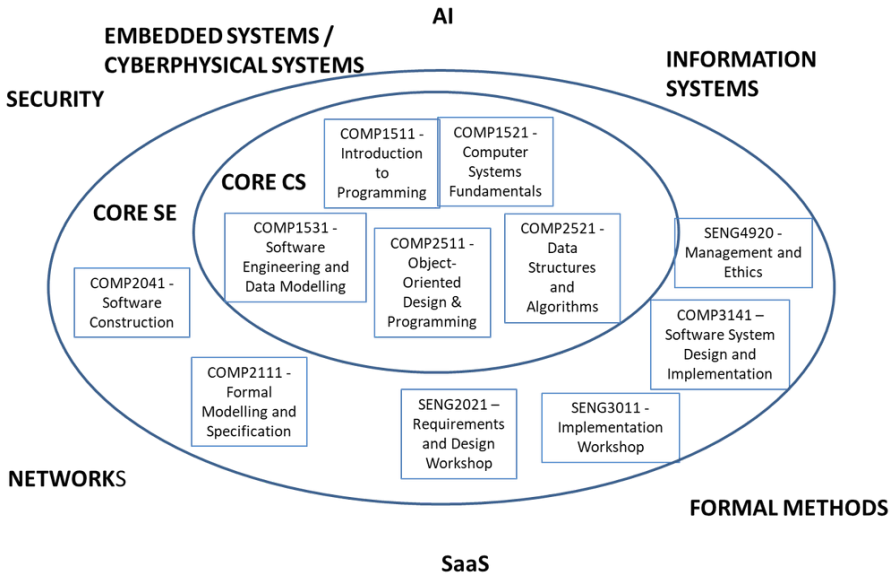
g. Co-Op Review - Imran Razzak (Lead)

2. Student representatives representing the student body giving their feedback. A meeting has taken place on 23 August 3pm. (Minutes in Appendix D)
3. Industry consultation committee representing the views of prospective employers comment of these recommendations**.** A meeting has taken place on 27 August 12-1.30pm. (Minutes in Appendix E)
4. Other CSE academic staff and teaching committee will approached for further feedback and for approval of final recommendations

Existing SE degree structure

The degree structure is described in the handbook Handbook - Software Engineering (unsw.edu.au)

It is illustrated in the diagram below:



Recommendations from 2021 Review

Recommended short-term solutions from 2021 review

Solutions	Actions
Making an existing course (COMP6721) a core course in the second-year software engineering degree and carry updates of courses in the formal methods specialization accordingly. Bridging material needs to be added, preferably in COMP1511.	(pending results from Formal Methods sub-group)
Making DESN2000 focus on User-Centred product design, Requirements Engineering (RE) and early stages in the software lifecycle and adjust SENG2021 workshop accordingly.	Actioned.
Re-brand COMP2511 to “Software Design and Architecture”	Not actioned.
Adjust SENG2021/SENG3011 workshops by introducing more on project management, design notations, some	Actioned.

DevOps.	
Evaluate the material in the core content to see if security issues are properly discussed throughout the development cycle.	Not actioned
Replace the existing core course (SENG4920) by DESN3000.	SENG4920 is now changed into COMP4920.
Incorporate discussion of security, risk and ethics in DESN2000 and DESN3000.	Not actioned.
Investigate students doing a project working on an existing code base rather than building a new system from scratch. Participation is an open-source project is a possibility that could be supported by some of our industry partners.	Partly done via introducing DevOps in SENG3011
Investigate opportunities with School of Information Systems, Technology and Management in the area of delivering course material on Security Management that could be included in existing courses or offered as INFS electives (e.g. INFS2701 Cyber Security Management and Governance and INFS4907 Managing Security and Ethics in Cyberspace).	Not actioned.

Recommended long-term solutions from 2021 review

Solutions	**Actions**
Looking at the cohesion of COMP1531, COMP2511, SENG2021, SENG3011 and the possibility of a new course, in the areas of software design, development and management to balance content.	Not actioned.
Ensuring that the following topics are addressed as part of the core SE degree: 1. Concurrency 2. Parallel and distributed computing 3. Cloud computing 4. Testing and Software Quality 5. Software performance, building software at scale, high integrity systems, performance modelling and analysis 6. Continuous Integration, DevOps, Effort estimation, release management, software product line development 7. Software systems analysis (e.g. qualitative analysis, simulation), model checking, metrics 8. Web Front-end Programming 9. Security of software systems	• COMP3142 addressed 4. • SENG workshops addressed 6. • COMP6080 addressed 8, but it is not a core course. • COMP6131 Software Security Analysis addressed 9 • COMP6452 Blockchain systems mentions quality attributes and performance (links to 2) -

Other non-curriculum recommendations

Pending from 2016

Issue	Comment
Resourcing the Software Engineering degree is a very urgent issue that needs to be addressed. Many core courses in Software Engineering don't have a permanent lecturer in charge. Leaving them to casuals at a time when they have been significantly revised will significantly affect the quality of the delivery of these courses at a time when student enrolments are growing.	Partly actioned by recruitment of 2 EF staff, 1 casual and 1 academic in SENG. Still insufficient due to increase in students numbers and introduction of project work in other degrees serviced by SENG staff.

2021 recommendations (not related to curriculum)

Issue	Comment
Students do not have sufficient information about the degree. Resources are needed to complete a Handbook that was started in 2016 containing useful information to help them navigate through various choices.	Not actioned.
Students require increased support for finding Industrial Placements	Not actioned.
Mentors for many project-based courses are hired at the last minute, would be better to recruit mentors very early to brief/train them properly	Partly actioned by offering contracts to tutors much earlier than before.
Students would benefit from having the same tools used across multiple courses (e.g. Github/Jira).	Partly actioned as Jira/Confluence and GitHub are increasingly used
Leveraging industry connections to strengthen some of the existing courses is highly recommended.	Partly Actioned.

Recommendations from Co-Op Review 2024

Co-Op scholarships are available to SENG students (See Appendix 😎. A Co-Op review has been conducted in 2024 and the report is presented as Appendix C.

The main conclusions are:

- All industry partners agree that placement is valuable to develop industry-relevant skills.
  - Action: keep the industry placement
- All industry partners felt that the workplace experience and industry relevant skills and professional network development are the most important benefits of the program.
  - Action: keep the industry placement
- Most industry partners recommended having better alignment between academic requirements and IT placements.
  - Action: introducing more industry-related tools, practices and topics, as well as probably industry-related projects.

- Some industry partners reported “misalignment of academic assessment with agile environment and realties of work on placement”.
  - Action: teaching and aligning Agile practices and tools with industry environments.
- Most believe the academic requirements of IT placements do not align
  - Action: introduce more industry-related and improved academic requirements for the placement program.
- Heavy reporting requirements and difficulty to assess student’s learning from the placement through assessable tasks
  - Action: identify better assessment to assess student’s learnings throughout courses and the industry placement
- Change the course requirements for IT, moving from a traditional single report to weekly reflections sharing and synthesising ideas on application of skills in industry to better suit the variety of placement experiences and potentially share fresh insights of industry applications of skills with their peers.
  - Actions: Introducing weekly reflections on student’s learning from application of skills and sharing and synthesising ideas. Agile management and communication --> Sprint demos and retrospectives
- Potentially introduce a regular drop-in time (in-person & online) that scholars can access support from the School if confused about progression or need academic support.
  - Actions: Introduce weekly consultation and help sessions to support students from academic point of view

Summary of 2024 feedback received and suggested actions

This is all feedback received from different sources. This is arranged in the form of themes.

Theme 1: Project Management

There seems to be many issues related to how much and how we should cover project management.

Issue	Origin	Possible Action
Project management is inadequately covered in particular 1. Teamwork (including communication, conflict management, collaboration, role delegation, agenda making) 2. Manage stakeholders 3.Documentation	Project management sub-group	COMP1531 covers Agile basics and give students some practice. SENG workshops will reinforce agile practices. DESN2000 looks at managing requirements. Comp9820 is a new course for MIT which can have some content useful for SE. COMP4920 needs to be checked to see how much PM content is there. Is there a recommendation we can tell SE students who want to take electives as part of a specialisation (Team Leadership ?).
	Core Modules sub-group (Rachid).	PM content has to be added in several core courses. An elective course is not enough for SE students. PM content can be based on Kathy Shwalbe textbook - Information Technology Project Management. 9th Edition. Cengage. 2018
	Industry review	Need to reinforce planning and tooling. Make students able to work with PMs rather than turning them into PMs.
	Co-Op Review	Introducing weekly reflections on student’s learning from application of skills and sharing and synthesising ideas

Project management principles are not covered enough	Students feedback	Adding lectures to project courses is not effective as students prefer focus on project work (where grades are)

Theme 2: Core SE fundamentals

There are disagreements as to what constitutes core principles in software engineering: more programming or breadth of principles in areas such as agile methods, RE, testing etc.

COMP1531 has deviated from SE fundamentals and covers more topics especially connected to Web Design	Core Modules sub-group	Refocus COMP1531 around the SE cycle. Web project should be just a way to illustrate the different phases. A synopsis is listed as Appendix A.
	Students feedback	Having stable teaching staff for this course will help preventing deviations
	Students feedback	Needs better integration between project work and the theory principles. This requires finding a good project that does teaches maximum principles while avoiding unnecessary efforts. Also the order of activities (theory/practice) has to be designed accordingly.
		Instead of having a single project, students should have the flexibility to do a project in an area that interests them (e.g. design or testing)
		The use of co-pilot to reduce programming burden is a possibility but raises many concerns
		Make this course more practical helps students get internships after 1st year
	Industry review	Foundational knowledge is important. Make students work on an existing codebase. Reinforce testing before coding. No Copilot should be allowed.

Theme 3: Software Design & HCI

How to address the gap in teaching students different aspects of software design and HCI

Gaps in design fundamentals and software architectures	Recommendations from 2021 Review	COMP1531 can introduce these concepts. They could potentially be reinforced in COMP2511. There is a course COMP6452 Software Architecture for Blockchain Applications but very specialised. Perhaps could be made more general

Should COMP2511 be a design or programming course ?	COMP2511 revision subgroup	Course needs revising to have a clearer learning outcomes. Either do more advanced programming or focus solely on design. The decision has to be aligned with COMP1531 revision.
COMP2511 teaches low-level design	Students feedback	Teaching high-level (e.g. architectural) design can be better done with a project
		Finding ways to do practical design activities is desired. How can it be done without going into full implementation which is time consuming ?
		Need teaching staff with expertise to implement the above requirements.
COMP2511 has some gaps in dependency Injection/Mock testing and mock objects and async programming (promises, async/await etc.)	COMP2511 revision subgroup	Include these topics in revised COMP2511
Teaching Design in a practical way is difficult	Industry Review	Navigating a large code base and extracting designs is also a good way to teach design. Using queing theory and simulation tools is a good way to teach students how to test designs and make a design course more practical
Students have difficulties understanding the separation between Business Requirements and Design	Industry review	Students need to see the different ways a system can be designed from the same set of requirements to understand the difference.
		Using APIs to teach design is recommended.

DESN2000 has overlappings with HCI course	Core Modules sub-group	Reduce overlapping: use Axure instead of Figma as it requires programming skills
	Students feedback	Teaching Axure is not a good idea. Either stick to Figma or go more into the programming.
DESN2000 comes after SENG2021 which is about design. Should be the other way round	Core Modules sub-group	Swap DESN2000 (moved from T2 to T1) and SENG2021 (moves from T1 to T2). Otherwise, do less requirements in SENG2021 (and remove Figma as they will do it again in DESN2000).

Theme 4: Interesting specialisations

Students in SE are not finding advanced subjects or electives to allow them to specialise in certain areas.

Teaching DevOps practices in SENG3011 has been well received but creates many technical challenges	DevOps sub-group	Need to some basic devops concepts before the workshop. Where ? (1531 does it but too basic). Requires separate budget. Some expert mentors and technical assistance (possibly from industry) too.
	Students feedback	Heavily relying on expert tutors creates a risk when they leave. Few students do PhDs because not interested in academic careers. Given Faculty has launched industry-based PhDs, could new types of positions across industry-academic be considered ?
Insufficient project courses	Students feedback	Involving student societies to create projects in which computing students are helping build software for other students (e.g. business students)
We need to strengthen our elective offerings in the area of DevOps/AI/Security/Reliability.	Specialisations sub-group	Offer a new DevOps Quality Assurance course, focusing on AI systems in general and large language models in particular. There is already a proposal for ML Engineering course (Jake) so content needs to be aligned.
	Industry review	DevOps is a complex area, but teaching it will be hard. Learning to work in a DevOps environment is more important.
Misalignment of academic assessment with agile environment and realities of work on placement	Co-Op Review	Teaching and aligning Agile practices and tools with industry environments.
Expose students to software engineering for AI or ML (SE4ML)	Specialisations sub-groupnee	A new elective course (similar to DESN2000 in structure) could look at developing AI applications using a range of off-the-shelf technologies (not an AI course). There is already a proposal for LM Engineering course (Jake) so content needs to be aligned.
Expose students to experience in developing large systems	Students feedback	Need more lecturers with expertise in microservices and cloud architectures

If COMP1531 removes Web design material, we need a pathway for software engineers to learn UI/UX design course and front-end stacks	Specialisations sub-group	COMP6080 could will relieve COMP1531 and provide technical expertise in this space in the form of an elective. Could possibly align with DESN2000 use of Axure.
COMP3141 as a core course is not adapted to the needs of SE students. As we have a new course on software quality	Specialisations sub-group	Make COMP3141 specific to Functional Programming. In this case, it needs to

(COMP3142), this one will be core so we may need to reduce other core courses.		become elective ? (but part of Formal methods specialisation)
MIT students need more SE skills	Specialisations sub-group	Create a SE specialisation for MIT students comprising COMP3142 (Software Testing and Quality Assurance) and other courses such as new COMP6131 Software Security Analysis and COMP3141 Functional Programming, and Algorithmic Verification COMP3153/COMP915
Students lack experience working with large projects	Industry review	Create course allowing students to work on large codebase.

Theme 5: Gaps

Fundamental things we should be teaching but we don't

Gaps in user experience, human factors	Recommendations from 2021 Review	DESN2000 has addressed this gap to some extent.
Gaps in rigorous (though informal) reasoning about program construction	Recommendations from 2021 Review	
Gaps in requirements engineering and business analysis	Recommendations from 2021 Review	DESN2000 addresses some of the gaps.
	Industry review	The most important thing for SE students is to bridge the gap between BA teams and Dev teams.
Gaps in dealing with security in general	Recommendations from 2021 Review	Partly addressed by new course COMP6131 Software Security Analysis
Gaps in Cloud Computing/platform engineering: although it partly covered in SENG, CSE students struggle without some of these core skills.	Industry review	This content could go into COMP2041 (core SE, popular CS elective). COMP2041 many skills are superseded by LLMs. Atlassian is offering to support (CI/CD, deployment platforms, etc.)
	Industry review	Distributed system design, especially with APIs. Also connecting with business services to designs is important.
Students lack ability to configure virtual networks.	Industry review	Creating virtual platforms should be taught, especially network configuration in a cloud environment.

Review recommendations

Divided into 3 areas:

1. On-line monitoring
2. Recruitment
3. Curriculum-related recommendations

4. Long-term actions

Courses Portfolio and On-line quality monitoring

Going forward, it is suggested that a dedicated Software Engineering Cluster within CSE will be responsible for reviewing the degree on an on-going basis.

The existing SENG courses portfolio is shown below:

Course code and name	**Term Offerings**	**Staff 2025**	Load
COMP1531 Software Engineering Fundamentals	T1,T2,T3	Still being sorted	0.3
COMP2041/9044 Software Construction	T1,T2	Permanent	0.25
COMP2511 Object-Oriented Design and Programming	T1,T2,T3	Casual	0.2
COMP3142 Software Testing and Quality Assurance	T3	Permanent	0.2
COMP3900/9900 Computer Science Project	T1,T2,T3	Permanent	0.2
COMP4920 Professional Issues and Ethics in Information Technology	T1,T3	Permanent	0.2
COMP6080 Web Front-End Programming	T1,T3	Casual	0.2
COMP6131 Static Analysis for Software Security	T2	Permanent	0.2
DESN1000 Introduction to Engineering Design and Innovation	T1,T3	Permanent/Casual	0.2
DESN2000 Engineering Design and Professional Practice - SENG	T2	Casual	0.2
SENG2011 Workshop on Reasoning about Programs	T3	Casual	0.2
SENG2021 Requirements and Design Workshop	T1	Permanent	0.2

SENG2991 Software Workplace Practice 1	T1 (Co-Op)	Permanent	0.05
SENG3011 Software Engineering Workshop 3	T1	Permanent	0.2
SENG3993 Software Workplace Practice 2	T1,T2 (Co-Op)	Permanent	0.05
SENG3994 Software Workplace Practice 3	T2,T3 (Co-Op)	Permanent	0.05
TOTAL			2.9 (1.7 permanent, 1.2 casual)

Recruitment

Recruitment of SENG staff is still urgent.

A preliminary analysis of the gaps based on the SWEBOK shows 3 gaps as listed in the table below. Research areas have been added that align with the curriculum recommendations.

SWOBOK AREA	Teaching Area	Research Area
Chapter 1: Software Requirements	o Software systems analysis (e.g. qualitative analysis, simulation), model checking, metrics o Semantic technologies	Requirements engineering for AI systems Designing Ontologies and Knowledge Graphs for Business Systems
Chapter 2: Software Design and Chapter 3: Software Construction	o Parallel and distributed system design o Service-Oriented and Cloud computing o Security by design of software systems o Business Process Modelling and Execution o Concurrency	Entreprise and Data Architectures for Large Scale Data Processing / AI pipelines Designing Fintech and Regtech solutions
Chapter 8: Software Engineering Process	o Software performance, building software at scale, high integrity systems, performance modelling and analysis o Continuous Integration, DevOps, Effort estimation, release management, software product line development	MLOps and new DevOps approaches for AI/ML systems. DevOps4AI and AI4DevOps

Curriculum-related actions

#	Recommendation	Description	Action
1	Refocus COMP1531 around the SE cycle.	• Project must be minimum to reinforce principles. • Make the project and principles more explicitly connected. • Improve the content about testing.	Yuchao, Yuekang

		• Suggest outline in Appendix A.	
2	Conduct COMP2511 revision	• rename course as software design and architecture • focus on principles and alignment with SWEBOK • explore new forms of experimentation that do not involve coding (like architecture evaluation techniques) • replace the java with REST services • add business process management	Ashesh
3	Review project management contents	• focus mainly on collaboration, planning, tooling and practices that are relevant to SE • check learning outcomes across the multiple courses in relation to the above • providing consistent support with tooling (for example, Jira/Confluence resources, accessible by all courses) • check compliance with accreditation requirements	SE cluster
4	Make HCI teaching more relevant to SE	• make DESN2000 HCI part more technical • investigate alternatives to Figma	SE cluster
5	Improves DevOps teaching	• conduct SENG3011 revision in light of recent introduction of DevOps • Investigate shared DevOps resources for multiple courses with support from industry	SE cluster
6	Explore offering an advanced requirements engineering and design course	• Advanced business analysis and requirements modelling techniques • Capturing non-functional requirements such as scalability, reliability, security • Dealing with trade-offs between these considerations. SEs need advanced design techniques that build on (new) COMP2511 • Designing modern cloud architectures • Case studies in different application areas: Finance, Health, IoT (with industry speakers)	SE cluster
7	Increase offerings in the area of software engineering for AI systems	• Adapt existing SE techniques in the AI age • Possibilities include: methods (MLOps), requirements engineering, modelling, testing, compliance checking etc.	Individual LICs
8	Teach more on cloud Computing and platform engineering	• Consider teaching system and network configuration using the cloud • Could be undertaken as part of COMP2041 revision	Individual LIC
9	Conduct COMP3141 revision	• Considering course will become an elective	Individual LIC

		• Make the course specific to Functional Programming • Align with other changes in the Formal Methods area	
10	Improve teaching of programming fundamentals	• Non SENG Students need to be better equipped with programming skills to undertake the capstone project	Computer Science Cluster
11	Improve evaluation of project/internship work	• Learn from Co-Op experience and tranfer lessons to other courses • Investigate assessment alternatives and introduce other techniques (self-reflections)	SE Cluster

Long-term recommendations

For expanding our course offerings, possibilities to investigate include:

- Review electives from other Schools (e.g. ISTM)
- UNSW Founders have some material that could be relevant (e.g. RAPID DIGITAL PROTOTYPING workshop will take students through the landscape of various digital prototyping tools.). Armin had experience, they were good but need advance notice.
- Involving industry partners and adjuncts in teaching. Many are willing but sustaining it is hard.
- Using material from companies (e.g. AWS academy)
- Leverage societies (CS students being used by Business students to build s/w)

Appendix A: Proposed COMP1531 structure

The proposed COMP1531 learning outcomes are as follows:

1. Software development lifecycle

- What and how - W1
- development methods and tools (including teamwork and git) - W2

2. Software Requirements - W3

- Use cases, User stories, Validation
- UX, human factors

3. Software Design Part 1: Conceptual modelling - W4
4. Software Construction - W5

- Javascript or python
- Package management (npm or pip)

5. Software Design Part 2 - W7

- Design fundamentals and architecture design
- Examples of design (Persistence (file-based database), HTTP servers)

6. Software Testing and Quality - W8

- Unit testing
- Linting (eslint)

7. Software Engineering Process - W9

- CI/CD, copilot

8. SE Professional Practice - W10

- SE (or general) project management
- Documentation

Appendix B: Co-op Sample Study Structure

Year	Term 1	UOC	Term 2	UOC	Term 3	UOC
1st	MATH1081 Discrete Mathematics DESN1000 Engineering Design and Innovation MATH1131 Maths 1A OR MATH1141 Higher Maths 1A	6 6 6	COMP1511 Programming Fundamentals COMP1521 Computer Systems Fundamentals General Education	6 6 6	COMP1531 Software Engineering Fundamentals COMP2521 Data Structures and Algorithms MATH1231 Maths 1B **OR** MATH1241 Higher Maths 1B	6 6 6
	Total UOC	18	Total UOC	18	Total UOC	18
2nd	**SENG2991 Co-op Industry Training 1** SENG2021 Requirements and Design Workshop	**6** 6	MATH2400 Finite Mathematics MATH2859 Probability and Statistics DESN2000 Engineering Design and Professional Practice General Education	3 3 6 6	COMP2511 Object-Oriented Design & Programming SENG2011 Software Engineering Workshop 2A COMP3311 Database Systems -	6 6 6
	Total UOC	12	Total UOC	18	Total UOC (nominal)	18

3rd	**SENG3993 Co-op Industry Training 2A** SENG3011 Software Engineering Workshop 3	**6** 6	**SENG3993 Co-op Industry Training 2B** **SENG3994 Co-op Industry Training 3A** COMP3141 Software System Design and Implementation	**6** **6** 6	**SENG3994 Co-op Industry Training 3B** COMP3331 Computer Networks and Applications	**6** 6
	Total UOC	12	Total UOC	18	Total UOC	12
4th	COMP4951 Research Thesis A Discipline Elective Software Construction Techniques and Tools	4 6 6	COMP4952 Research Thesis B Discipline Elective Level 4 Free Elective	4 6 6	COMP4953 Research Thesis C SENG4920 Management and Ethics Discipline Elective Level 4	4 6 6
	Total UOC (nominal)	16	Total UOC (nominal)	16	Total UOC (nominal)	16

Appendix C: Software Engineering Co-Op Program Review – Stakeholders Survey 2024

Sponsor Feedback (4 responses, 3 companies represented)

Overall feedback themes

· Quality of students is the standout feature of the program.

· No negative effects of the integration of IT and shortening of the program.

· Continued challenge of scholars being more valuable on placement the further they are into their degree, but ideally completing all placements in time for graduate recruitment processes in the beginning of their final year.

· Number of Co-op scholars dropping out is a concern. It seems the diverse candidates are the ones more likely to drop out of the Co-op program.

· One sponsor would like to see more diverse candidates picked for scholarships (i.e. from underrepresented backgrounds, low/er socioeconomic background).

· One sponsor would like more say in allocations.

Summary: Unfortunately a small number of respondents, however sponsor companies appear to be satisfied with the change in Software Engineering Co-op program structure overall. They have observed no negative impacts of the integration of graded IT placements into the

structure. If scholars undertake their IT placements in the scheduled timing they complete their final placement prior to the recruitment period, which is preferred by Sponsors.

Scholar feedback (10 responses, SEN20, 21 and 22 represented)

Beliefs about the changes:

- All agree with or are neutral to the integration of IT into the degree.
- Most scholars strongly agree with the condensed 4-year program change, three say the workload is too intense.
- Most scholars like the new placement timing, one scholar would have liked to have had placement in fourth year after more courses.

Support provided:

- Most were satisfied with Co-op office support, one scholar disagreed about feeling supported but did not provide any further explanation.
- Most felt adequately prepared for their placements, one scholar was neutral.
- Very mixed feedback about satisfaction levels regarding School support, reports of confusion about course requirements, concerns about progression checks and difficulty contacting their Academic Coordinator for advice/support.

Value of the Program:

- All scholars agree that placement is valuable to develop industry-relevant skills.
- All felt that the workplace experience and industry relevant skills and professional network development were the standout out features of the program.

Academic requirements:

- Most believe the academic requirements of IT placements do not align, find the reporting requirements too heavy, difficult to align the realities of their placement and/or redundant.
- Very mixed response regarding ability to demonstrate their learnings through the assessable tasks.

Concerns included:

- Progression checks needed and difficult to get
- Consistent marking
- Misalignment of academic assessment with agile environment and realties of work on placement
- Reporting requirement is too heavy
- Interstate placements are an imposition

Scholar suggestions for improvements:

- Weekly blogging to replace one major report
- Higher payments occur during placement and no payments during study to reflect the effort put in
- Increase communication in 2nd and 3rd year (placement years)
- Reduce the 24 weeks placements so that students are less prone to burn out

Recommendations based on feedback:

· Change the course requirements for IT, moving from a traditional single report to weekly reflections sharing and synthesising ideas on application of skills in industry to better suit the variety of placement experiences and potentially share fresh insights of industry applications of skills with their peers.

· Provide greater clarity on the course requirements and expected value of completing the academic components

· Potentially introduce a regular drop-in time (in-person & online) that scholars can access support from the School if confused about progression or need academic support.

Appendix D: Student Feedback

Meeting with Student Representatives - 23 Aug 2024 Summary (Draft 2)

Theme 1: Project Management

- Students prefer a practical approach to learning project management, focusing on industry preparation It's crucial to avoid the path currently taken by ISTM, which places excessive focus on theoretical aspects. This approach ultimately fails to add significant value to students' education or future careers.
- One suggestion was to cover project management theory in COMP3900 after gaining practical experience on other courses.
- Mentioned reasons for low lecture attendance:
- Prioritizing working on their project instead.
- Time constraints and shorter period to work on assignments, especially with trimesters.
- Cost-benefit analysis of attending lectures vs. watching them online, considering the long commute
- Redesign lecture content, particularly in COMP3900, to have a tangible effect on grades (similar to COMP1531).
- Create a history of previous course changes and analyse reasons behind previous changes before implementing new ones to avoid drifting back again in the future.

Theme2: Core SE Fundamentals

- Current tutors believe COMP1531 content is well-covered but could benefit from clearer explanations and emphasis on key topics and with an emphasis on key topics highlighted by the academic board
- Students suggest an effective learning approach that combines practical experience with theory. They recommend:
- Starting with hands-on project work for the first two iterations.
- Gradually introducing theoretical concepts throughout.
- Focusing more on theory in the final iteration.

This method allows students to gain practical skills while keeping theoretical concepts in mind, ultimately helping them understand the connections between different components. By reflecting on theory at the end, they can solidify their understanding of how practice and theory interrelate.

- Use bonus marks to explore various software engineering areas beyond backend development.

Suggested focus areas for teams mentioned by former tutors/mentors:

- Frontend design for a new feature/improvement e.g. wireframe, and quick frontend with html and css (no functionality needed)
- Software Architecture (stack); as a course we've made decisions around the project stack i.e. npm, node, TypeScript, Express server framework, Vercel deployment for students. However, if this project was to be built professionally, what would students recommend instead? Extension could be on the architecture to deploy this worldwide i.e. require CDN, potentially a regional load manager, etc.
- Project Management; reflection on current group's project management throughout term. Having the practical experience, if students could return to the start and act as the official manager for their team, what techniques based on project management theory would they choose? Describe the effect and justify suitability for this team, in comparison to a professional team.

- More advanced testing techniques e.g. mocking timers, mocking database connections, mock deployment, hypothesis testing, testing randomness, industry user acceptance testing
- More advanced CI/CD e.g. automated deployment through pipelines, containerisation of frontend and backend, additional software checks, exploring git pre-commit hooks
- Structure COMP1531 similar to the Algorithm course in which students can decide which tasks to do to achieve a certain mark. Also we can more specific about criteria in each iteration that is required to produce a certain grade.
- Consider introducing AI tools like Copilot in later courses for which students already learnt the fundamentals of programming (e.g., SENG workshops or COMP3900). Introducing the use of ai when students aren't 100% with their fundamentals could also be a detriment to their learning. as it may lead to an over reliance on ai to complete work. If introduced too early, research supports AI severely hindering students developing their own problem solving skills.
- AI is suggested as an extremely useful tool that is heavily used by engineers in industry whenever they can. Maybe it can be introduced in a course where its more open ended and students are unlikely to have similar solutions regardless of AI generating it. COMP3900 or SENG workshops come to mind when they are completing different types of projects from other students.
- Working on a real practical project earlier to make them job ready and help students secure internships more easily. Practical projects:
- Develop technical skills (an important baseline)
- Develop their ability to understand project management theory
- Develop their ability to understand interconnection
- Develop teamwork skills, conflict management skills, leadership skills
- Improve their maturity

This is because most jobs care about maturity, communication, and ability to work in a team. Technical roles care about baseline technical skills and ability to work in a technical team. By understanding the gaps in the technical field, students can be inspired to research certain areas.

- Teamwork is extremely valuable and should be maintained.

Theme 3: Software Design & HCI

- System design (where multiple codebases are used) can be better taught in courses that will have multiple codebases, like capstone type courses.
- Update COMP2511 course name to "Software Engineering Design."
- Students strongly oppose replacing Figma with Axure due to the lack of popularity in the industry and the limited value of Axure programming.
- Address redundancy in Figma learning between SENG2021 and DESN2000. Students are expected to learn Figma on their own in SENG2021, but it is then taught again in DESN2000.

Theme 4: Interesting Specialization

- There is a need for more lecturers with expertise in microservices and cloud architecture.
- Formalize documentation for SENG2021 and SENG3011 to manage knowledge transfer. SENG2021 and SENG3011 rely heavily on tutors, and their departure could create significant knowledge gaps. There is a need to formalize documentation for running these courses so the future course admins can manage the course easier. COMP2511 has a really good example of this that the admins try to maintain.
- Some courses are staffed only at the minimum level required to keep them running, which creates little to no incentive to document the processes involved. However, this is not the case for COMP2511.
- Unfortunately, this is a common issue at UNSW CSE. Many new LiCs and course admins at CSE are often thrown into the deep end, having to ask others for help or figure things out on their own. For example, they may struggle with setting up the exam environment, managing the SMS, or pulling enrolment data, because there is no formal documentation on this.
- Students may wrongly assume that pursuing a PhD leads to only an academic career. The low pay and high competition in academia can deter students from pursuing a PhD. While companies hire PhDs e.g. trading companies, Google, etc, this is a longer and more uncertain path to industry. To support student interest in research:

- Mentioning the current pertinent research in a field to help give them ideas
- More support from supervisors e.g. guidance on how to get started
- Many societies value COMP students, and collaborations with other disciplines, such as business, can be beneficial for both parties. Additionally, most societies look to build some website/specific technology for running their work. This could also provide practical experience opportunities for SE students. Something to note about Business societies:
- Business students only have ~3 required contact hours a term, and are more free to work on Case Competitions or other extracurriculars. These competitions usually benefit from some demo of working technology.
- ISTM students also have a coding capstone course, potentially possible to combine the courses so the Business school student works on the project management side and SE stays on the technical side
- The high number of casual lecturers can cause courses to drift.
- Some courses may need to be broken up. Courses with high dropout rates (42% by census) are likely candidates for updates.
- A problem is that students complete COMP2511 in an earlier stage in their degree progression, and as such, won't have skills needed to cover much harder design topics like system/cloud design.
- SENG tutorials could be reworked to address issues with students not finding lectures useful, but this raises concerns about knowledge gaps.

Appendix E: Industry Panel Review -

Minutes of Meeting 25 August

Present: Rob Pike; Yacine Rabhi; Nick Patrikeos; Chinmay Manchanda; George Wright;Alan Hsiao; Lawrence Yao; Andrew Gerrand

Apologies: Daniel Wirjo

Agenda

1. Panel members introduce themselves
2. Fethi provided a quick summary of the review so far
3. Feedback was sought from panel members for each of the themes below

Theme 1: Project Management

- Students do not need to be expert in PM but in some aspects of it. For example, students struggle with breaking work into small pieces. Also, starting with simple tasks instead of taking on too much. There is confusion regarding project management expectations, especially around project delivery timelines.
- PM is a large area with two components: planning using micro (Agile) and macro (managing multiple projects) approaches and tooling (planning, Microsoft and Atlasssian tools, devOps).
- Project management is often learned through the experience of doing projects, rather than formal training. There is a debate about whether to teach project management explicitly or let it be learned organically.
- Accreditation bodies often require evidence that project management is covered in academic programs, pressuring institutions to include it in the curriculum.
- In conclusion, it is not realistic to make SEs become project managers, but the teaching needs to make them able to work with program managers and understand their language. Also, they need to be able to perform some essential tasks like project planning and use appropriate tooling. At the same time, we need to be able to satisfy accreditation requirements.

Theme 2: Core SE Fundamentals

- Software Engineering (SE) fundamentals, such as problem-solving and system design, are important. The SE core course should focus on equipping students with foundational knowledge, integrating industry practices and research.
- There are discussions on how far to go into core SE skills and how to balance them with practice (doing a large scale project is time consuming). One suggestion is to make students work on an existing code base.
- Making students write test cases before coding is a good way to make them understand the whole process

- There were strong feelings about allowing students to use Co-pilot (most felt it was a bad idea).

Theme 3: Software Design & HCI

- Software design is highlighted as a crucial skill, requiring extensive practice and experience. But it takes a long time for someone to reach a point to become a software architect or designer. A design course needs to be modest in its objectives as most graduates will not be designing applications for a long time.
- The importance of good design principles is acknowledged, but it is noted that these can only be mastered through repeated application in real-world scenarios. Also one way to achieve this is giving students examples of well or poorly designed systems.
- Using queing theory and simulation tools is a good way to teach students how to test designs and make a design course more practical.
- Students have difficulties understanding the separation between Business Requirements and Design. Some effort has to be spent teaching them the difference. Students also have a tendency to retrofit requirements instead of doing investigations/asking stakeholders for information.
- Students need to see the different ways a system can be designed from the same set of requirements to understand the difference.
- Navigating a large code base and extracting designs from it is also a good way to teach design.
- Discussions also touched on the integration of Human-Computer Interaction (HCI) concepts within software design education.

Theme 4: Interesting Specializations / Gaps

- There's a challenge in balancing core SE content with electives, especially in areas that could become important specializations. There are gaps in the curriculum related to keeping up with evolving technologies and languages identified in previous reviews. Emphasis is placed on the need to address gaps in skills and knowledge, particularly in relation to the SE body of knowledge.
- In general, the idea of working on existing projects as part of the learning process is seen as an interesting approach which should be encouraged.
- Specializations like DevOps are considered complex but valuable areas for students to explore. Testing code automation and building deployment pipelines is important. However, learning DevOps in detail is not important, what was important was to work in a DevOps environment.
- Distributed system design is considered an important area. In particular, designing software components and APIs is important and in particular how to use them to build a business service in a “goal-oriented” design approach. Meeting business requirements using APIs in general is part of this.
- There is also a feel that network concepts are important as practitioners increasingly need to configure virtual networks.
- Given the rise of AI, data modelling in general is also important as a specialisation
- On the subject of business analysis, the most important thing for SE students is to understand the process and be able to bridge the gap between BA teams and dev teams.
- On the subject of HCI, this is a discipline that is not part of SE so again, SE students need to understand the interface with HCI.
- Staffing challenges are also noted as a contributing factor to existing gaps, affecting the ability to cover all necessary areas. Greater involvement from industry can help ensure a greater coverage.

Appendix F: COMP3142 structure

Course Summary

Software plays an important role in our daily life. It is important to construct robust, operational software, especially under limited development budgets and time constraints. To address this problem, a thorough verification and validation process is needed. In this course, we will study classic and modern techniques for the automated testing and analysis of software systems for reliability, security, and performance. Throughout the course, students will gain insight into a spectrum of software quality assurance techniques, including but not limited to fuzz testing and symbolic execution. These techniques will be not only studied but also applied in real-world scenarios, providing practical skills that are highly relevant in the ever-evolving landscape of software development.

Assumed Knowledge

You need to have successfully completed the core programming, algorithm, and software development courses.

The prerequisites of COMP3142 are COMP1531, COMP2511, and COMP2521.

Student Learning Outcomes

By the end of the course, students will be able to:

1. Understand the fundamental concepts and principles of software testing and quality assurance.
2. Identify and address common quality assurance challenges in software development.
3. Evaluate and select appropriate testing tools and frameworks.
4. Apply various software testing techniques to identify defects and ensure software reliability.
5. Analyze and interpret test results to make informed decisions.
6. Create effective automated test tools.

Appendix G: COMP613 structure

Course Details & Outcomes

Course Description

This course is designed to provide a systematic exploration of automated source code analysis and verification techniques, with the aim of gaining hands-on experience in implementing code analysis tools to identify common yet important software vulnerabilities in software systems. By taking this course, students can put static analysis and verification theories and advanced techniques into practice. They will be able to build source code analysis tools (e.g., written in C++) based on modern compilers and popular open-source frameworks to scan, comprehend and detect programming mistakes and vulnerabilities with the purpose of enhancing code quality and security.

Course Aims

The primary goal of this course is to familiarize students with a variety of source code analysis techniques and algorithms, ranging from fundamental to state-of-the-art. Students will be encouraged to develop their own tools based on an open-source framework that uses a compiler, and they will learn how these techniques and tools can be applied to detect real-world code vulnerabilities. Upon completing the course, students will have a comprehensive understanding of the history of source code analysis, the current techniques used, and the future challenges that must be addressed in this field.

Course Learning Outcomes

Course Learning Outcomes
CLO1 : Explain source code vulnerabilities in system software and motivations for software analysis and verification
CLO2 : Explain basic compiler intermediate representation and its importance for precise software vulnerability detection
CLO3 : Implement source code analysis techniques including control- and data-flow analysis
CLO4 : Develop source code verification techniques including constraint solving and assertion-based verification using automated theorem provers.
CLO5 : Design and implement well-considered, high performance, static analysis algorithm to solve problems
CLO6 : Create effective and minimum unit tests and documentation to validate the correctness of the code analysis tools