UNSW Engineering

Education Specification

1. Program Overview

Program Title: Bachelor of Engineering (Honours)

Award Title: Bachelor of Engineering (Honours) (Software Engineering)

Engineering Discipline: Software Engineering

Software Engineering aims to imbue students with an understanding that software system design is an engineering activity. They are designing and producing systems that must meet the standards expected of other engineering disciplines. Unfortunately, these standards are often not achieved in current software development practice.

2. Specialisation Framework

The SENGAH specialisation learning outcomes were developed by the SENGAH Director of Studies in conjunction with CSE's Deputy Head of School (education), and considered/refined by both the CSE Education Committee and the Industry Advisory Board. The suggestions from both the Committee and Board were incorporated into the final statement of the outcomes.

On how the students develop the specialisation learning outcomes through the program:

As noted in the Introduction, the core computing and mathematics courses give students a solid foundation on which to build subsequent study in the more advanced aspects of designing and implementing software systems.

After completing their core computing courses, Software Engineering students undertake a series of team-based workshops (DESN2000, SENG2011,SENG2021) which strengthen their skills in working on software development projects, both as a member and leader of the project team. In these workshops, and indeed in some earlier course, they make use of industry-standard platforms to help with the management of the project.

Outside the strict boundaries of software engineering, they also complete computing courses (COMP2041, COMP3311, COMP3331) which extend their skills in areas critical to today's IT profession (databases, networks, scripting). They also take courses (COMP3121 and COMP3141) that enhance their ability to do formal reasoning about the software systems they build.

SENG4920 Professional Issues and Ethics caps the technical content of the Software Engineering specialisation with development of workplace-relevant skills and attitudes.

The 4th-year thesis allows students to undertake a large-scale individual project, which allows them to employ the software engineering skills they developed over the previous three years in carrying out an end-to-end software development project: requirements analysis, design, implementation, testing.

On successful completion of this specialisation, graduates will be able to:

1. Demonstrate a solid understanding of the software engineering knowledge and skills, necessary to begin practice as a software engineer
2. Appropriately define and apply relevant abstractions from algorithmics, computer science, and mathematics to complex software system development
3. Design and build a system, component, or process to meet desired needs within realistic constraints such as technical, economic, security and ethical constraints
4. Think at multiple levels of detail and abstraction encompassing an appreciation for the structure of computer systems and the processes involved in their construction and analysis
5. Design software systems from the perspective of the end user and to communicate clearly and effectively with business stakeholders
6. Understand that software interacts with many different domains and the ability to be able to communicate with, and learn from, practitioners from different domains
7. Be knowledgeable about current and emerging software engineering practices in the workplace, collaborative software development and management processes and their role in the development of quality software systems processes and their role in the development of quality software systems.

3. Continuous Improvement

The Software Engineering specialisation was one of the earliest Software Engineering programs in Australia when it was introduced in the late-1990's, and was unique in having a sequence of worship courses where student could practice authentic software engineering tasks. Graduates have a variety of job prospects, being able to work in and manage teams anywhere that software development and maintenance takes place … which nowadays is everywhere.

The curriculum map shows that student's engineering capabilities are very strong (2.1-2.4). Similarly, their knowledge of core computing concepts is extensive (1.1, 1.2). Their soft skills (3.1 - 3.4) are adequate, but CSE could invest more time in developing these, especially in the workshop courses.

The Software Engineering specialisation has more scope than either Bioinformatics Engineer or Computer Engineering for students to take a range of technical electives. The two General Education courses and one Free Elective course give some scope for study outside the discipline. Students who want to study computing with some other discipline have two options: the 3-year Computer Science degree, or a dual award e.g Software Engineering/ Commerce).

While there is a strong Ethics component in fourth year, both accrediting bodies (EngAust and ACS) have requested that Ethics be covered more through the degree. CSE has recently hired a lecturer in Epistemics who delivered an ethics lecture in our first programming course in 22T1, will roll out ethics lectures in subsequent core courses, and will take over the teaching of the 4th-year Ethics course.

A substantial problem over the last two years is integrity of assessment, and especially final exams. Exams have been taken online, with no invigilation and open access to the Web. There is evidence of collusion between a small number of students (easily detectable with standard plagiarism checking techniques). A more insidious problem is the use of online "tutoring" sites during the exam. Many courses have aimed to mitigate the problem by downgrading the weight of the final exam in overall assessment. Alternatively, courses have used randomisation techniques (such as STACK questions, or choosing from a large pool of questions) to reduce the scope for collusion. The real solution to this problem is a return to on-campus, invigilated exams. CSE plans to return to this exam style as soon as feasible.

Note that plagiarism on take-home assignments has been an issue for some time. We make use of sophisticated plagiarism-checking systems (MOSS from Stanford) for checking copying in programming assignments. More recently, though, “contract cheating”, where students out-source the development of their assignment code to a third-party , has become an issue, and is much harder to detect. Some research on this has been conducted at Deakin University and we plan to extend their work to attempt to better detect such cheating.

Enrolments in the Software Engineering specialisation are relatively large (some 150 per year, but still much smaller than the Computer Science enrolments) and are growing

4. Review Process

UNSW’s Academic Offering Review and Monitoring Procedure outlines a structured approach to maintaining the quality and relevance of academic programs and courses. It includes both program-level and course-level review processes, with defined responsibilities and timelines.

Program Monitoring is conducted annually for all programs and specialisations. A comprehensive program review must occur at least once every five years for accredited programs, and every seven years for others. These reviews include a self-evaluation report (SER), review panel, review event, and a formal response with an implementation plan. Oversight is provided by the Academic Board and University Academic Quality Committee (UAQC), with input from Faculty Education Committees and Deans.

Course Review within UNSW Engineering is managed through a two-tiered process: Routine Course Review and Comprehensive Course Review. Routine reviews are conducted at the end of each term by Schools, using data such as enrolment, assessment outcomes, academic integrity issues, WAM differences, and student feedback (myExperience). Courses flagged through this process are added to the Comprehensive Course Review roster.

Comprehensive Course Reviews are detailed evaluations led by the Course Convenor in collaboration with a Faculty Educational Developer, Nexus Fellow, or Senior Academic. These reviews assess course design, pedagogy, alignment with learning outcomes, and feedback mechanisms. Outcomes are documented in a Course Development Plan and an Evaluation Report following the next course delivery. Schools must review at least 10% of their courses annually.

Stakeholder involvement spans multiple levels, including the Academic Board, UAQC, Faculty and School committees, Course Convenors, and external contributors such as students and professional bodies.

Frequency of updates includes termly course reviews, annual program monitoring, and five-yearly comprehensive reviews for accredited programs.

5. Curriculum Mapping

Table 1. Mapping of the specialisation learning outcomes to the Engineers Australia Stage 1 Competencies

SLO/PLO	PLO1	PLO2	PLO3	PLO4	PLO5	PLO6	PLO7	PLO8	PLO9	PLO10	PLO11	PLO12	PLO13	PLO14	PLO15	PLO16
1. Demonstrate a solid understanding of the software engineering knowledge and skills, necessary to begin practice as a software engineer	X		x		x			x			x					
2. Appropriately define and apply relevant abstractions from algorithmics, computer science, and mathematics to complex software system development		x	x				x	x								
3. Design and build a system, component, or process to meet desired needs within realistic constraints such as technical, economic, security and ethical constraints																
4. Think at multiple levels of detail and abstraction encompassing an appreciation for the structure of computer systems and the processes involved in their construction and analysis					x	x	x		x		x		x			
5. Design software systems from the perspective of the end user and to communicate clearly and effectively with business stakeholders					x				x			x				x
6. Understand that software interacts with many different domains and the ability to be able to communicate with, and learn from, practitioners from different domains					x	x						x		x		x
7. Be knowledgeable about current and emerging software engineering practices in the workplace, collaborative software development and management processes and their role in the development of quality software systems			x	x			x	x		x					x	x

Table 2. Mapping of courses to the specialisation learning outcomes

Course code	1. Demonstrate a solid understanding of the software engineering knowledge and skills, necessary to begin practice as a software engineer	2. Appropriately define and apply relevant abstractions from algorithmics, computer science, and mathematics to complex software system development	3. Design and build a system, component, or process to meet desired needs within realistic constraints such as technical, economic, security and ethical constraints	4. Think at multiple levels of detail and abstraction encompassing an appreciation for the structure of computer systems and the processes involved in their construction and analysis	5. Design software systems from the perspective of the end user and to communicate clearly and effectively with business stakeholders	6. Understand that software interacts with many different domains and the ability to be able to communicate with, and learn from, practitioners from different domains	7. Be knowledgeable about current and emerging software engineering practices in the workplace, collaborative software development and management processes and their role in the development of quality software systems
Level 1 Core Courses							
COMP1511	Introduced	Introduced		Introduced			
COMP1521	Introduced		Introduced	Introduced			
COMP1531	Introduced			Introduced	Introduced		Introduced
DESN1000	Introduced	Introduced	Introduced		Introduced	Introduced	Developed
MATH1081		Introduced				Introduced	Proficient
MATH1131		Introduced				Introduced	Proficient
MATH1141		Introduced				Introduced	

MATH1231		Introduced				Introduced	
MATH1241		Introduced				Introduced	
Level 2 Core Courses							
COMP2041	Introduced	Introduced	Introduced			Introduced	
COMP2521	Developed	Introduced	Introduced	Introduced			
COMP2511							
DESN2000	Developed	Developed			Developed	Developed	Developed
MATH2400		Introduced				Developed	
MATH2859		Introduced					
SENG2011		Developed					Introduced
SENG2021	Introduced	Developed	Developed	Developed	Developed	Introduced	Introduced
Level 3 Core Courses							
COMP3142	Developed	Developed	Developed	Proficient	Introduced	Developed	Developed
COMP3311	Introduced	Introduced		Introduced			
COMP3331	Introduced	Introduced	Introduced				
SENG3011	Developed	Developed	Proficient	Developed	Developed	Developed	Proficient
Level 4 Core Courses							
COMP4920				Developed		Developed	Developed
COMP4951	Proficient					Developed	Proficient
COMP4952	Proficient		Proficient				Proficient
COMP4953	Proficient		Proficient				Proficient
Discipline Electives							
COMP6131	Developed	Developed		Developed			Developed
ENGG2600	Introduced	Introduced				Introduced	Introduced
ENGG3060	Introduced		Introduced		Introduced		
ENGG3600	Developed	Developed				Developed	Developed
ENGG4600	Proficient					Proficient	Proficient
COMP3901							

COMP3902							
COMP4141							
COMP6752							
COMP9517							
ELEC4614							
TELE3113							
TELE4653							
COMP3231							
COMP3151							
COMP3161							
COMP3431							
COMP3891							
COMP3110							
COMP4418							
COMP4336							
COMP4601							
COMP4128							
COMP4951							
COMP4952							
COMP4953							
COMP4961							
COMP4962							
COMP4963							
COMP6721							
COMP6733							
COMP6448							
COMP6452							
COMP6713							

COMP9201							
COMP9332							
COMP9444							
COMP9447							
COMP9301							
COMP9302							
COMP9491							
COMP9727							
ELEC3104							
ELEC3105							
ELEC4601							
ELEC4604							
ELEC4613							
ELEC4621							
ELEC4622							
TELE4642							
COMP3121							
COMP3141							
COMP3211							
COMP3411							
COMP3421							
COMP3222							
COMP3601							
COMP3959							
COMP3445							
COMP4121							
COMP4920							
COMP4431							

COMP6714							
COMP6741							
COMP6447							
COMP6845							
COMP6445							
COMP6451							
COMP6080							
COMP6420							
COMP9242							
COMP9243							
COMP9333							
COMP9415							
COMP9417							
COMP9334							
COMP9313							
COMP9418							
COMP9312							
ELEC3117							
ELEC4122							
ELEC4123							
ELEC4611							
ELEC4623							
ELEC4617							
ELEC4951							
ELEC4952							
ELEC4953							
COMP3131							
TELE4651							

COMP3311							
COMP3331							
COMP3153							
COMP3142							
COMP3444							
COMP3453							
COMP3443							
COMP4161							
COMP4337							
COMP6771							
COMP6843							
COMP6443							
COMP6991							
COMP6453							
COMP6131							
COMP9315							
COMP9331							
COMP3821							
COMP3900							
ELEC3106							
ELEC3114							
ELEC3115							
ELEC3146							
ELEC3145							
ELEC3705							
ELEC4602							
ELEC4603							
ELEC4612							

ELEC4631							
ELEC4633							
ELEC4605							
TELE3118							
TELE4652							
COMP6441							
COMP9311							
COMP9321							
COMP9319							
COMP4511							
ELEC3111							
ELEC4632							
ELEC4445							
COMP3511							
COMP6841							
COMP9511							

The above table shows that coverage of the specialisation learning outcomes is spread fairly evenly across the four years of the degree. One area that is possibly a little weak, although with coverage across the degree, is the area of designing software from the perspective of the end user; this could be rectified by the inclusion of the HCI course (COMP3511) in the core of the specialisation.

6. Assessments

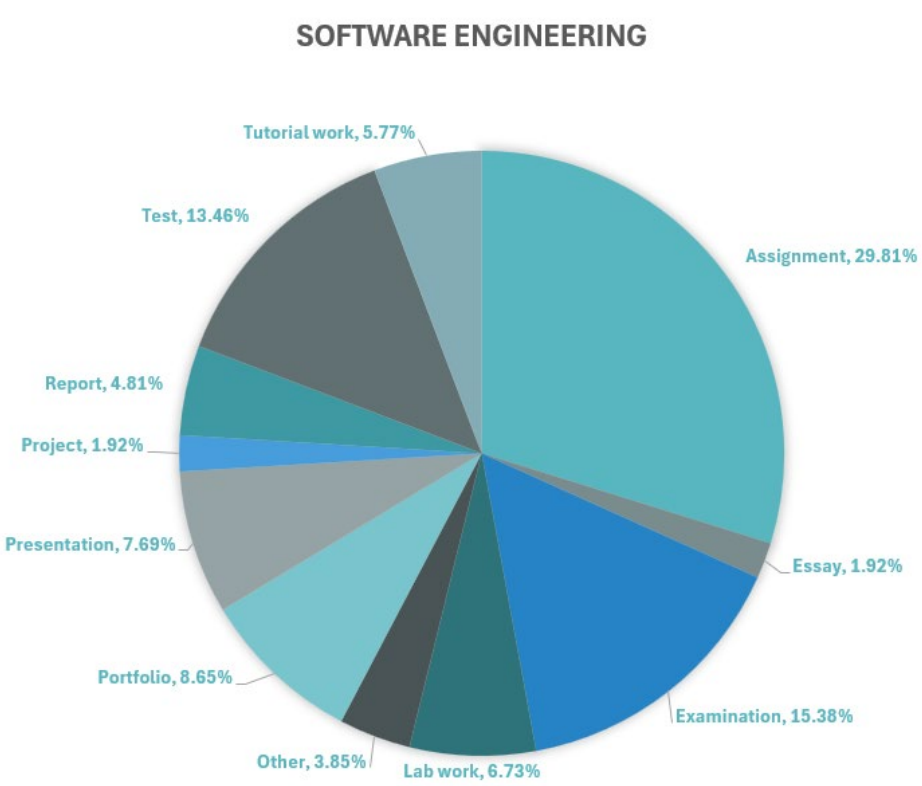

Figure 1. Percentage of assessment types used within the specialisation.

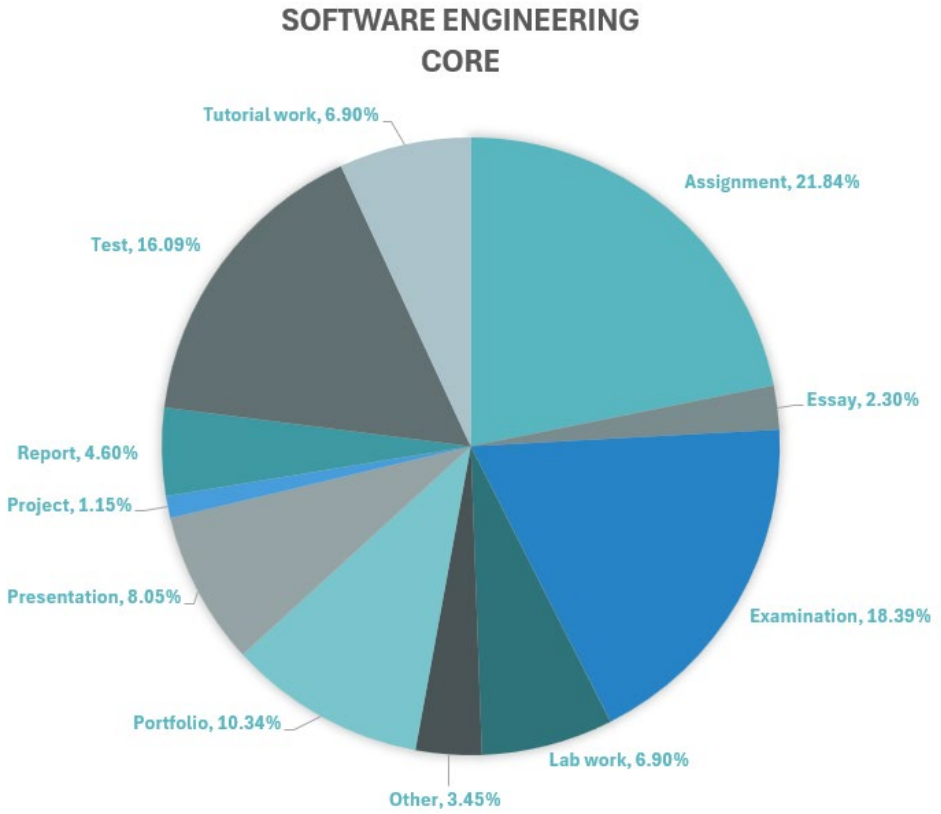

Figure 2. Percentage of assessment types used within the core of the specialisation.

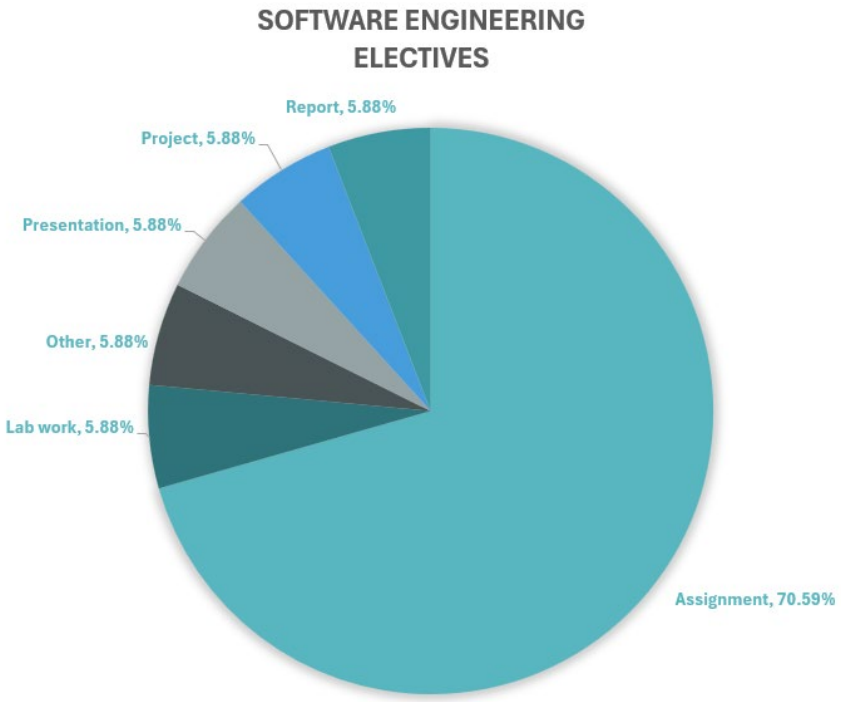

Figure 3. Percentage of assessment types used within the electives of the specialisation.

As can be seen in the table, there is heavy emphasis on assignment work and exams, to ensure that students have developed solid foundations. During the four years, there are also a number of team-based project work, to develop skills in solving larger problems in collaboration with other people. For example, DESN2000 and the SENG workshops are also team-based project courses, despite the classification of their assessments. While the 4thyear thesis is usually an individual project, it requires close interaction with the supervisor as a mentor, and presentations to the supervisor and their peers.

7. Specialisation Progression Plan

Students can track their progression through the “myPlan” checker tool.

myPlan | Current Students - UNSW Sydney

A progression checklist and/or study plan is also available for students for the single degree and the double degree offerings.

Progression checksheets & study plans | Engineering - UNSW Sydney