

SE Degree Curriculum Review

Working Group Draft Report (Draft 7 October 2021)

Introduction

The working group's purpose is to look at the area of software engineering and make recommendations to the School of Computer Science and Engineering (CSE) regarding reviewing SE's existing structure and offerings in the light of the changes to the core courses. The working group should as much as possible make recommendations based on evidence from different perspectives as well as constraints in the university/school.

The composition of the Working Group is:

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

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

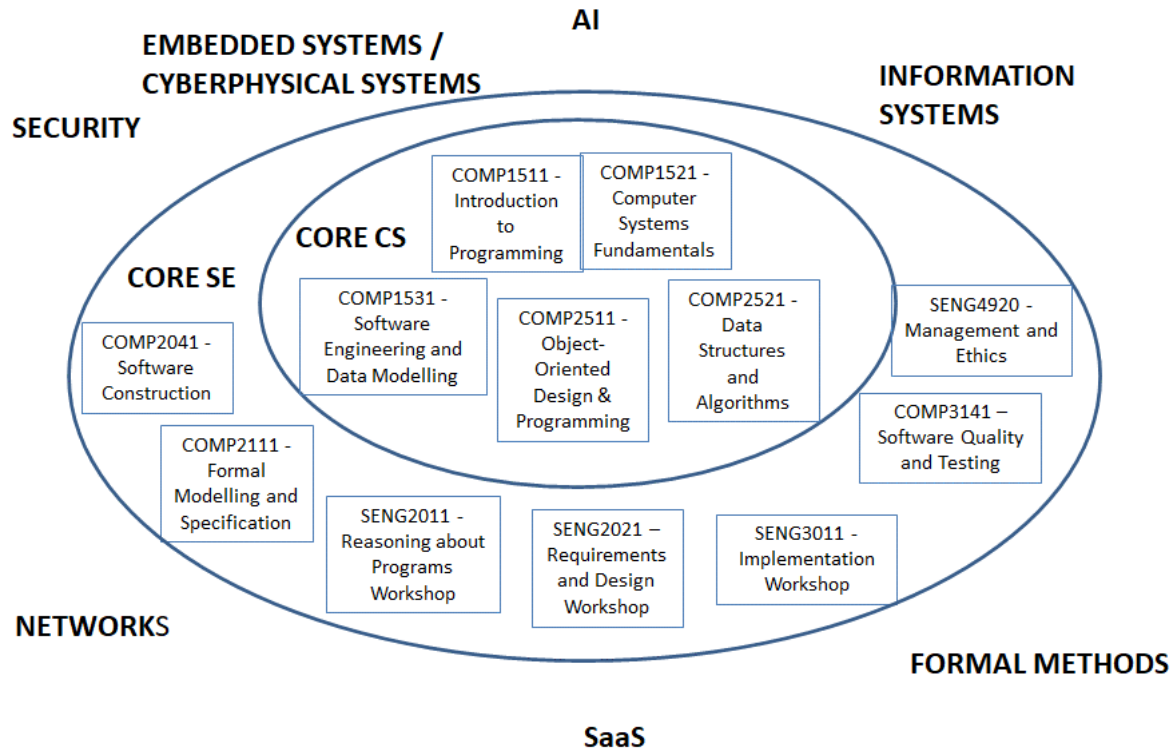
Process followed

The process was similar to the process in previous review, get feedback from different perspectives:

- Summarize the recommendations of the different working groups considering software engineering body of knowledge
- Redefining learning objectives: does existing structure supports these objectives ?
- Industry perspective: are we meeting the needs of industry ?
- Research priorities perspective: are we leveraging our research strengths ?
- Alumni/new students perspectives: are we meeting needs of existing students and aspirations of new ones ?
- New developments: are we in line with new developments in the field ?

A draft SE Degree handbook has been prepared, containing proposed degree structure assuming all changes have been carried out. The draft handbook is provided as an attachment to this report.

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



Specializations recommended in 2016 are listed in Appendix B.

Summary of Sub-groups recommendations

The recommendations are grouped according to Software Engineering Body of Knowledge areas (table of content listed as Appendix A at the end of this document) and which are available for download at [Software Engineering Body of Knowledge Version 3 | IEEE Computer Society](#)

Each recommendation is preceded by a number indicating the sub-group it originates from.

- [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)

Formal Methods

Course Name	Recommendations	SWEBOK Sub-Areas
New course (referred to as X6721), based on existing COMP6721	<p>[1] First course on Formal Methods Practice that will become core for the software engineering degree</p> <p>[1] Focus on informal but rigorous reasoning. Explicit teaching of propositional- and predicate calculus. Informal rigour for imperative code; informal rigour for data-structure</p>	<p>1 Software requirements</p> <ul style="list-style-type: none"> -Requirements process - Requirements elicitation - Requirements analysis - Requirements validation

	abstraction;	
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	9 Software Engineering Models and Methods -Modeling -Types of Models -Analysis of Models
COMP2111	[1] Introduce the basic and general theoretical formalisation concepts needed as in introduction to more specific and focused CS theory in later years	9 Software Engineering Models and Methods -Modeling -Types of Models -Analysis of Models
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 1 st 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

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	
No particular course identified		

Software Design and Programming in the Large

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

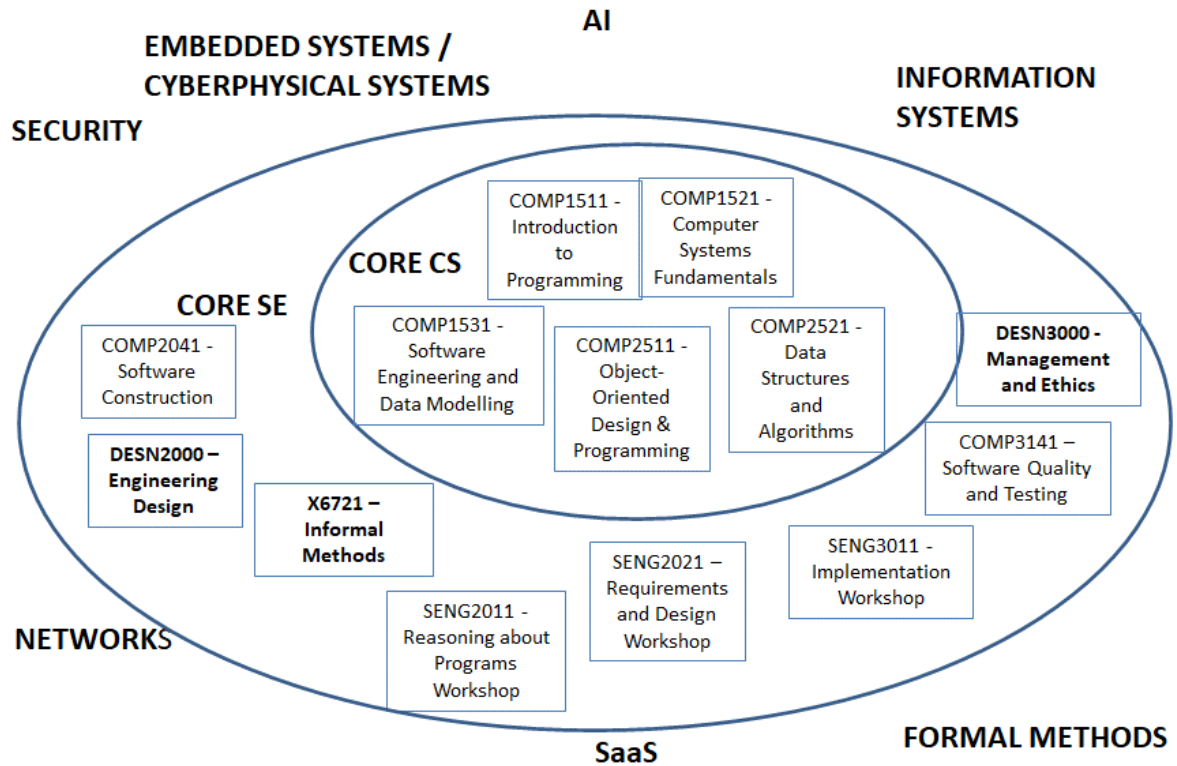
		-Structure and Architecture -Strategies and Methods 3. Software Testing -Design evaluation techniques (not listed)
SENG3011	[2] Include more Project Management	3. Software Construction -Managing Construction -Construction Technologies -Software Construction Tools 7. Software engineering management -Software Project Planning
	[2] Include the use of tools to support complete software development and deployment pipelines	
	[3] Needs more discussion on API integrations in software construction	
	[3] Make students continue a project started by someone else	
	[4] Project management to be redistributed amongst the SE workshops and DESN2000	
No particular course identified but will affect COMP6080 and COMP4511	[3] Need for a dedicated course that teaches Web-Front end programming from a technical perspective building on COMP1531	
No particular course identified	[3] Software quality and testing is not adequately covered. There is minimal coverage of benchmarking, software performance	
No particular course identified	[3] In area of DevOps, need to cover these topics -Multi-instance, lambdas -Advanced continuous integration and pipelines -Cloud distribution -Software maintenance -Managing dependencies	
	[3] In area of software design and construction, need to cover these topics: -Lack of “Good software design” practices:	

	<p>non object-oriented software architecture is not covered in depth (e.g. asynchronous programming and concurrency), more experience and interaction with frameworks, while maintaining understanding of implementations, scalability as a design consideration for software (includes use of microservices), Hyrum's Law as a principle/consideration</p> <p>-Lack of team-code building: reading and working with code written by others, working with a large codebase, writing code as part of a team or organisation</p> <p>-Not enough discussion of data persistence (trivially covered in COMP1531 & 2511) outside of pure database courses</p>	
--	--	--

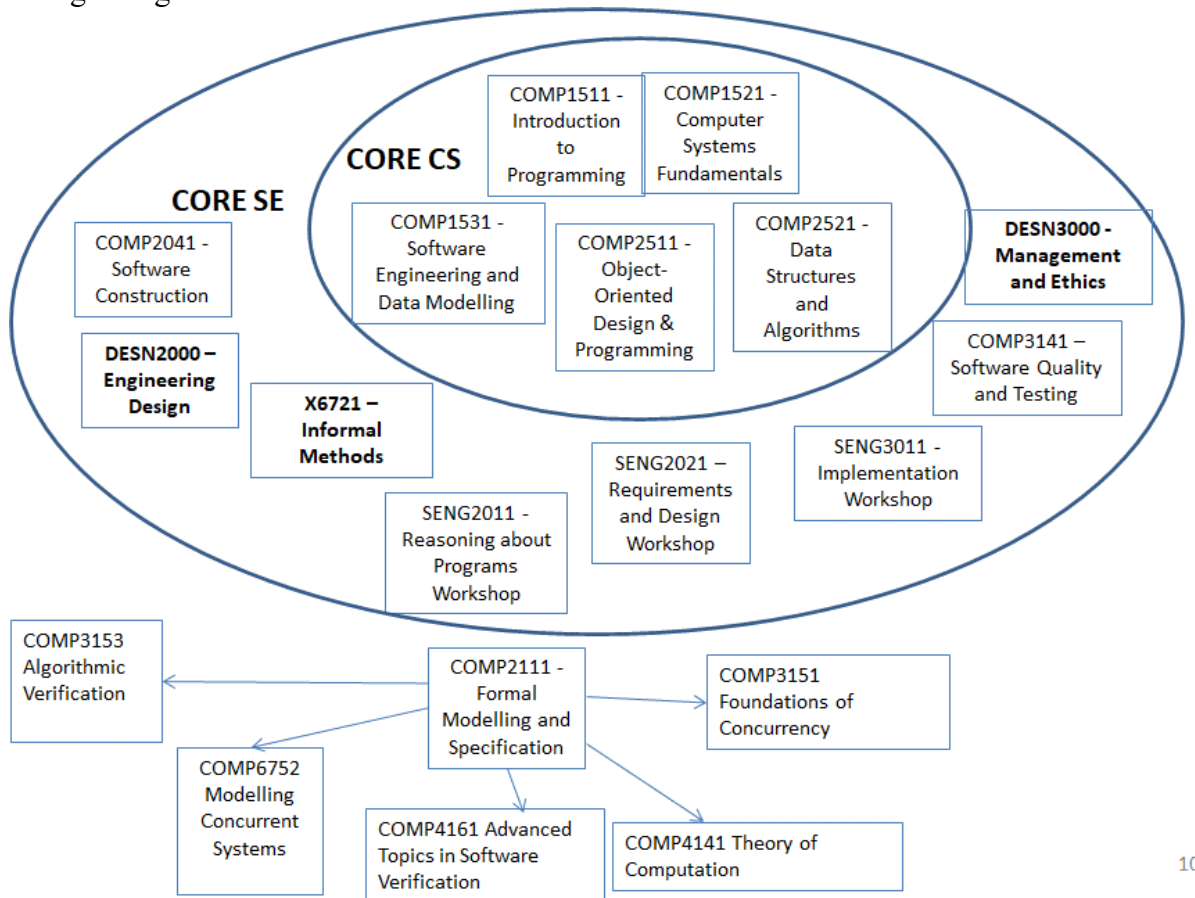
Management and Ethics

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.)	

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



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



Revisiting Learning Objectives

Comments are invited for possible changes

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, 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 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

APPENDIX A: SWEBOK Areas

Chapter 1: Software Requirements 1-1

1. Software Requirements Fundamentals 1-1
 - 1.1. Definition of a Software Requirement 1-1
 - 1.2. Product and Process Requirements 1-2
 - 1.3. Functional and Nonfunctional Requirements 1-3
 - 1.4. Emergent Properties 1-3
 - 1.5. Quantifiable Requirements 1-3
 - 1.6. System Requirements and Software Requirements 1-3
2. Requirements Process 1-3
 - 2.1. Process Models 1-4
 - 2.2. Process Actors 1-4
 - 2.3. Process Support and Management 1-4
 - 2.4. Process Quality and Improvement 1-4
3. Requirements Elicitation 1-5
 - 3.1. Requirements Sources 1-5
 - 3.2. Elicitation Techniques 1-6
4. Requirements Analysis 1-7
 - 4.1. Requirements Classification 1-7
 - 4.2. Conceptual Modeling 1-8
 - 4.3. Architectural Design and Requirements Allocation 1-9
 - 4.4. Requirements Negotiation 1-9
 - 4.5. Formal Analysis 1-10
5. Requirements Specification 1-10
 - 5.1. System Definition Document 1-10
 - 5.2. System Requirements Specification 1-10
 - 5.3. Software Requirements Specification 1-11
6. Requirements Validation 1-11

6.1. Requirements Reviews	1-11
6.2. Prototyping	1-12
6.3. Model Validation	1-12
6.4. Acceptance Tests	1-12
7. Practical Considerations	1-12
7.1. Iterative Nature of the Requirements Process	1-13
7.2. Change Management	1-13
7.3. Requirements Attributes	1-13
7.4. Requirements Tracing	1-14
7.5. Measuring Requirements	1-14
8. Software Requirements Tools	1-14
Chapter 2: Software Design	2-1
1. Software Design Fundamentals	2-2
1.1. General Design Concepts	2-2
1.2. Context of Software Design	2-2
1.3. Software Design Process	2-2
1.4. Software Design Principles	2-3
2. Key Issues in Software Design	2-3
2.1. Concurrency	2-4
2.2. Control and Handling of Events	2-4
2.3. Data Persistence	2-4
2.4. Distribution of Components	2-4
2.5. Error and Exception Handling and Fault Tolerance	2-4
2.6. Interaction and Presentation	2-4
2.7. Security	2-4
3. Software Structure and Architecture	2-4
3.1. Architectural Structures and Viewpoints	2-5
3.2. Architectural Styles	2-5
3.3. Design Patterns	2-5
3.4. Architecture Design Decisions	2-5
3.5. Families of Programs and Frameworks	2-5
4. User Interface Design	2-5
4.1. General User Interface Design Principles	2-6
4.2. User Interface Design Issues	2-6
4.3. The Design of User Interaction Modalities	2-6
4.4. The Design of Information Presentation	2-6
4.5. User Interface Design Process	2-7
4.6. Localization and Internationalization	2-7
4.7. Metaphors and Conceptual Models	2-7
5. Software Design Quality Analysis and Evaluation	2-7
5.1. Quality Attributes	2-7
5.2. Quality Analysis and Evaluation Techniques	2-8
5.3. Measures	2-8
6. Software Design Notations	2-8
6.1. Structural Descriptions (Static View)	2-8
6.2. Behavioral Descriptions (Dynamic View)	2-9
7. Software Design Strategies and Methods	2-10
7.1. General Strategies	2-10
7.2. Function-Oriented (Structured) Design	2-10
7.3. Object-Oriented Design	2-10
7.4. Data Structure-Centered Design	2-10
7.5. Component-Based Design (CBD)	2-10
7.6. Other Methods	2-10
8. Software Design Tools	2-11
Chapter 3: Software Construction	3-1
1. Software Construction Fundamentals	3-1
1.1. Minimizing Complexity	3-3

1.2. Anticipating Change	3-3
1.3. Constructing for Verification	3-3
1.4. Reuse	3-3
1.5. Standards in Construction	3-3
2. Managing Construction	3-4
2.1. Construction in Life Cycle Models	3-4
2.2. Construction Planning	3-4
2.3. Construction Measurement	3-4
3. Practical Considerations	3-5
3.1. Construction Design	3-5
3.2. Construction Languages	3-5
3.3. Coding	3-6
3.4. Construction Testing	3-6
3.5. Construction for Reuse	3-6
3.6. Construction with Reuse	3-7
3.7. Construction Quality	3-7
3.8. Integration	3-7
4. Construction Technologies	3-8
4.1. API Design and Use	3-8
4.2. Object-Oriented Runtime Issues	3-8
4.3. Parameterization and Generics	3-8
4.4. Assertions, Design by Contract, and Defensive Programming	3-8
4.5. Error Handling, Exception Handling, and Fault Tolerance	3-9
4.6. Executable Models	3-9
4.7. State-Based and Table-Driven Construction Techniques	3-9
4.8. Runtime Configuration and Internationalization	3-10
4.9. Grammar-Based Input Processing	3-10
4.10. Concurrency Primitives	3-10
4.11. Middleware	3-10
4.12. Construction Methods for Distributed Software	3-11
4.13. Constructing Heterogeneous Systems	3-11
4.14. Performance Analysis and Tuning	3-11
4.15. Platform Standards	3-11
4.16. Test-First Programming	3-11
5. Software Construction Tools	3-12
5.1. Development Environments	3-12
5.2. GUI Builders	3-12
5.3. Unit Testing Tools	3-12
5.4. Profiling, Performance Analysis, and Slicing Tools	3-12
Chapter 4: Software Testing	4-1
1. Software Testing Fundamentals	4-3
1.1. Testing-Related Terminology	4-3
1.2. Key Issues	4-3
1.3. Relationship of Testing to Other Activities	4-4
2. Test Levels	4-5
2.1. The Target of the Test	4-5
2.2. Objectives of Testing	4-5
3. Test Techniques	4-7
3.1. Based on the Software Engineer's Intuition and Experience	4-8
3.2. Input Domain-Based Techniques	4-8
3.3. Code-Based Techniques	4-8
3.4. Fault-Based Techniques	4-9
3.5. Usage-Based Techniques	4-9
3.6. Model-Based Testing Techniques	4-10
3.7. Techniques Based on the Nature of the Application	4-10
3.8. Selecting and Combining Techniques	4-11
4. Test-Related Measures	4-11
4.1. Evaluation of the Program Under Test	4-11

4.2. Evaluation of the Tests Performed	4-12
5. Test Process	4-12
5.1. Practical Considerations	4-13
5.2. Test Activities	4-14
6. Software Testing Tools	4-15
6.1. Testing Tool Support	4-15
6.2. Categories of Tools	4-15
Chapter 5: Software Maintenance	5-1
1. Software Maintenance Fundamentals	5-1
1.1. Definitions and Terminology	5-1
1.2. Nature of Maintenance	5-2
1.3. Need for Maintenance	5-3
1.4. Majority of Maintenance Costs	5-3
1.5. Evolution of Software	5-3
1.6. Categories of Maintenance	5-3
2. Key Issues in Software Maintenance	5-4
2.1. Technical Issues	5-4
2.2. Management Issues	5-5
2.3. Maintenance Cost Estimation	5-6
2.4. Software Maintenance Measurement	5-7
3. Maintenance Process	5-7
3.1. Maintenance Processes	5-7
3.2. Maintenance Activities	5-8
4. Techniques for Maintenance	5-10
4.1. Program Comprehension	5-10
4.2. Reengineering	5-10
4.3. Reverse Engineering	5-10
4.4. Migration	5-10
4.5. Retirement	5-11
5. Software Maintenance Tools	5-11
Chapter 6: Software Configuration Management	6-1
1. Management of the SCM Process	6-2
1.1. Organizational Context for SCM	6-2
1.2. Constraints and Guidance for the SCM Process	6-3
1.3. Planning for SCM	6-3
1.4. SCM Plan	6-5
1.5. Surveillance of Software Configuration Management	6-5
2. Software Configuration Identification	6-6
2.1. Identifying Items to Be Controlled	6-6
2.2. Software Library	6-8
3. Software Configuration Control	6-8
3.1. Requesting, Evaluating, and Approving Software Changes	6-8
3.2. Implementing Software Changes	6-9
3.3. Deviations and Waivers	6-10
4. Software Configuration Status Accounting	6-10
4.1. Software Configuration Status Information	6-10
4.2. Software Configuration Status Reporting	6-10
5. Software Configuration Auditing	6-10
5.1. Software Functional Configuration Audit	6-11
5.2. Software Physical Configuration Audit	6-11
5.3. In-Process Audits of a Software Baseline	6-11
6. Software Release Management and Delivery	6-11
6.1. Software Building	6-11
6.2. Software Release Management	6-12
7. Software Configuration Management Tools	6-12
Chapter 7: Software Engineering Management	7-1

1. Initiation and Scope Definition 7-4
 - 1.1. *Determination and Negotiation of Requirements* 7-4
 - 1.2. *Feasibility Analysis* 7-4
 - 1.3. *Process for the Review and Revision of Requirements* 7-5
 2. Software Project Planning 7-5
 - 2.1. *Process Planning* 7-5
 - 2.2. *Determine Deliverables* 7-5
 - 2.3. *Effort, Schedule, and Cost Estimation* 7-6
 - 2.4. *Resource Allocation* 7-6
 - 2.5. *Risk Management* 7-6
 - 2.6. *Quality Management* 7-6
 - 2.7. *Plan Management* 7-7
 3. Software Project Enactment 7-7
 - 3.1. *Implementation of Plans* 7-7
 - 3.2. *Software Acquisition and Supplier Contract Management* 7-7
 - 3.3. *Implementation of Measurement Process* 7-7
 - 3.4. *Monitor Process* 7-7
 - 3.5. *Control Process* 7-8
 - 3.6. *Reporting* 7-8
 4. Review and Evaluation 7-8
 - 4.1. *Determining Satisfaction of Requirements* 7-8
 - 4.2. *Reviewing and Evaluating Performance* 7-9
 5. Closure 7-9
 - 5.1. *Determining Closure* 7-9
 - 5.2. *Closure Activities* 7-9
 6. Software Engineering Measurement 7-9
 - 6.1. *Establish and Sustain Measurement Commitment* 7-9
 - 6.2. *Plan the Measurement Process* 7-10
 - 6.3. *Perform the Measurement Process* 7-11
 - 6.4. *Evaluate Measurement* 7-11
 7. Software Engineering Management Tools 7-11
- Chapter 8: Software Engineering Process 8-1**
1. Software Process Definition 8-2
 - 1.1. *Software Process Management* 8-3
 - 1.2. *Software Process Infrastructure* 8-4
 2. Software Life Cycles 8-4
 - 2.1. *Categories of Software Processes* 8-5
 - 2.2. *Software Life Cycle Models* 8-5
 - 2.3. *Software Process Adaptation* 8-6
 - 2.4. *Practical Considerations* 8-6
 3. Software Process Assessment and Improvement 8-6
 - 3.1. *Software Process Assessment Models* 8-7
 - 3.2. *Software Process Assessment Methods* 8-7
 - 3.3. *Software Process Improvement Models* 8-7
 - 3.4. *Continuous and Staged Software Process Ratings* 8-8
 4. Software Measurement 8-8
 - 4.1. *Software Process and Product Measurement* 8-9
 - 4.2. *Quality of Measurement Results* 8-10
 - 4.3. *Software Information Models* 8-10
 - 4.4. *Software Process Measurement Techniques* 8-11
 5. Software Engineering Process Tools 8-12
- Chapter 9: Software Engineering Models and Methods 9-1**
1. Modeling 9-1
 - 1.1. *Modeling Principles* 9-2
 - 1.2. *Properties and Expression of Models* 9-3
 - 1.3. *Syntax, Semantics, and Pragmatics* 9-3
 - 1.4. *Preconditions, Postconditions, and Invariants* 9-4

- 2. Types of Models 9-4
 - 2.1. *Information Modeling* 9-5
 - 2.2. *Behavioral Modeling* 9-5
 - 2.3. *Structure Modeling* 9-5
- 3. Analysis of Models 9-5
 - 3.1. *Analyzing for Completeness* 9-5
 - 3.2. *Analyzing for Consistency* 9-6
 - 3.3. *Analyzing for Correctness* 9-6
 - 3.4. *Traceability* 9-6
 - 3.5. *Interaction Analysis* 9-6
- 4. Software Engineering Methods 9-7
 - 4.1. *Heuristic Methods* 9-7
 - 4.2. *Formal Methods* 9-7
 - 4.3. *Prototyping Methods* 9-8
 - 4.4. *Agile Methods* 9-9
- Chapter 10: Software Quality 10-1**
 - 1. Software Quality Fundamentals 10-2
 - 1.1. *Software Engineering Culture and Ethics* 10-2
 - 1.2. *Value and Costs of Quality* 10-3
 - 1.3. *Models and Quality Characteristics* 10-3
 - 1.4. *Software Quality Improvement* 10-4
 - 1.5. *Software Safety* 10-4
 - 2. Software Quality Management Processes 10-5
 - 2.1. *Software Quality Assurance* 10-5
 - 2.2. *Verification & Validation* 10-6
 - 2.3. *Reviews and Audits* 10-6
 - 3. Practical Considerations 10-9
 - 3.1. *Software Quality Requirements* 10-9
 - 3.2. *Defect Characterization* 10-10
 - 3.3. *Software Quality Management Techniques* 10-11
 - 3.4. *Software Quality Measurement* 10-12
 - 4. Software Quality Tools 10-12

Chapter 11: Software Engineering Professional Practice 11-1

- 1. Professionalism 11-2
 - 1.1. *Accreditation, Certification, and Licensing* 11-3
 - 1.2. *Codes of Ethics and Professional Conduct* 11-4
 - 1.3. *Nature and Role of Professional Societies* 11-4
 - 1.4. *Nature and Role of Software Engineering Standards* 11-4
 - 1.5. *Economic Impact of Software* 11-5
 - 1.6. *Employment Contracts* 11-5
 - 1.7. *Legal Issues* 11-5
 - 1.8. *Documentation* 11-7
 - 1.9. *Tradeoff Analysis* 11-8
- 2. Group Dynamics and Psychology 11-9
 - 2.1. *Dynamics of Working in Teams/Groups* 11-9
 - 2.2. *Individual Cognition* 11-9
 - 2.3. *Dealing with Problem Complexity* 11-10
 - 2.4. *Interacting with Stakeholders* 11-10
 - 2.5. *Dealing with Uncertainty and Ambiguity* 11-10
 - 2.6. *Dealing with Multicultural Environments* 11-10
- 3. Communication Skills 11-11
 - 3.1. *Reading, Understanding, and Summarizing* 11-11
 - 3.2. *Writing* 11-11
 - 3.3. *Team and Group Communication* 11-11
 - 3.4. *Presentation Skills* 11-12

Chapter 12: Software Engineering Economics 12-1

- 1. Software Engineering Economics Fundamentals 12-3

- 1.1. Finance 12-3
- 1.2. Accounting 12-3
- 1.3. Controlling 12-3
- 1.4. Cash Flow 12-3
- 1.5. Decision-Making Process 12-4
- 1.6. Valuation 12-5
- 1.7. Inflation 12-6
- 1.8. Depreciation 12-6
- 1.9. Taxation 12-6
- 1.10. Time-Value of Money 12-6
- 1.11. Efficiency 12-6
- 1.12. Effectiveness 12-6
- 1.13. Productivity 12-6
- 2. Life Cycle Economics 12-7
- 2.1. Product 12-7
- 2.2. Project 12-7
- 2.3. Program 12-7
- 2.4. Portfolio 12-7
- 2.5. Product Life Cycle 12-7
- 2.6. Project Life Cycle 12-7
- 2.7. Proposals 12-8
- 2.8. Investment Decisions 12-8
- 2.9. Planning Horizon 12-8
- 2.10. Price and Pricing 12-8
- 2.11. Cost and Costing 12-9
- 2.12. Performance Measurement 12-9
- 2.13. Earned Value Management 12-9
- 2.14. Termination Decisions 12-9
- 2.15. Replacement and Retirement Decisions 12-10
- 3. Risk and Uncertainty 12-10
- 3.1. Goals, Estimates, and Plans 12-10
- 3.2. Estimation Techniques 12-11
- 3.3. Addressing Uncertainty 12-11
- 3.4. Prioritization 12-11
- 3.5. Decisions under Risk 12-11
- 3.6. Decisions under Uncertainty 12-12
- 4. Economic Analysis Methods 12-12
- 4.1. For-Profit Decision Analysis 12-12
- 4.2. Minimum Acceptable Rate of Return 12-13
- 4.3. Return on Investment 12-13
- 4.4. Return on Capital Employed 12-13
- 4.5. Cost-Benefit Analysis 12-13
- 4.6. Cost-Effectiveness Analysis 12-13
- 4.7. Break-Even Analysis 12-13
- 4.8. Business Case 12-13
- 4.9. Multiple Attribute Evaluation 12-14
- 4.10. Optimization Analysis 12-14
- 5. Practical Considerations 12-14
- 5.1. The “Good Enough” Principle 12-14
- 5.2. Friction-Free Economy 12-15
- 5.3. Ecosystems 12-15
- 5.4. Offshoring and Outsourcing 12-15
- Chapter 13: Computing Foundations 13-1**
- 1. Problem Solving Techniques 13-3
- 1.1. Definition of Problem Solving 13-3
- 1.2. Formulating the Real Problem 13-3
- 1.3. Analyze the Problem 13-3
- 1.4. Design a Solution Search Strategy 13-3
- 1.5. Problem Solving Using Programs 13-3

2. Abstraction	13-4
2.1. Levels of Abstraction	13-4
2.2. Encapsulation	13-4
2.3. Hierarchy	13-4
2.4. Alternate Abstractions	13-5
3. Programming Fundamentals	13-5
3.1. The Programming Process	13-5
3.2. Programming Paradigms	13-5
4. Programming Language Basics	13-6
4.1. Programming Language Overview	13-6
4.2. Syntax and Semantics of Programming Languages	13-6
4.3. Low-Level Programming Languages	13-7
4.4. High-Level Programming Languages	13-7
4.5. Declarative vs. Imperative Programming Languages	13-7
5. Debugging Tools and Techniques	13-8
5.1. Types of Errors	13-8
5.2. Debugging Techniques	13-8
5.3. Debugging Tools	13-8
6. Data Structure and Representation	13-9
6.1. Data Structure Overview	13-9
6.2. Types of Data Structure	13-9
6.3. Operations on Data Structures	13-9
7. Algorithms and Complexity	13-10
7.1. Overview of Algorithms	13-10
7.2. Attributes of Algorithms	13-10
7.3. Algorithmic Analysis	13-10
7.4. Algorithmic Design Strategies	13-11
7.5. Algorithmic Analysis Strategies	13-11
8. Basic Concept of a System	13-11
8.1. Emergent System Properties	13-11
8.2. Systems Engineering	13-12
8.3. Overview of a Computer System	13-12
9. Computer Organization	13-13
9.1. Computer Organization Overview	13-13
9.2. Digital Systems	13-13
9.3. Digital Logic	13-13
9.4. Computer Expression of Data	13-13
9.5. The Central Processing Unit (CPU)	13-14
9.6. Memory System Organization	13-14
9.7. Input and Output (I/O)	13-14
10. Compiler Basics	13-15
10.1. Compiler/Interpreter Overview	13-15
10.2. Interpretation and Compilation	13-15
10.3. The Compilation Process	13-15
11. Operating Systems Basics	13-16
11.1. Operating Systems Overview	13-16
11.2. Tasks of an Operating System	13-16
11.3. Operating System Abstractions	13-17
11.4. Operating Systems Classification	13-17
12. Database Basics and Data Management	13-17
12.1. Entity and Schema	13-18
12.2. Database Management Systems (DBMS)	13-18
12.3. Database Query Language	13-18
12.4. Tasks of DBMS Packages	13-18
12.5. Data Management	13-19
12.6. Data Mining	13-19
13. Network Communication Basics	13-19

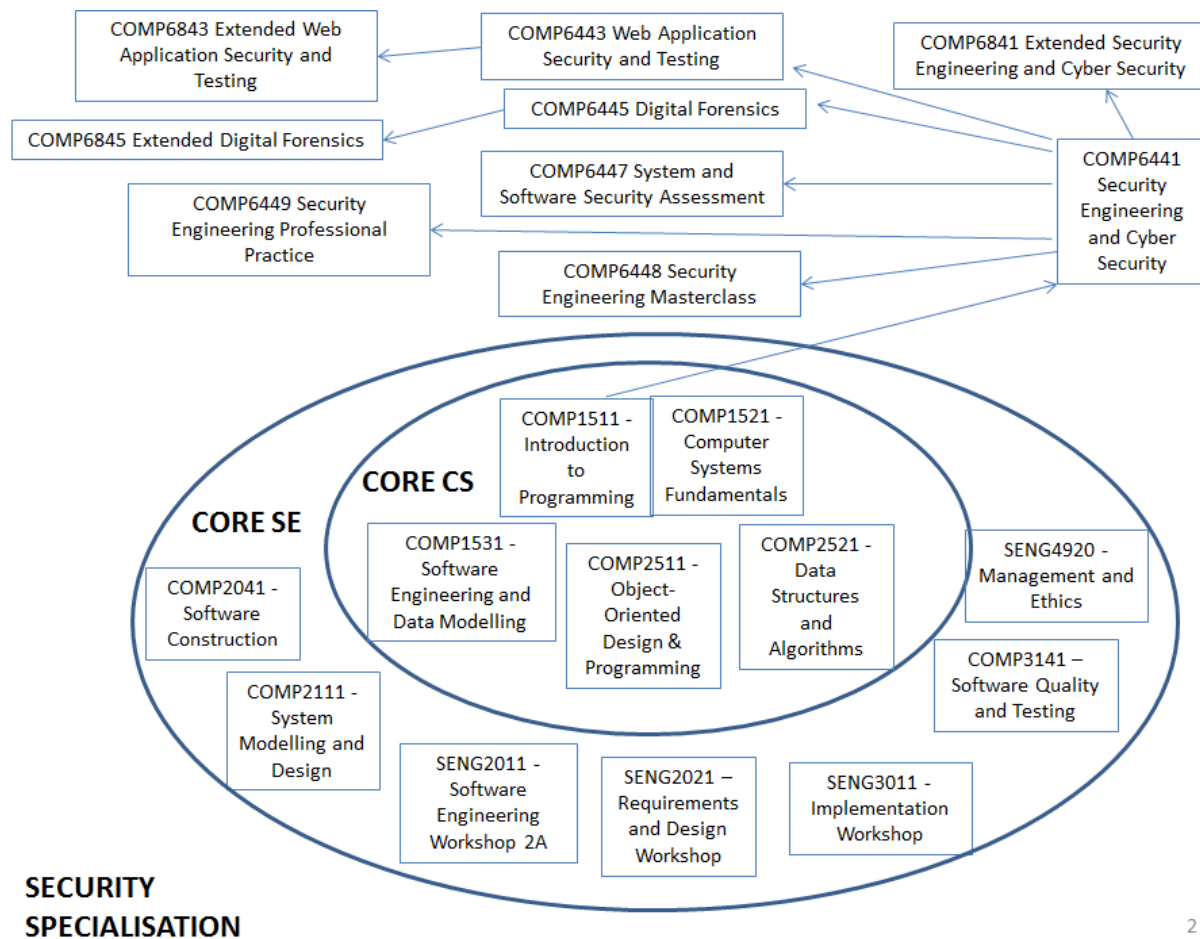
13.1. Types of Network	13-19
13.2. Basic Network Components	13-19
13.3. Networking Protocols and Standards	13-20
13.4. The Internet	13-20
13.5. Internet of Things	13-20
13.6. Virtual Private Network (VPN)	13-21
14. Parallel and Distributed Computing	13-21
14.1. Parallel and Distributed Computing Overview	13-21
14.2. Difference between Parallel and Distributed Computing	13-21
14.3. Parallel and Distributed Computing Models	13-21
14.4. Main Issues in Distributed Computing	13-22
15. Basic User Human Factors	13-22
15.1. Input and Output	13-22
15.2. Error Messages	13-23
15.3. Software Robustness	13-23
16. Basic Developer Human Factors	13-23
16.1. Structure	13-24
16.2. Comments	13-24
17. Secure Software Development and Maintenance	13-24
17.1. Software Requirements Security	13-24
17.2. Software Design Security	13-25
17.3. Software Construction Security	13-25
17.4. Software Testing Security	13-25
17.5. Build Security into Software Engineering Process	13-25
17.6. Software Security Guidelines	13-25
Chapter 14: Mathematical Foundations 14-1	
1. Set, Relations, Functions	14-1
1.1. Set Operations	14-2
1.2. Properties of Set	14-3
1.3. Relation and Function	14-4
2. Basic Logic	14-5
2.1. Propositional Logic	14-5
2.2. Predicate Logic	14-5
3. Proof Techniques	14-6
3.1. Methods of Proving Theorems	14-6
4. Basics of Counting	14-7
5. Graphs and Trees	14-8
5.1. Graphs	14-8
5.2. Trees	14-10
6. Discrete Probability	14-13
7. Finite State Machines	14-14
8. Grammars	14-15
8.1. Language Recognition	14-16
9. Numerical Precision, Accuracy, and Errors	14-17
10. Number Theory	14-18
10.1. Divisibility	14-18
10.2. Prime Number, GCD	14-19
11. Algebraic Structures	14-19
11.1. Group	14-19
11.2. Rings	14-20
Chapter 15: Engineering Foundations 15-1	
1. Empirical Methods and Experimental Techniques	15-1
1.1. Designed Experiment	15-1
1.2. Observational Study	15-2
1.3. Retrospective Study	15-2
2. Statistical Analysis	15-2

- 2.1. *Unit of Analysis (Sampling Units), Population, and Sample* 15-2
- 2.2. *Concepts of Correlation and Regression* 15-5
- 3. Measurement 15-5
 - 3.1. *Levels (Scales) of Measurement* 15-6
 - 3.2. *Direct and Derived Measures* 15-7
 - 3.3. *Reliability and Validity* 15-8
 - 3.4. *Assessing Reliability* 15-8
- 4. Engineering Design 15-8
 - 4.1. *Engineering Design in Engineering Education* 15-8
 - 4.2. *Design as a Problem Solving Activity* 15-9
 - 4.3. *Steps Involved in Engineering Design* 15-9
- 5. Modeling, Simulation, and Prototyping 15-10
 - 5.1. *Modeling* 15-10
 - 5.2. *Simulation* 15-11
 - 5.3. *Prototyping* 15-11
- 6. Standards 15-12
- 7. Root Cause Analysis 15-12
 - 7.1. *Techniques for Conducting Root Cause Analysis* 15-13

APPENDIX B: Specialisations in 2016 review

Security specialisation

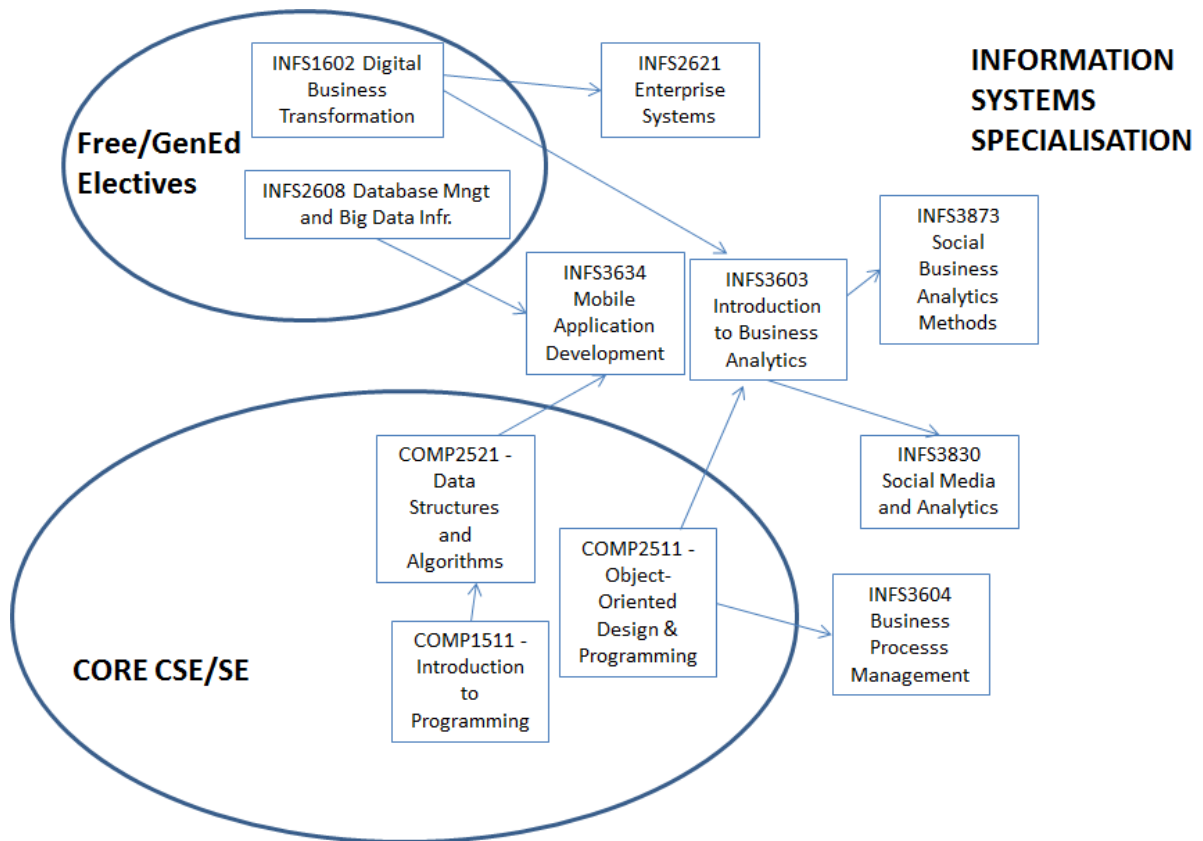
Defined around new courses funded by CBA initiative.



2

Information Systems Specialisation

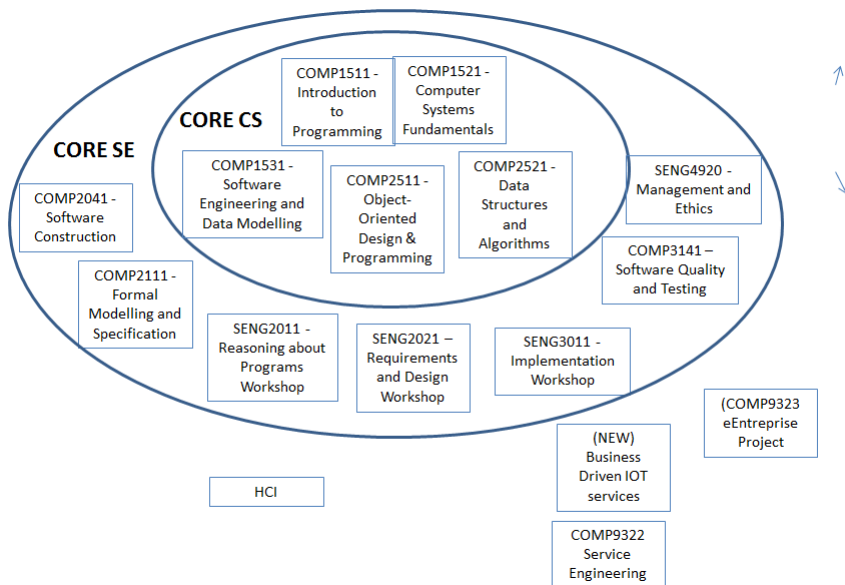
Defined around courses offered by SISTM (www.sistm.unsw.edu.au)



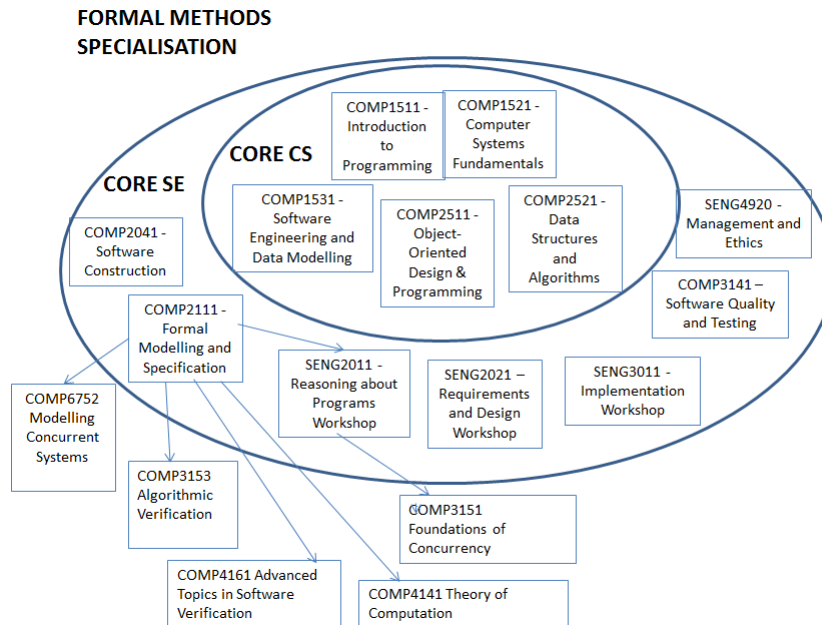
Software as a Service Specialisation

Defined around the work of Service Oriented Computing Research Group

<https://sites.google.com/site/unswsoc/>



Formal Methods Specialisation



Data science Specialisation (awaiting information)

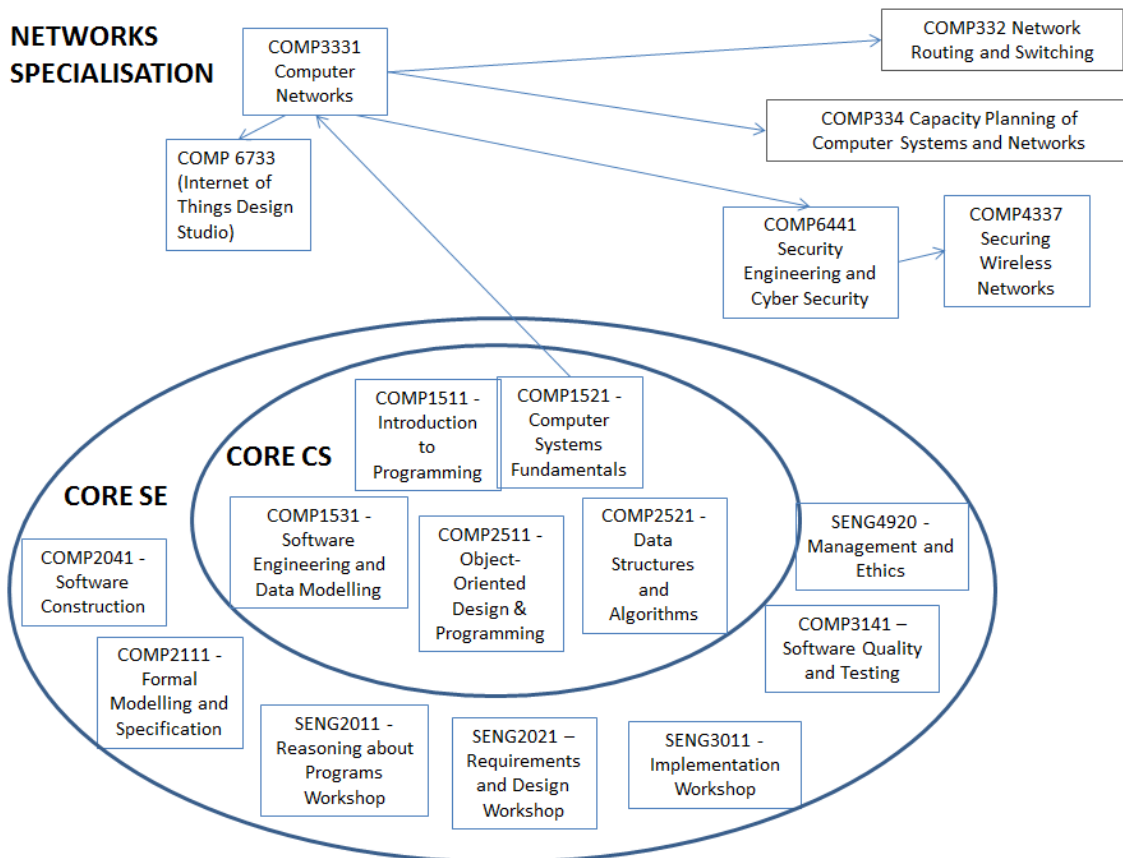
Defined around the work of Databases Research Group

<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>



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

