SE Degree Curriculum Review

Working Group Draft Report (Draft 3 November 2021)

Introduction

UNSW School of Computer Science and Engineering (CSE) has set up a working group whose purpose is to look at software engineering (SE) undergraduate degree and make recommendations for reviewing SE's existing structure and offerings in the light of several changes in the last 5 years. The working group should as much as possible make recommendations based on evidence from difference perspectives as well as constraints in the university/school.

The composition of the Working Group is:

- Chair: Fethi Rabhi
- CSE members: Jake Renzella, Nick Patrikeos
- External members: George Joukhadar (SISTM UNSW), Sherry Xu (Data61)

A number of sub-groups were created to look at particular teaching areas.

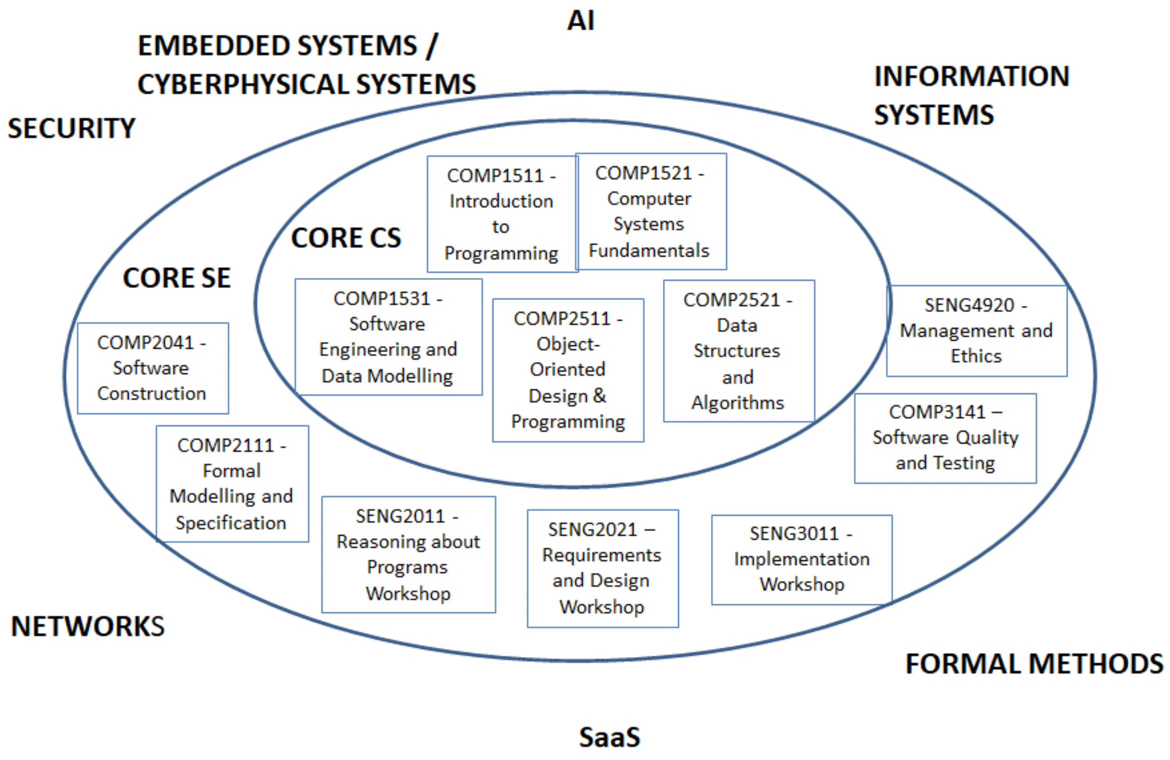
Process followed

The process was similar to the process in previous review, get feedback from different perspectives (a code is associated with each perspective):

- Different working groups considering different knowledge areas and the connection to the software engineering body of knowledge (SWEBOK)
    - [1] Review of formal methods (chaired by Carroll Morgan)
    - [2] Review of early software lifecycle (Chaired by Fethi Rabhi)
    - [3] Review of Software Design (Chaired by Nick Patrikeos)
    - [4] Review of Management and ethics Issues (Chaired by Wayne Wobke)
- [I] Industry consultation committee representing the views of prospective employers
- [S] Student representatives representing the student body
- [A] Academic staff and teaching committee

The learning objectives may be refined based on these recommendations.

As a reference, the structure of the degree at the last review in 2016 is illustrated below.


AI
EMBEDDED SYSTEMS / CYBERPHYSICAL SYSTEMS
INFORMATION SYSTEMS
SECURITY
CORE CS
CORE SE
COMP1511 - Introduction to Programming
COMP1521 - Computer Systems Fundamentals
COMP1531 - Software Engineering and Data Modelling
COMP2511 - Object-Oriented Design & Programming
COMP2521 - Data Structures and Algorithms
SENG4920 - Management and Ethics
COMP2041 - Software Construction
COMP3141 – Software Quality and Testing
COMP2111 - Formal Modelling and Specification
SENG2011 - Reasoning about Programs Workshop
SENG2021 – Requirements and Design Workshop
SENG3011 - Implementation Workshop
NETWORKS
FORMAL METHODS
SaaS

Specializations recommended in 2016 are listed in Appendix C.

Summary of Sub-groups recommendations

We summarise the recommendations made by these sub-groups and link them to the Software Engineering Body of Knowledge areas (table of content listed as Appendix B at the end of this document) and which are available for download at:

https://www.computer.org/education/bodies-of-knowledge/software-engineering

Formal Methods

A summary of the motivations and recommendations:

- Motivation: the removal of a core course (COMP2111) and changes in staff in other courses has made content between different formal methods courses inconsistent and their connection with the rest of the degree unclear
- Most salient proposition: making an existing course (COMP6721) a core course in the second year, referred to as X6721 in the rest of this document, whose purpose is to teach informal but rigorous reasoning and act as a bridge between formal methods courses and the rest of the degree
- Advantages: keeping formal methods as a valuable part of the degree, something that distinguishes us from other degrees
- Disadvantages: The introduction of a new core course limits the number of electives students can take

Specific notes/recommendations as well as connections to SWEBOK are included in the table below:

Course Name	Recommendations	SWEBOK Sub-Areas
New course (referred to as X6721), based on existing COMP6721	[1] First course on Formal-methods based programming techniques that will become core for the software engineering degree. [1] Not “full scale” use of programming logic or similar, but rather a focus on informal but rigorous reasoning. Explicit teaching of propositional- and predicate calculus. Informal rigour for imperative code; informal rigour for data-structure abstraction;	1 Software requirements -Requirements process - Requirements elicitation - Requirements analysis - Requirements validation
SENG2011	[1] Focus on automated reasoning using Dafny as well as introducing project-oriented tools. [1] showing how to automate some of the informal reasoning introduced in X6721 [1] Will have X6721 as a prerequisite	3. Software Construction 3.1 Software Construction Fundamentals 3.1.3 Constructing for Verification 3.3.2 Construction Languages 3.3.3 Coding 4. Software Testing 4.3 Test Techniques 4.3.3 Code-based Techniques 4.3.4 Model-based Testing Techniques 9.Software Engineering Models and Methods 9.1 Modelling 9.3 Analysis of Models 9.3.3 Analysing for Correctness 9.4 Software Engineering Methods 9.4.2 Formal Methods 14 Mathematical Foundations 14.1 Set, Relations, Functions 14.2 Basic Logic 14.3 Proof Techniques
COMP2111	[1] Introduce the basic and general theoretical formalisation concepts needed as in introduction to more	9 Software Engineering Models and Methods -Modeling -Types of Models -Analysis of Models

	specific and focused CS theory in later years	
COMP1511/COMP2521	[1] Integrate a small amount of X6721-style content into COMP1511 and COMP2521 in a way that would make some students better programmers, and the others no worse	13. Computing Foundations -Programming Fundamentals -Programming Language Basics -Data Structure and Representation
New (for 1st year)	[1] As an alternative to the above, introduction of an extra core course to relieve some of the pressure on COMP1511 and COMP2521 Data structures and algorithms", which could then allow better integration between informal methods and first-year students	

Early Phases in Software Cycle

A summary of the motivations and recommendations:

- Motivation: the introduction of a new Faculty course (DESN2000) has created the need to include "design material" that is specific to SE yet needs to connect with a more generic part provided by Faculty around "user-centred" design
- Most salient proposition: making DESN2000 focus on User-Centred product design, Requirements Engineering (RE) and early stages in the software lifecycle
- Advantages: takes pressure off teaching RE in other courses, gives all students a basic introduction to product and UI design
- Disadvantages: teaching such a course requires good tools and resources

Specific notes/recommendations as well as connection to SWEBOK are included in the table below:

Course Name	Recommendations	SWEBOK Sub-Areas
DESN2000 (new course introduced in 2020)	[2] Keep main focus of the course on User-Centred design, design thinking and Requirements Engineering (RE) and early stages of software lifecycle. DESN2000 to become a course which teaches students to see software as a product.	1 Software requirements -Requirements process - Requirements elicitation - Requirements analysis - Requirements validation 2 Software Design -UI Design 7. Software Engineering Management -Review and Evaluation
	[2] Focus on "product management" – which user stories are most important and which features should be developed first –	

	planning sprints at a high level rather than specifics of a project implementation	
	[2] Include the use of a UI design tool like Figma	
	[2] Make COMP151 (Software Engineering Fundamentals) a pre-requisite	
SENG2021	[2] Play down Requirements Engineering, User stories and UI design as these will be part of DESN2000	2 Software Design -Software structure and architecture -Software design notation 13 Computing Foundations -Abstraction -Basic Concept of a System -Database Basics
	[2] Include introduction of a Project Management tool (e.g. Jira) and architecture design, sequence diagrams and data models	
	[4] Project management to be redistributed amongst the SE workshops and DESN2000	
	[3] Make students continue a project started by someone else	

Software Design and Programming in the Large

A summary of the motivations and recommendations:

- Motivation: most of existing teaching still focused on programming in the small with few courses looking at programming in the large. Changes in technologies and agile practices gives the need to look at high-level design approaches and techniques for developing complex software systems
- Most salient proposition: reinforce existing courses with additional material: COMP1531 (testing), COMP2511 (design techniques and APIs), SENG2021/SENG3011 (Project management, design notations, some DevOps)
- Advantages: better coverage of SWEBOK areas and connections between core courses
- Disadvantages: not all material can be accommodated in existing core courses, there are still gaps in Front-End Technologies, software quality, data persistence, DevOps, etc.

Specific notes/recommendations as well as connection to SWEBOK are included in the table below:

Course Name	Recommendations	SWEBOK Sub-Areas
COMP1531	[2] Should cover testing fundamentals	1 Software requirements -Fundamentals 2 Software Design -Fundamentals 3. Software Construction

		-Fundamentals 4. Software Testing -Fundamentals 8. Software Engineering Process -Software Process Definition -Software Lifecycles
COMP2511	[2] should cover testing at design level like API testing	2 Software Design -Key issues -Structure and Architecture -Strategies and Methods 4. Software Testing -Design evaluation techniques (not listed)
	[4] Project management to be redistributed amongst the SE workshops and DESN2000	
COMP6452	[Sherry] Some material on Software Architecture basics should be moved to SENG2021 and COMP2511 to better prepare students for this course	
No particular course identified but will affect COMP6080	[3] Need to teach web front-end programming earlier in the degree as core	3. Software Construction
COMP60080 + COMP4511	[3] Make COMP6080 a prerequisite to COMP4511 to remove duplicated content, allow COMP4511 to teach more in-depth web development	
SENG2021/ SENG3011	Switch focuses of courses – 2nd year workshop on implementation, 3rd year workshop on Requirements & Design (after students have completed DESN2000 + COMP2511)	
Implementation Workshop + additional unidentified course	[3] In area of DevOps, need to cover these topics -Multi-instance, lambdas -Advanced continuous integration and pipelines -Cloud distribution -Software maintenance -Managing dependencies	6. Software Config Management
Implementation Workshop	[3] Data persistence, state-based applications and simple databases	3. Software Construction
	[3] Use abstractions while understanding implementations, allow students to	

	become more familiar with using frameworks/abstractions	
	[3] Discuss using 3rd party integrations	
	[3] Discuss advanced testing techniques – volume, regression, smoke, fuzzing	4. Software Testing & Quality
	[3] Make students continue a project started by someone else	
Design Workshop	[3] Discuss Software Quality as a specific concept, including benchmarking and software performance	10. Software Quality
Design Workshop + some other unidentified course	[3] Discuss asynchronous + concurrent programming	2. Software Design
	[3] Discuss full-stack architecture design considerations	
	[3] Discuss non object-oriented software architecture	
	[3] Design building software to work at scale and scale as a design consideration	
Both workshops	[2] Include more Project Management	3. Software Construction -Managing Construction -Construction Technologies -Software Construction Tools 7. Software engineering management -Software Project Planning
New	[A] Need course/project SE for particular application areas like AI systems	
New	[A] Make SE students take INFS electives on Security. There is a new course from 2022 called INFS2701 Cyber Security Management and Governance and another fourth year (Honours) course called INFS4907 Managing Security and Ethics in Cyberspace. Normally, INFS2701 requires INFS1701 Networking and Security as pre-requisite but we could ask this to be waived for SE students.	

Management and Ethics

A summary of the motivations and recommendations:

- Motivation: the introduction of a new Faculty course (DESN3000) has created the need to include "strategic design innovation" material that is specific to SE yet needs to connect with more generic part provided by Faculty.

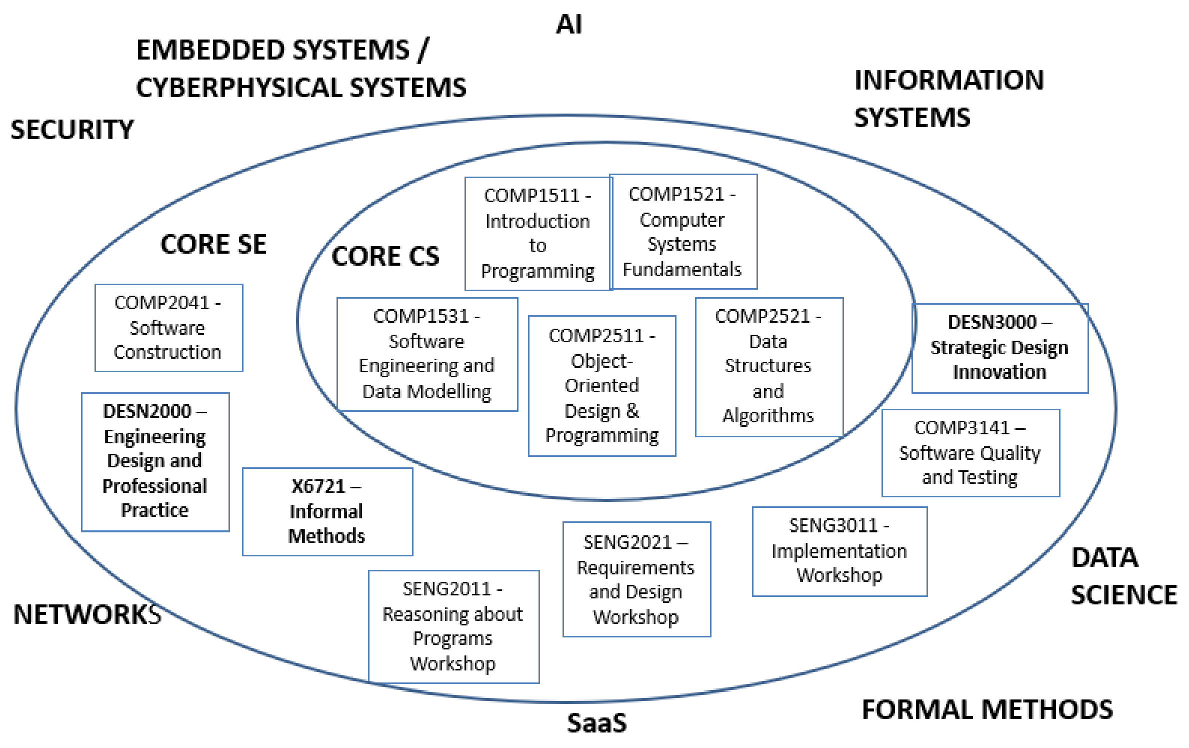
- Most salient proposition: replace an existing core course (SENG4920) by DESN3000. SENG4920 will drop project management to make room for more topics.
- Advantages: the change does not have a major impact, there is an opportunity to have a dedicated course on how to create a business plan and ethics considerations
- Disadvantages: there is a limit on how much can be taught in this course given the number of managements topics in the SWEBOK that are not currently covered.

Specific notes/recommendations as well as connection to SWEBOK are included in the table below:

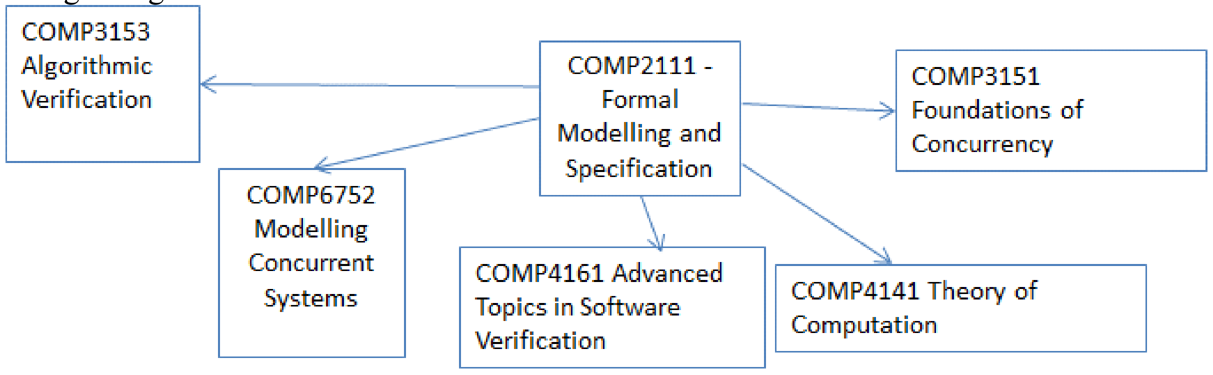
Course Name	Recommendations	SWEBOK Sub-Areas
DESN3000 (new course to be introduced in 2022)	[2] can be about lean canvas and creating business plans	11. Software Engineering Professional Practice 12. SE Economics
	[4] DESN3000 to replace SENG4920 as core to the SE programme	
	[4] Work with DesignNext to ensure sufficient coverage of ethics in a Software Engineering context to meet accreditation requirements	
	[4] Include ethical design considerations (such as data privacy, algorithmic decision-making bias, etc.)	
SENG2021/SENG3011 + potentially another course(s)	[4] General leadership theory – e.g. behaviour theory, team dynamics, risk management, decision making Software-specific management – theory of constraints, software processes	11. Software Engineering Professional Practice

New degree structure

As a result of the recommendations, the structure of the degree is as shown below (new courses shown in bold):


AI
EMBEDDED SYSTEMS / CYBERPHYSICAL SYSTEMS
INFORMATION SYSTEMS
SECURITY
CORE SE
CORE CS
COMP1511 - Introduction to Programming
COMP1521 - Computer Systems Fundamentals
COMP2041 - Software Construction
COMP1531 - Software Engineering and Data Modelling
COMP2511 - Object-Oriented Design & Programming
COMP2521 - Data Structures and Algorithms
DESN3000 – Strategic Design Innovation
DESN2000 – Engineering Design and Professional Practice
COMP3141 – Software Quality and Testing
X6721 – Informal Methods
SENG3011 - Implementation Workshop
SENG2021 – Requirements and Design Workshop
DATA SCIENCE
SENG2011 - Reasoning about Programs Workshop
NETWORKS
FORMAL METHODS
SaaS

Most specializations will have only minor changes except for Formal Methods where the changes might be more substantial:


COMP3153 Algorithmic Verification
COMP2111 - Formal Modelling and Specification
COMP3151 Foundations of Concurrency
COMP6752 Modelling Concurrent Systems
COMP4161 Advanced Topics in Software Verification
COMP4141 Theory of Computation

Revisiting Learning Objectives

Changes in indicated in yellow, mostly as a result of the consultation with industry.

SLO 1: demonstrate a solid understanding of the software engineering knowledge and skills, necessary to begin practice as a software engineer

SLO 2: ability to appropriately define and apply relevant abstractions from algorithmics, computer science, and mathematics to complex software system development

SLO 3: ability to design and build a system, component, or process to meet desired needs within realistic constraints such as technical, economic, security and ethical constraints

SLO 4: ability to 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

SLO 5: ability to think and design secure software systems from the perspective of the end user and to communicate clearly and effectively with business stakeholders

SLO 6: have the understanding that software interacts with many different domains and the ability to be able to communicate with, and learn from, practitioners from different domains

SLO 7: be knowledgeable about current software engineering practices in the workplace, collaborative software development and management processes and their role in the development of quality software systems

Students Point of View

The Working Group engaged with representatives of the student body to gather the student point of view on the working review documents. The following items were summarised from the meeting. Throughout this section of the degree review, the term "students" refers to the student representatives present in the Working Group.

Increased support for Industrial Placements

Students indicate increased support for industrial placements is required, citing the following points:

- Students would like to see more assistance from CSE/UNSW in help finding technical internships.
- Students point favourably to the program that University of Technology Sydney implement, which secures placements for each student.
- Students and the Working Group would like to see renewed efforts to better utilise and support the existing CSESOC Jobs Board.

Comments on Unit Design

- SENG students indicate that SENG2021 and SENG3011 are too similar in content and has potential to deliver more impact following changes to better separate the courses.
- Informal Methods (X6721) – Currently, SENG students are not able to enrol as indicated in the 2022 handbook
- Students feel that 6080 should be made core

Lacking Front-End Web Development Exposure

Students feel as though there is a lack of exposure to web front-end development, and that web knowledge is then expected (but not taught) in later courses. Some suggestions to alleviate this include:

- Making COMP6080 core, as it is a highly rated unit
- Make available in T3 of first year and cite as a "recommended course" in the new course handbook
- Re-aligning Fundamentals of Software Engineering (COMP1531) with its original design of being web-based course may help expose students to web front-end concepts earlier in the degree to better prepare for later courses.

Exposure to "Real" projects

Students would like increased exposure to authentic or real software development practices and projects to better prepare for industry. Students indicate the following specific areas would provide value:

- Development Operations (DevOps): Concepts such as Cloud hosting, Docker, Docker Compose, Kubernetes, etc are lacking.

- A discussion regarding an open-source course was held as a viable option for exposing students to realistic software engineering and dev/ops.

Request for updated public course handbook on the web
Students would like to see a public web version of the course handbook along recommended course structures.

Industry Point of View
The minutes of the industry consultation meeting are provided in Appendix A.

Specialisations

Security	COMP6441/6841 Security Engineering and Cyber Security COMP6443/6843 Web Application Security and Testing COMP6445/6845 Digital Forensics COMP6447 System and Software Security Assessment COMP6449 Security Engineering Professional Practice
Networks	COMP3331 Computer Networks & Applications (Core) COMP3332 Network Routing & Switching COMP3334 Capacity Planning of Computer Systems and Networks COMP4337 Securing Wireless Networks COMP6733 Internet of Things Design Studio
Databases	COMP3311 Database Systems (Core) COMP9313 Big Data Management COMP9315 Database Systems Implementation COMP9318 Data Warehousing and Data Mining
Algorithms	COMP3121 Algorithms & Programming Techniques COMP4121 Advanced Algorithms COMP4128 Programming Challenges COMP6741 Algorithms for Intractable Problems
Operating Systems	COMP3231/3891 Operating Systems COMP9242 Advanced Operating Systems
Artificial Intelligence & Machine Learning	COMP3411 Artificial Intelligence COMP4418 Knowledge Representation and Reasoning COMP9444 Neural Networks and Deep Learning COMP9417 Machine Learning & Data Mining

	COMP9418 Advanced Topics in Statistical Machine Learning COMP9517 Computer Vision COMP9727 Recommender Systems
Formal Methods	X6721 (In)-Formal Methods: The Lost Art (Core) COMP2111 Formal Modelling and Specification COMP3153 Algorithmic Verification COMP3151 Foundations of Concurrency COMP6752 Modelling Concurrent Systems COMP4161 Advanced Topics in Software Verification COMP4141 Theory of Computation
Programming Languages	COMP3131 Programming Languages and Compilers COMP3161 Concepts of Programming Languages COMP6771 Advanced C++ Programming COMP6772 ???

Final recommendations

Curriculum-related recommendations

The recommendations are primarily addressing the following gaps that were identified during this review:

- User experience, human factors
- Rigorous (though informal) reasoning about program construction
- Requirements engineering and Business Analysis.
- Design Fundamentals and Software Architectures

The recommendations are divided into short and long term solutions. The recommended short-term solutions are:

1. 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
2. Making DESN2000 focus on User-Centred product design, Requirements Engineering (RE) and early stages in the software lifecycle and adjust SENG2021 workshop accordingly.
3. Re-brand COMP2511 to "Software Design and Architecture"
4. Adjust SENG2021/SENG3011 workshops according to Recommendations 2 and 3 by introducing more on project management, design notations, some DevOps
5. Evaluate the material in the core content to see if security issues are properly discussed throughout the development cycle
6. Replace the existing core course (SENG4920) by DESN3000.
7. Incorporate discussion of security, risk and ethics in DESN2000 and DESN3000 as part of Recommendation 6.
8. 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.
9. 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).

Longer-term recommendations include:

- 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
- ensuring that the following topics are addressed as part of the core SE degree:
  - Concurrency
  - Parallel and distributed computing
  - Cloud computing
  - Testing and Software Quality
  - Software performance, building software at scale, high integrity systems, performance modelling and analysis
  - Continuous Integration, DevOps, Effort estimation, release management, software product line development

  - Software systems analysis (e.g. qualitative analysis, simulation), model checking, metrics
  - Web Front-end Programming
  - Security of software systems

Other non-curriculum recommendations

Pending from 2016

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.

New recommendations

- 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.
- Students require increased support for finding Industrial Placements
- 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
- Students would benefit from having the same tools used across multiple courses (e.g. Github/Jira)
- Leveraging industry connections to strengthen some of the existing courses is highly recommended

APPENDIX A

SE Degree Curriculum Review

Industry Feedback Meeting Minutes (22 October 2021)

Attending Members

The meeting members are:

- Chair: Fethi Rabhi (CSE)
- UNSW Representatives:
    - Jake Renzella (CSE)
    - Nick Patrikeos (CSE)
    - George Joukhadar (SISTM)
- Industry Representatives:
    - Ali Dasdan, Atlassian
    - Ned Farhat, Sage Consulting
    - Willis Li, New Relic, Inc.
    - Dr Aarthi Natarajan, AWS
    - Rob Pike
    - Yacine Rabhi, Tyro
    - Ben Reed, San José State University
    - Ben Smillie, GitHub
    - George Wright, Nine

Self Introductions and presentation of degree

Each attending member introduced themselves as described their connection to SE:

- Jake: I am lecturer in the school of CSE and my research area is in computing for education
- Nick: I am a student and work casually in CSE teaching first and second year software engineering courses.
- George J.: From the school of information technology management and i'm here to manage the connection between our school and the software engineering degree.
- Ali: I lead engineering for Confluence within Atlassian. 20+ years in software, small and large companies. I was excited to talk to you because Ben R. and I actually designing something similar to cover the needs of industry, as well as Atlassian along the same topic.
- Ned: I've been in the software space for far too long and in the last decade or so we moved more into cyber and machine learning, so I've got a big bias on how machine learning is portrayed in the industry and what the students are learning.
- Willis: I work in the consulting team based in Sydney, with SAS and all cloud devops. Hopefully I can learn something from this session and also contribute.
- Aarthi: I've been in the software space well over 20 years of studying and studying out as a full stack developer working in the industry and worked as a casual academic in CSE, rolled out one of the core courses (COMP1531) in for

the software engineering degree. Now with AWS primarily responsible for delivering training to customers in the areas of architecting, machine learning and data analytics.

- Rob Pike: I worked at Bell labs computing science, research, for many years and then at Google where I've been responsible for a number of large software projects, including operating systems and compilers and distributed systems, anything that underpins Google's data centers. I've also done some teaching: operating systems at Princeton on sabbatical and in another sabbatical I did suffer engineering at Sydney uni. I have a fairly strong ideas about how we need to balance the getting students up to date with the newest technology, while also making sure they know the difference between performance of sorting algorithms. I've seen a lot of issues with students coming out of school nowadays not having the fundamentals but being very good at working in a language like javascript or Python or something, but in a way that violates the principles of computing, as I understand it, so it's an interesting topic.
- Yacine: I come from a company called Tyro which specialises in payment and banking products for businesses, so my role there is what we call delivery lead. I manage three or four teams of engineers across Sydney and China to deliver software end to end like lifecycle of development and deployment management on the cloud.
- Ben S.: I work for Github based in Melbourne.My current role is looking after the technical aspects of our partner network globally and moonlighting, as part of our global education team in the apac region. In terms of connecting with universities, our education program has for probably two and a half years. A couple years at salesforce before that, and my relationship with UNSW comes from a number of years looking after emerging technology for the office of ANZ Bank's CTO engaging from an industry perspective with hackathons and getting students up to speed with the kind of industry tools that we were using at the time in banking.
- George W.: My biases are towards data, particularly data web analytics and the use of web analytics, but I have been forced to go into the grey waters of online advertising and marketing platforms, so I have also some skin in that game. We take on a fair number of students, I have three in my team at the moment we love having fresh recruits so I’m passionate about education. I have similar feelings to Rob, students are coming out more and more professional and ready to sort of turn code out but I questioned them on some fundamentals and they scratch their head. The last time we did this, where I actually complained that they didn't know enough about restful architectures, now I’m complaining that they are actually too good at that and forgotten some fundamentals.
- Ben R: I'm a relatively new professor at San Jose State University, been there for three, this is my fourth year. Before that I was at Yahoo and IBM and Facebook and I work on distributed operating systems.
- Fethi: I joined UNSW in 2000 in the Business School because it used to run the software engineering with CSE. After it pulled out, I went to CSE 11 years ago so I've been basically at UNSW managing the software engineering degree from both the business and technical perspectives. I always introduce myself as a bridge builder between the university and industry and between different areas in computing. Software engineering is great in bringing together different technologies for a common purpose and I try to get that reflected in the teaching.

This was followed by Fethi making a short presentation of the SE degree and the material circulated prior to the meeting.

Curriculum-related discussion

The discussion was centered around a number of topics, each of which is summarized in a separate subsection.

Overarching issues

Overall vision of the degree

Rob Pike asked a question related to the strategy and the big picture behind the degree. He said software engineering means different things to different people. There are software engineers researchers who write papers about meta analysis and building software at large, some people consider it as a management discipline, how do you make software work reliably. Some consider it having a range of different programming abilities, combined with a set of programmers to build a robust safe product that comes in, on budget on time and safe. There are a lot of things in software engineering beyond programming, some of which are in the curriculum but many of them are not. It's not just a matter of dealing with formal methods and having specification languages and that kind of thing. Is the purpose of the degree to make people with good managerial skills for managing software projects, who could then work in industry or is the purpose trying to build an academic software engineering curriculum to help ?

Fethi Rabhi answered that the degree is about making people understand software engineering from different perspectives so the different courses will give them an understanding of the issues encountered when programming in the large as well as other management issues.It exposes students to the the diversity of opinions and then the specialization will push them into one or the other of the views that were stated. For example, a specialisation could be about making someone ready for research and another one ready for industry.

Rob Pike said there are many interesting SE methods but in practice, they have very little direct use for most of the time. As opposed to cybersecurity which he firmly believes is a fundamental thing about making secure data systems today, which falls through the cracks because the fundamentals aren't there. When setting up a curriculum, much more time should be invested in a sort of systemic approach across the entire curriculum about how to write safe robust software and what it means to have secure software than teaching formal methods tools.

Aarthi Natarajan said that in her experience teaching about the degree, one of the major reasons that COMP1531 was introduced was because students graduating were graduating without even understanding the process of source control. The problem is not the technology, it is just a means to address some customer needs. When trying to solve a domain problem, one has put on a customer hat on and think about the design phase, requirements phase. What are the key requirements functional and non-functional so the implementation would just come right at the end ?

Working with large software systems

Rob Pike said that the biggest gap he's seen when hiring students in Australia is although they are very, very smart capable programmers, they're always astonished at the gap between what that means versus working in a large team on a large code base. On the first day in the job, where they suddenly got you know 10 or 50 or 100 people that are working with that could be 100,000 1,000,010 million lines of code that they're suddenly part of. They have probably never read more than a few thousand lines of anyone else's software they've only just seen assignments and written their own. They've never worked in a large team they never coded to a standard that someone else has given them they've never had to find a bug and code they didn't write. I think it's important to at least be aware of that that's a gap that's going to be hard for a university curriculum to fill. What it really means to program in the large can't be done within a small computer science faculty, it needs to be done in a team of large people.

He added that one of the key pieces of being a good software engineer is understanding the software you write is part of a process. Code has to be maintained by people other than you down the road. Writing code that works, is understandable by a large team and can be adopted by other teams should be part of the essence of a software engineering program.

Fethi Rabhi agreed that this was a good point. One area that came up in earlier meeting and we were thinking of using Jake's experience in encouraging students to participate in open source projects.

Ben Smillie mentioned that GitHub has an open source grant program they run in terms of getting students participate in Open Source projects and in organizations leveraging Open Source. they can help the broader Open Source kind of engagement.

Jake Renzella mentioned that such engagement could be spread out over several semesters or trimesters.

SE curriculum scope and teaching methods

Ali Dasdan stated that it is not feasible to teach everything in a generic way, skills that allow students to do everything. So there have to be some objectives about what a student needs to know when they graduate. For some things, some awareness needs to be taught like the fact that students are going to work with lots of teams and they need source control, either using open source or otherwise. They just need to have some experience with it, within reasonable university constraints, because some things will be covered later. There are many examples like debugging, verification, Web design, documentation are other areas where teaching needs to develop an awareness. Students can pick/extend the skills on a need basis depending on their work situation.

Fethi Rabhi said that the degree has the ability to deal with different types of knowledge areas and skills in different ways. Some have to be delivered the traditional ways and others via workshops/projects. If we diversify in the way we deliver teaching, we may be able to address some of these challenges. Recently, we have had industry offering teaching material and lectures like AWS (Aarthi did it).

Aarthi Natarajan said that the students found that really valuable with many questions that were sent me through linkedin about cloud computing. In this case, it helped students understand the value proposition behind cloud computing and how it adds value to organizations and the business

Rob Pike said one could imagine a course or a lab where you go to the Open Source world or you, you have a list of maybe half a dozen large Open Source projects students can choose from. and the lab is to work with that software to fix a bug or at a feature, not necessarily push it back out, because it's fundamentally similar to working on a team, where the code isn't yours, to start with.

Fethi Rabhi said that because we have workshops and workshops is about giving them a project which could be anything, there is space and a way to do it. There are many challenges and obviously one of them is assessment.

George Wright mentioned that during his studies, he learned to design his own solutions, whereas in the real world it's designed by committee. There are many issues like how do you even interact with the Community, how do you take, how do you stand up for your design and how do you take constructive feedback. Perhaps this is not something that needs to be taught but the expectation is for people to mature into the role.

Elegant vs Cheap/easy coding

George Wright said students are surprised when he mentions elegant code. They know the idea of correct code or fast code, but elegant code is something that's a lot harder to quantify. It is something reading other people's code and seeing that it is expressed beautifully.

Ned Farhat believes the problem is that machines have so much memory and storage capacity, that awareness about performance/memory usage does not matter much.

George Wright stated that moving everything into the cloud and it's auto scaled etc., people don't even need to think about index design or resource management it's just all managed automatically. But there are implications e.g. cost/time implications. Although cloud has been such a blessing, it's also making programmers very lazy.

Jake Renzella agrees but stated that we can't make every graduate write elegant code and for now the focus is more on the output.

Cybersecurity

Place of security/cybersecurity in the core curriculum

Ned Farhat stressed the importance of cyber security which is much more than technical aspects like dealing with hacking and setting firewalls. It permeates across multiple courses that are going to overlap somehow. For example, from a programming perspective you need start thinking of the threats upfront, then work that into a design so it's secure by design before you even begin coding and then discover all the issues. That also can flow back into say requirements again, getting people early on to start considering, not just the happy scenario, but also what all the nasty scenarios. Embedding key security principles must be done from the very beginning.

Aarthi Natarajan also stated that in the space of software engineering, security plays an important role. Currently, it seems to be like a number of electives (mostly funded by CBA) but many things should be components in the core degree.

Ben Smillie said that his company made a massive investment in terms of supply chain software composition and analytics some of it in the areas of security, although not explicitly cybersecurity, in collaboration with partners like CBA. They have been introducing security into earliest stages software development and this is something should be considered in the curriculum at some point in time.

New ways of designing/implementing systems (with security in mind)

Aarthi Natarajan said a lot of elements were covered when reviewing the curriculum in 2016, but security was missing. It's not just about cyber security and talking about SQL injection but also how to build secure systems, right from the start. Building agile software with test driven modeling but with a focus on assessing security threats.

George Wright added that many new graduates want to work in cybersecurity and that's too much of the focus at the moment. In the past, students were trying to build systems collaboratively but now the environment is hostile so the question is how do you build robust systems inside a hostile environment. Many new graduates expect to be basically coding all day whereas the amount of code needed right now is getting smaller and smaller and becoming less meaningful. He really likes the idea of formal and informal methods but he doesn't know if they can help in this context.

Rob Pike agreed and said that a focus on security on its own encourages thinking about security at the cost of software engineering, whereas a focus on security as a part of software engineering makes secure system as much likely to happen. One can make very secure systems that are unusable. So with a focus on good design with security as a fundamental component, the result might not be quite as secure but things that are absolutely secure are worthless. Making compromises and getting those judgments decisions into the design of software is a critical piece of all of this and it's tricky.

Ned Farhat believes moving away from the tech side to more like threat modeling so actually thinking upfront about how a system could be attacked and abused and being able to balance design with security with functionality, with even budgets.

George Wright warned against overfeeding the degree otherwise the result will be a lot of very, very talented people who are good at risk assessments with a system implemented in an inefficient way.

Ned Farhat against stressed that he views security not just as a non-functional requirement but affects functional requirements as well.

Jake Renzella concluded by saying security is not a matter of just creating one or two security streams that are optional but it's about talking about it at the different levels and what it impacts how we're doing software engineering.

Specific topics

Formal methods

Ali Dasdan asked clarifications on exactly what was going to be taught in the new core formal methods course. Fethi Rabhi answered that it is not specifically about formal methods but teaches formal reasoning without the distraction of learning a formal notation, a bit like pseudo-code teaches programming concepts without being bogged down by a language's syntax. It's an articulation course between the informal and formal world and will make students better appreciate the role of formal methods and how they link to concepts they are most familiar with.

ML/AI/Ethics

Ned Farhat's opinion is that a large part of machine learning should be about the ethics side. George Wright goes one step further to say that a lot of the battles he is fighting in the advertising and web analytics space is around these Grey systems that sit in the middle and apply AI. What they apply is a bit of a mystery but cost you an extra 5% more and claim they can do better than humans. He has two students at the moment, who are data scientists and trying to solve everything with machine learning. Sometimes, the statistical method will run better, faster cheaper and it's provable.

Ned Farhat agreed that people just tend to think more complex is better, which means one important issue from the ethical side is that the explainability of these models. The complexity is getting ridiculous to the point where people are trusting black box magic because they've taken the human element, judgment, the emotion etc.

Ben R said one thing to note is when we talk about ethics often we talk about machine learning but the ethics problem has always been with us from the very beginning. We need to start thinking about ethics and everything related to security. Every time in his classes when he talks about networking or about management or even operating systems, he mentions ethics. Questions like what are the ethical considerations of locking down API, using protected boot loaders.

Rob Pike mentioned the problem they had with safety and Google self-driving cars with people driving on the highway alongside them taking their eyes off the road to take self-pictures.

George Wright agreed that there is definitely room for something about unintended consequences and the idea that we're all optimistic when designing. But in the real world, we've got to think pessimistically, and he doesn’t think we're going way into the realm of almost metaphysics and philosophy.

Ned Farhat concluded by saying maybe we need a philosophical subject because people have trouble articulating these sorts of issues to an executive, to get them to understand that. He estimates 50% of his job is communication to non-technical people, convincing a jury or judge or an executive.

Design concepts

Rob Pike noticed that CSE offers a course on object-oriented design and programming. He has nothing against object-oriented design and programming. But he thinks it's important to understand that it's fading from the way a lot of software gets written now even the languages like Java and c++ that have promulgated type heavy design are leading away more towards compositional design. There are other paradigms like concurrent programming parallel programming functional programming, aspect-oriented programming etc. He feels this is teaching an old way of thinking about how we write large software now.

Fethi Rabhi agreed and said one of the recommendations in this review is that this course is going to move from object-oriented design to just design. It will keep some focus on object-oriented design but will go into other types of design and design patterns, which are more general rules of you construct designs and what does good design means.

George Joukhadar said it is not clear where business analysis is covered in most of the courses' content as there seems to be more focus on design.

Fethi Rabhi answered that to some extent, that's what DESN2000 is going to go into more of the business analysis and requirements elicitation.

Non-curriculum related discussion

Fethi Rabhi gave an overview of various ways for industry to engage with universities. Even when engaging at the research project level, industry will indirectly contribute to increase the quality of teaching by helping forming tomorrow's new academic staff.

George Wright pointed out to some difficulties when engaging with Universities in research projects, particularly finding ways that we can get more in sync with each other, because that is hard to do.

Fethi Rabhi's experience is that in many cases, having direct engagements which cut the middleman works out better/cheaper that some administration-heavy research programs.

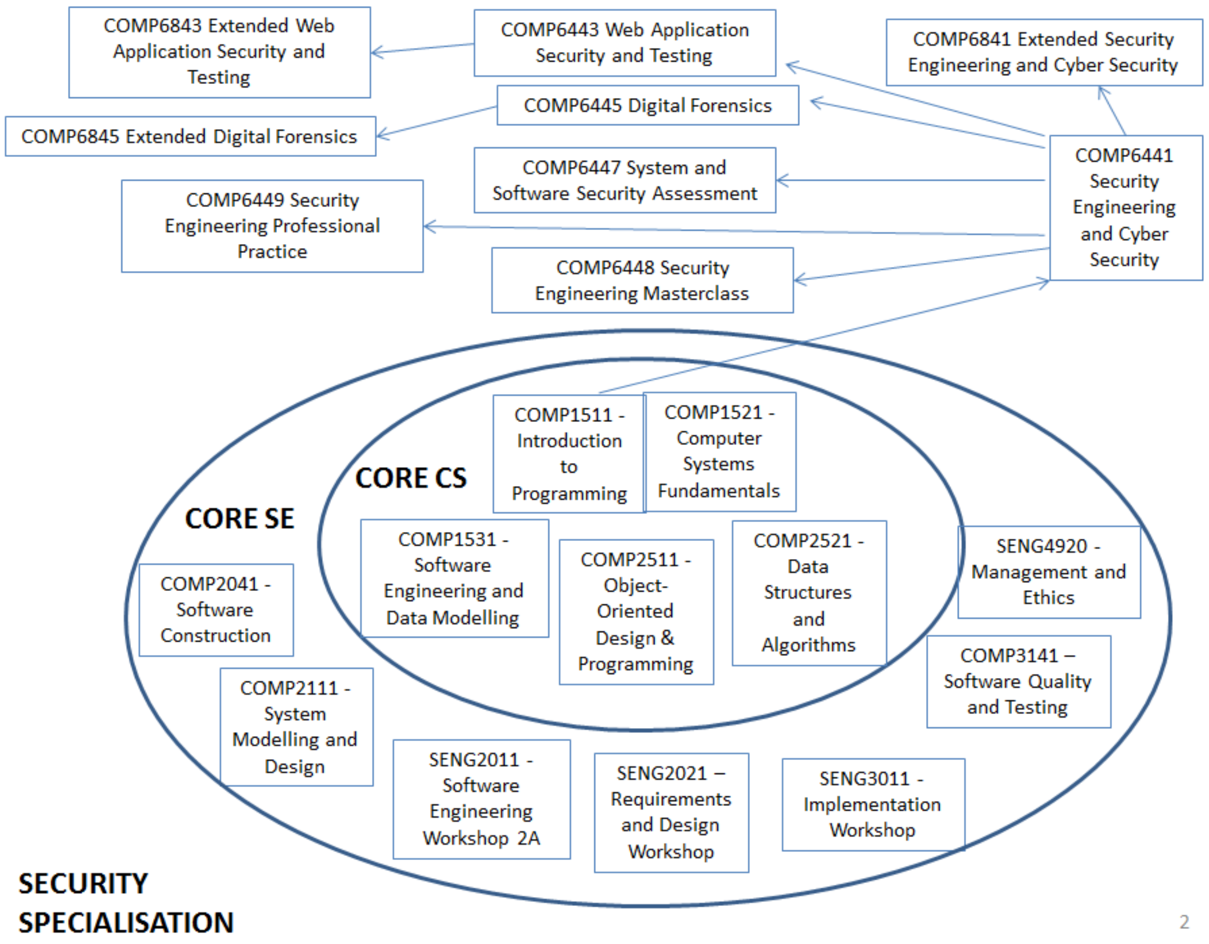
Ben Smillie indicated that GitHub was prepared to help engaging in a collaborative research grant potentially in the security domain.

APPENDIX B: SWEBOK Areas

APPENDIX C: Specialisations in 2016 review

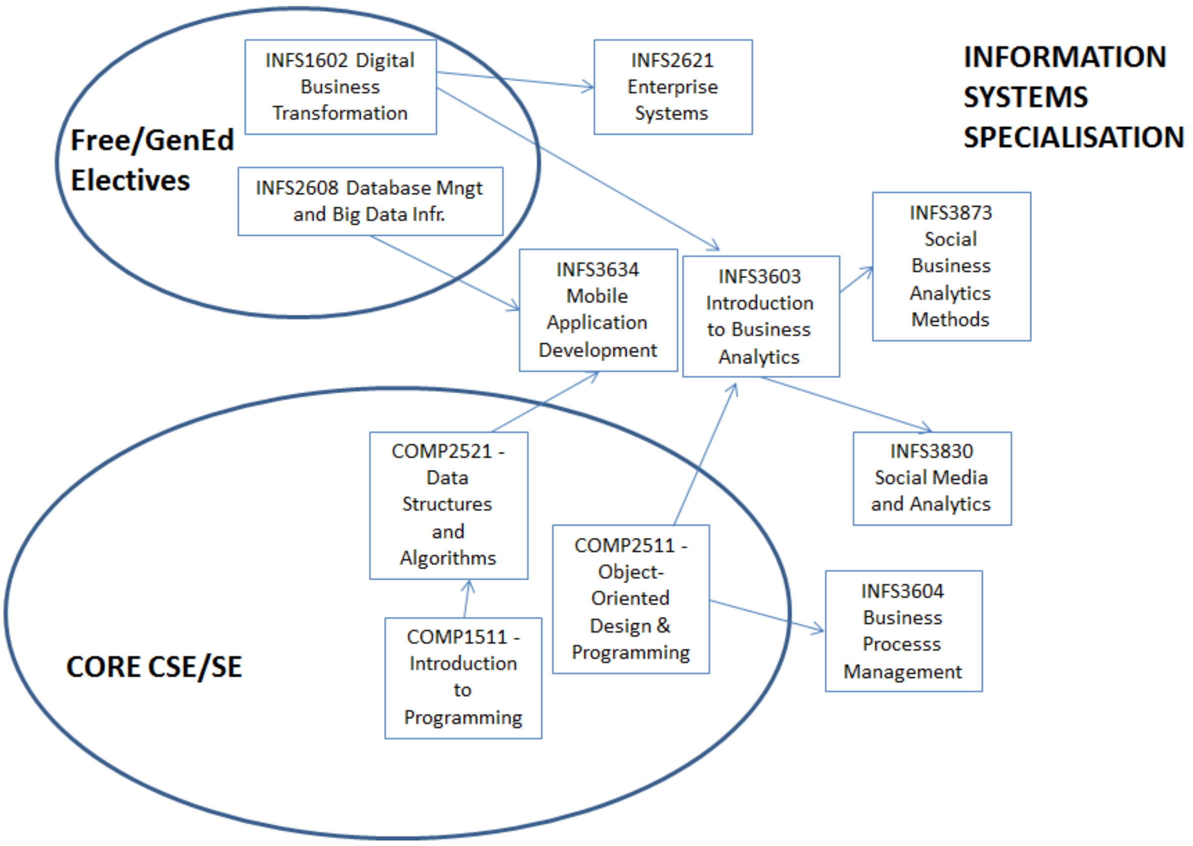
Security specialisation

Defined around new courses funded by CBA initiative.


COMP6843 Extended Web Application Security and Testing
COMP6443 Web Application Security and Testing
COMP6841 Extended Security Engineering and Cyber Security
COMP6445 Digital Forensics
COMP6845 Extended Digital Forensics
COMP6441 Security Engineering and Cyber Security
COMP6447 System and Software Security Assessment
COMP6449 Security Engineering Professional Practice
COMP6448 Security Engineering Masterclass
CORE CS
COMP1511 - Introduction to Programming
COMP1521 - Computer Systems Fundamentals
CORE SE
COMP1531 - Software Engineering and Data Modelling
COMP2511 - Object-Oriented Design & Programming
COMP2521 - Data Structures and Algorithms
SENG4920 - Management and Ethics
COMP2041 - Software Construction
COMP3141 – Software Quality and Testing
COMP2111 - System Modelling and Design
SENG2011 - Software Engineering Workshop 2A
SENG2021 – Requirements and Design Workshop
SENG3011 - Implementation Workshop
SECURITY SPECIALISATION
2

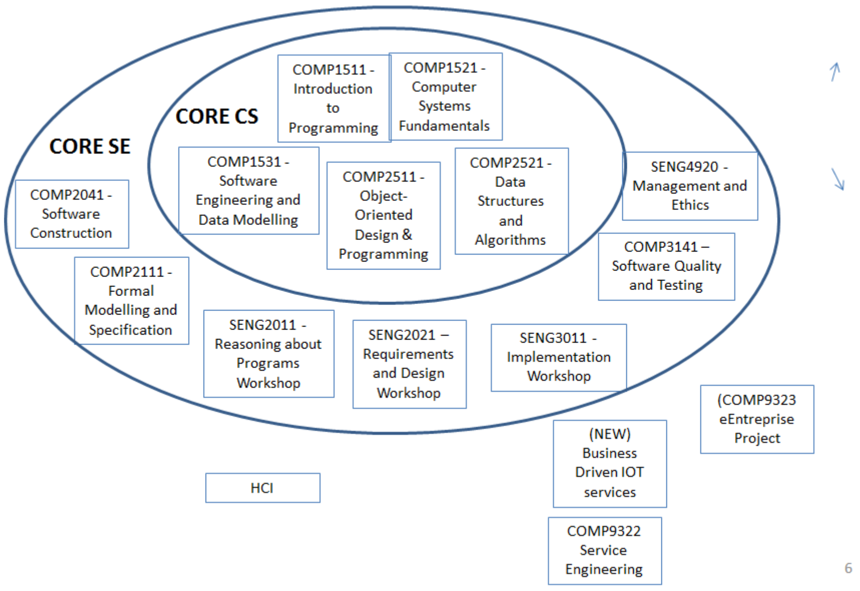
Information Systems Specialisation

Defined around courses offered by SISTM (www.sistm.unsw.edu.au)



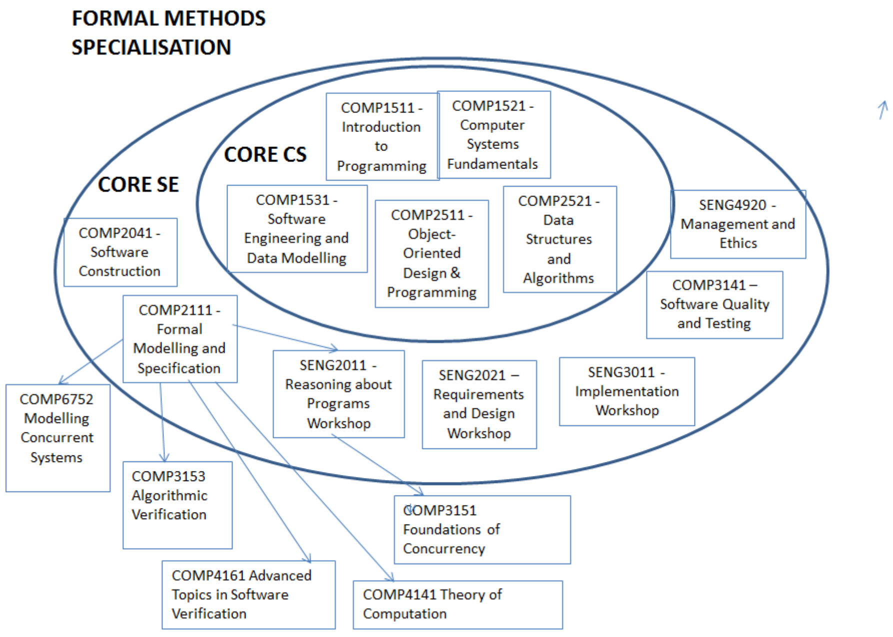
INFORMATION SYSTEMS SPECIALISATION
Free/GenEd Electives
INFS1602 Digital Business Transformation
INFS2608 Database Mngt and Big Data Infr.
INFS2621 Enterprise Systems
INFS3634 Mobile Application Development
INFS3603 Introduction to Business Analytics
INFS3873 Social Business Analytics Methods
INFS3830 Social Media and Analytics
INFS3604 Business Processs Management
CORE CSE/SE
COMP2521 - Data Structures and Algorithms
COMP1511 - Introduction to Programming
COMP2511 - Object-Oriented Design & Programming

Defined around the work of Service Oriented Computing Research Group
https://sites.google.com/site/unswsoc/



CORE SE
CORE CS
COMP1511 - Introduction to Programming
COMP1521 - Computer Systems Fundamentals
COMP1531 - Software Engineering and Data Modelling
COMP2511 - Object-Oriented Design & Programming
COMP2521 - Data Structures and Algorithms
COMP2041 - Software Construction
COMP2111 - Formal Modelling and Specification
SENG2011 - Reasoning about Programs Workshop
SENG2021 – Requirements and Design Workshop
SENG3011 - Implementation Workshop
SENG4920 - Management and Ethics
COMP3141 – Software Quality and Testing
(COMP9323 eEntreprise Project
(NEW) Business Driven IOT services
COMP9322 Service Engineering
HCI
6

Formal Methods Specialisation

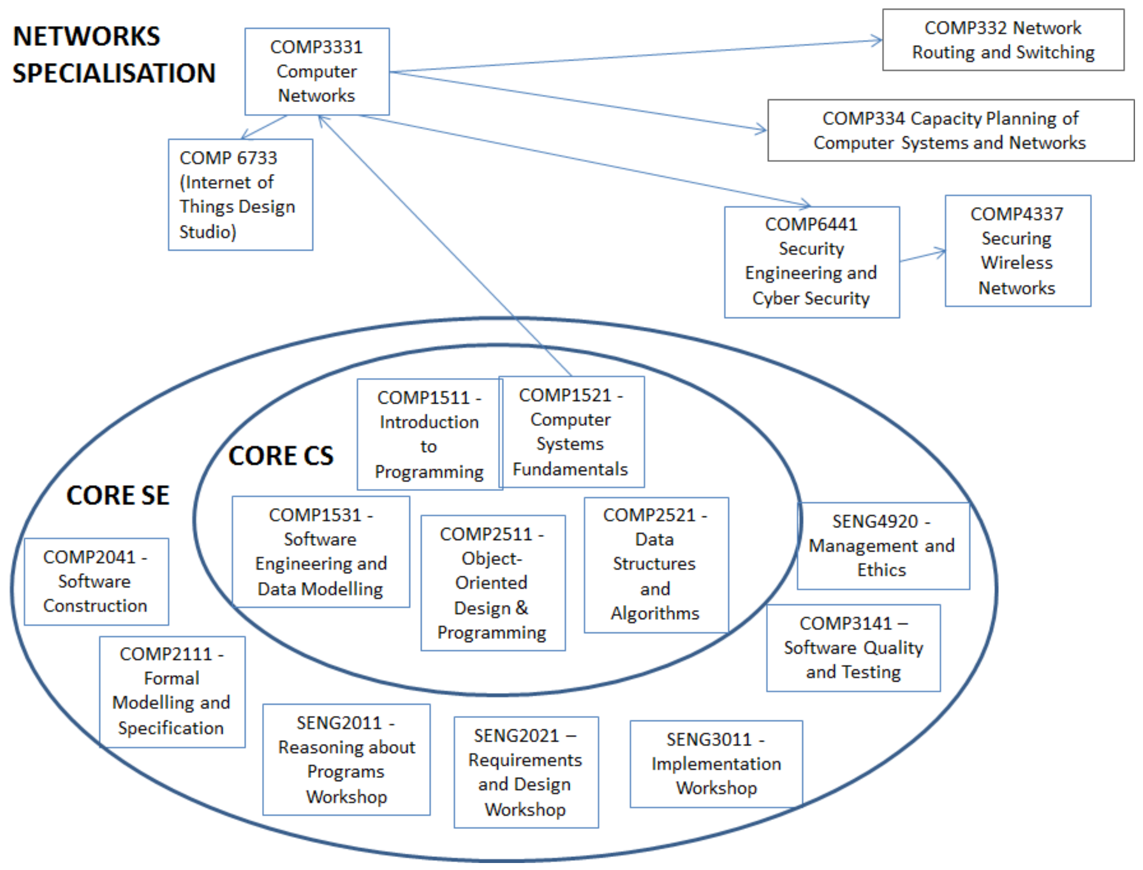

FORMAL METHODS SPECIALISATION
CORE SE
CORE CS
COMP1511 - Introduction to Programming
COMP1521 - Computer Systems Fundamentals
COMP1531 - Software Engineering and Data Modelling
COMP2511 - Object-Oriented Design & Programming
COMP2521 - Data Structures and Algorithms
SENG4920 - Management and Ethics
COMP3141 – Software Quality and Testing
COMP2041 - Software Construction
COMP2111 - Formal Modelling and Specification
SENG2011 - Reasoning about Programs Workshop
SENG2021 – Requirements and Design Workshop
SENG3011 - Implementation Workshop
COMP6752 Modelling Concurrent Systems
COMP3153 Algorithmic Verification
COMP3151 Foundations of Concurrency
COMP4161 Advanced Topics in Software Verification
COMP4141 Theory of Computation

Data science Specialisation (awaiting information)

Defined around the work of Databases Research Group
https://www.engineering.unsw.edu.au/computer-science-engineering/research/research-activities/database-research-group

Networks Specialisation

Defined around the work of Networked Systems and Security Research Group
https://www.engineering.unsw.edu.au/computer-science-engineering/research/research-activities/networked-systems-and-security-group-netsys


NETWORKS SPECIALISATION
COMP3331 Computer Networks
COMP332 Network Routing and Switching
COMP334 Capacity Planning of Computer Systems and Networks
COMP 6733 (Internet of Things Design Studio)
COMP6441 Security Engineering and Cyber Security
COMP4337 Securing Wireless Networks
CORE CS
CORE SE
COMP1511 - Introduction to Programming
COMP1521 - Computer Systems Fundamentals
COMP1531 - Software Engineering and Data Modelling
COMP2511 - Object-Oriented Design & Programming
COMP2521 - Data Structures and Algorithms
SENG4920 - Management and Ethics
COMP3141 – Software Quality and Testing
COMP2041 - Software Construction
COMP2111 - Formal Modelling and Specification
SENG2011 - Reasoning about Programs Workshop
SENG2021 – Requirements and Design Workshop
SENG3011 - Implementation Workshop

Embedded Systems Specialisation (awaiting information)

Defined around the work of Trustworthy (http://ts.data61.csiro.au/projects/TS/) and Embedded Systems Research Groups

AI Specialisation (awaiting information)

Defined around the work of AI Research Group