SE Degree Handbook

(04/10/2016)

SE Degree General Information

Comment [FA1]: This is reproduced from brochure. Comments welcome

What is Software Engineering?

Software Engineering is an Engineering profession concerned with the processes, methods and tools for the design and development of high quality, reliable software systems. This involves the study and application of software specification, design, implementation, testing and documentation of software. Target systems may range from simple software applications to mission-critical real-time systems.

Career Opportunities

The software industry is one of the fastest growing industries in the world. Even companies that have been associated largely with hardware in the past estimate that 80-90% of their engineers are involved in software development. As a consequence of this rapid expansion there is a serious worldwide shortage of software engineers who are able to deal with the complexity of developing high-quality software systems.

Given the ubiquitous nature of software in modern society, software engineers can find employment opportunities in many areas. These include, but are not limited to, Information and Communication Technologies (ICT), Business, Hardware and Defence industries.

Assumed Knowledge

Mathematics Extension 1, English Standard Band 3 or English (ESL) Band 4.
Students who do not meet these levels should contact our Student Office about alternatives, including bridging courses and alternative Program structures.

Advantageous Knowledge

Mathematics Extension 2.
Subjects listed under the Advantageous Knowledge will be useful for a more in-depth study of the field. Obtaining a result in Band E4 in Mathematics Extension 2 allows students to take the higher level maths course MATH1141.

SE Degree Learning Outcomes

Comment [FA2]: These are being revised as part of the SE Degree Revision Process. Comments still welcome

Formatted: Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0.63 cm + Indent at: 1.27 cm

1. Demonstrate a solid understanding of the software engineering knowledge and skills, necessary to begin practice as a software engineer.
2. ability to appropriately define and apply relevant abstractions from algorithmics, computer science, and mathematics to complex software system development.
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.
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.

5. ability to think and design software systems from the perspective of the end user and to communicate clearly and effectively with business stakeholders
6. have the understanding that software interacts with many different domains and the ability to be able to communicate with, and learn from, experts from different domains
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.

SE Degree Structure

Core Courses Level 1

Core Courses Students must take 42 UOC of the following courses.

- COMP1511 - Introduction to Programming (6 UOC)
- COMP1521 - Computer Systems Fundamentals (6 UOC) (Proposed Apr 2015)
- COMP1531 - Software Engineering Fundamentals (6 UOC) (Proposed Apr 2015)
- one of the following:
  - MATH1131 - Mathematics 1A (6 UOC)
  - MATH1141 - Higher Mathematics 1A (6 UOC)
- MATH1081 - Discrete Mathematics (6 UOC)
- ENGG1000 - Introduction to Engineering Design and Innovation (6 UOC)
- one of the following:
  - MATH1231 - Mathematics 1B (6 UOC)
  - MATH1241 - Higher Mathematics 1B (6 UOC)

Core Courses Level 2

Core Courses Students must take 48 UOC of the following courses.

- COMP2041 - Software Construction: Techniques and Tools (6 UOC)
- COMP2111 - System Modelling and Design (6 UOC)
- COMP2511 - Object-Oriented Design & Programming (6 UOC) (Proposed Apr 2016)
- COMP2521 - Data Structures and Algorithms (6 UOC) (Proposed May 2016)
- MATH2400 - Finite Mathematics (3 UOC)
- MATH2859 - Probability, Statistics and Information (3 UOC)
- SENG2011 - Software Engineering Workshop 2A (6 UOC)
- SENG2021 - Software Engineering Workshop 2B (6 UOC)

Core Courses Level 3

Core Courses Students must take 24 UOC of the following courses.

- SENG3011 - Software Engineering Workshop 3 (6 UOC)
- COMP3311 - Database Systems (6 UOC)
- COMP3141 - Software System Design and Implementation (6 UOC)

Comment [FA3]: Content still being finalised (see track changes on outline)

Comment [FA4]: Possibly rename this course as it is more concerned with Unix/scripting/OS/Web programming

Comment [FA5]: A proposal for creating a course "Mathematical Foundations of Computer Science" is underway

Comment [FA6]: Content still being finalised (see track changes on outline)

Comment [FA7]: A proposal for aligning this workshop with "Mathematical Foundations" is underway

Comment [FA8]: Since data modelling is now in first year, this course will become an elective to create more space for electives

Comment [FA9]: Possibly rename this course to focus on testing and functional programming

- COMP3331 - Computer Networks and Applications (6 UOC)

Comment [FA10]: It is proposed to make this an elective to create more space for electives

Core Courses Level 4

Core Courses Students must take 18 UOC of the following courses.

- COMP4920 - Management and Ethics (6 UOC)
- COMP4930 - Thesis Part A (6 UOC)
- COMP4931 - Thesis Part B (6 UOC)

Prescribed Electives

Professional Electives. Students must take 36 UOC of the following courses.

- any level 3 Computer Science (COMP) course
- any level 4 Computer Science (COMP) course
- any level 6 Computer Science (COMP) course
- any level 9 Computer Science (COMP) course
- any level 3 Information Systems (INFS) course
- any level 4 Information Systems (INFS) course
- any level 3 Mathematics (MATH) course
- any level 4 Mathematics (MATH) course
- any level 6 Mathematics (MATH) course
- any level 3 Electrical Engineering (ELEC) course
- any level 4 Electrical Engineering (ELEC) course
- any level 3 Telecommunications (TELE) course
- any level 4 Telecommunications (TELE) course

Limit Requirements

Level 4 UOC Minimum Students must complete a minimum of 30 UOC of the following courses.

- COMP4920 - Management and Ethics (6 UOC)
- COMP4930 - Thesis Part A (6 UOC)
- COMP4931 - Thesis Part B (6 UOC)
- any level 4 Computer Science (COMP) course
- any level 4 Information Systems (INFS) course
- any level 4 Mathematics (MATH) course
- any level 4 Electrical Engineering (ELEC) course

Industrial Training

At least 60 days of approved Industrial Training must be completed before graduation. Industrial Training should be concurrent with enrolment and is best accumulated in the summer recesses at the end of years 2 and 3.

Overview of Core computing courses

Most up-to-date version available from

http://webapps.cse.unsw.edu.au/cse/core/index.php

- COMP1511 Introduction to Programming
- COMP1521 Computer Systems Fundamentals
- COMP1531 Software Engineering and Data Modelling
- COMP2511 Object-oriented Design and Programming
- COMP2521 Data Structures and Algorithms

More details will be added as the courses are developed.

COMP1531 Software Engineering and Data Modelling~~Fundamentals - Proposed~~

School of Computer Science and Engineering, UNSW

Overview

Code/Title	COMP1531 Software Engineering Fundamentals
Abbreviations	SEF, 1531
Units of Credit	6
Pre-requisites	COMP1511
Excluded	SENG1031
Equivalent	SENG1031
Offered In	S1, S2 (commencing 17s2)
Classes	2 hours lectures/week, 2 hours tute-lab/week, 1 hour mentor meeting/week
Assessment	Exam (theory+prac), Labs, Assignments, Quizzes
Technologies	Python, Linux, web frameworks, HTML, css, javascript, git

Introduction

This course provides an introduction to software engineering principles: basic software lifecycle concepts, modern development methodologies, conceptual modelling and how these activities relate to programming. It also introduces the basic notions of team-based project management via conducting a project to design, build and deploy a simple web-based application. It is typically taken in the semester after completing COMP1511, but could be delayed and taken later. It provides essential background for the teamwork and project management required in many later courses.

The goal of this course is to expose the students to:

- Basic principles of conceptual data modelling and databases
- basic elements of software engineering - derived from the lifecycle of a software system, including requirements elicitation, analysis and specification; design; construction; verification and validation; deployment; and operation and maintenance
- software engineering methodologies, processes, tools and techniques
- Web-based project~~system architecture~~ and development practices on Web platforms

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Assumed Knowledge

We assume that students have taken a first programming course, which has included exposure to a moderate-sized, team-based project and some testing/debugging ideas.

Learning Outcomes

- On successful completion of this course, students should be able to ...
- describe the phases of software development and life-cycle of software - and illustrate them from experience
- understand conceptual data modelling and develop simple data models
- get some exposure to~~effectively choose and use a range of~~ project management and software development tools
- describe common behaviour that contribute to the effective functioning of a team and identify necessary roles in a software development team

~~articulate software design principles and use a design paradigm to design a simple software system (e.g., simple Web application in MVC)~~

- understand agile methods and the principles of testing, code development and validating software~~demonstrate robust coding practices (e.g., handling exceptions, following coding standards)~~

~~describe effective coding validation and verification techniques (e.g., code reviewing, fault logging, a range of test measures)~~

- ~~demonstrate effective usage of testing fundamentals (e.g., unit tests, integration tests, test plan/cases, test automation)~~
- understand the architecture of simple Web systems

Topics

- software processes and project management
- software tools and development environments
- web technologies and web application archtectures
- software requirements engineering
- software architectures and software design
- agile software development practice
- software construction
- software validation
- teamwork strategies

Schedule

Week	Lectures	Labs
Week 1	Intro (to course aims and software engineering). Sotware Design Cycle. Methodologies for Software Development	xxx
~~Week 2~~	~~Web applications (I) - architecture, principles, HTTP request/response,~~	~~xxx~~

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Formatted Table

	~~URI/URL~~	
~~Week 3~~	~~Web applications (II) - web app design patterns (model-view-controller)~~	~~xxx~~
~~Week 4~~	~~Software Processes and Project Management - different software development models, software lifecycle, management tools, communication/conflict resolution strategies~~	~~xxx~~
Week 2~~5~~	Software Requirement Engineering - requirement elicitation, describing system data using UML or ER, functional/non-functional requirements	xxx
~~Week 6~~	~~Software Design - software architectures (e.g., client-server, n-layer), design paradigms (OO, event-driven, component-based, aspect-oriented, service-oriented, etc.), Web application architectures~~	~~xxx~~
Weeks 3-6	Conceptual data modelling and introduction to databases	
Week 7	Agile Software Development Practice - requirement engineering/software design/coding/testing in practice	xxx
Week 8	Software Construction - robust coding practice (tracking defects/logging, checking input, initialisation, exception handling, coding standards, framework development)	xxx
Week 9	Group Project. Overview of ~~Software Tools and Environment - unix and scripting,~~ version control, build/deployment management, tool integration	xxx
Week 10	Group project: ~~Software Validation - testing, testing design, automation~~Web front-ends, Javascript/CSS Introduction	xxx
Week 11	Group Project Demonstrations	-
Week 12	Group Project Showcase	-

Formatted Table

COMP2511 Object-Oriented Design & Programming

School of Computer Science and Engineering, UNSW

Overview

Code/Title	COMP2511 Object-Oriented Design & Programming
Abbreviations	OOP, OODP, 2511
Units of Credit	6
Pre-requisites	COMP1531, COMP2521
Excluded	COMP2911
Equivalent	COMP2911
Offered In	S1, S2 (commencing 18s1)
Classes	3 hours lectures/week, 2 hours tute-lab/week
Assessment	Exam (theory/prac), Labs, Assignments, Quizzes
Technologies	Java, UML, JUnit, Eclipse

Introduction

This course aims to introduce students to the principles of object-oriented design and to fundamental techniques in object-oriented programming. Such knowledge is important in later project-based courses.

The goal of the course is to expose students to:

- the fundamental principles of object-oriented design
- an object-oriented programming language such as Java
- the systematic application of object-oriented software design processes
- problem solving and modelling real world problems using the object-oriented paradigm

Assumed Knowledge

Students should have some experience in working on team-based assignments, and at least two semesters of programming, including a course on data structures and algorithms. This requirement restricts the course to being taken in second year or later.

Learning Outcomes

On successful completion of this course, students should be able to ...

Formatted: Heading 1, None, Space Before: 0 pt, After: 0 pt, Pattern: Clear

Comment [WW11]: I'm going to use only 2 hours lectures, but think it should be kept at 3 hours so that whoever else teaches this may use 3 hours.

Comment [WW12]: I use a "theory" exam so I've changed this to make it flexible that someone else can use a theory OR prac OR both exam. Also I don't assess Labs or Quizzes. But others might.

Formatted: Font: +Body (Calibri), Strikethrough

Formatted: Font: +Body (Calibri), Strikethrough

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Comment [WW13]: So here it could be something other than Java, but below there are some Java-specific things in the schedule.

- design appropriate solutions to medium-scale problems ~~using~~ using ~~Java~~an object-oriented approach
- apply a systematic ~~the~~ object-oriented design ~~principles~~ process as a part of following software engineering best practices~~such as separation of concerns, responsibility analysis, and design by contract~~
- apply an ~~object-oriented analysis and design practice~~agile software development method to ~~complex software systems~~organize team-based projects
- ~~to~~ create ~~and refactor~~ medium-scale object-oriented programs ~~in Java~~ using appropriate design principles~~.~~
- ~~make use of the most important design patterns~~use appropriate software engineering tools for the development of medium-scale software systems

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Formatted: Font: +Body (Calibri), 12 pt, Font color: Auto, Strikethrough

Comment [WW14]: Does COMP2511 require Java?

Topics

- object-oriented design
- object-oriented programming
- agile ~~development~~ software processes
- design patterns

~~methods~~

- ~~programming~~ introduction to user ~~interfaces~~interface design and programming

~~parallelism~~

- ~~design patterns~~introduction to concurrency

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Schedule

Week	Lectures	Tut-Labs
Week 1	Principles of Design; Software Engineering; Java basics, Eclipse~~: Classes, Objects, Abstraction, Inheritance, Polymorphism, class hierarchy~~	~~xxx~~
Week 2	Object-Oriented Programming (Objects vs ADTs, inheritance, encapsulation, polymorphism, Java object model, equality, cloning)~~O Design Process and Tools: UML and CRC~~	Basic Java programming~~xxx~~
Week 3	~~Engineering Design~~Design by Contract (Pre- and Postconditions, Class invariants, exceptions, Javadoc, Unit testing, JUnit)	Programming by Contract~~xxx~~
Week 4	~~OO Design Principles: separation of concerns, collaboration analysis, design by contract~~Object-Oriented Design (Use cases, CRC Cards, UML)	~~xxx~~Object-Oriented Design
Week 5	~~Development Methods: Testing OO Systems, Code Review, Debugging, Documentation, Agile Development~~Generic Types and Polymorphism (Java type system, generic types, Arrays and Lists, Interface types)	~~xxx~~Generic Types
Week	Introduction to Design Patterns (Iterator Pattern, Strategy	~~xxx~~Search

6	Pattern)~~Generics; Collections; Iterators; Frameworks~~	Algorithms
Week 7	~~Interface Design and Polymorphism; Inheritance in OO Design and Composition~~Problem-Solving Algorithms (A* search, heuristics)	~~xxx~~Problem-Solving Algorithms
Week 8	~~User Interfaces; Event-Driven & Reactive Programming (MVC)~~Agile Software Processes (Scrum, Extreme Programming, Agile Planning and Estimation)	~~xxx~~Agile Software Processes
Week 9	~~Design Patterns~~User Interface Design and Programming (Java GUI programming, event-driven programming, Observer Pattern)	~~xxx~~User Interface Programming
Week 10	Complex ~~Parallelism vs. Concurrency; Decomposition; Synchronisation; Communication; Coordination~~Design Patterns (Decorator Pattern, Composite Pattern)	~~xxx~~Sprint Reviews
Week 11	~~Atomicity; Consistency; Shared Memory; Concurrency Bugs (e.g. data races & deadlocks)~~Introduction to Concurrency (multi-threaded systems, race conditions, synchronization, locks)	~~xxx~~Design Patterns
Week 12	~~More Design Patterns~~Review	Concurrency~~xxx~~

COMP1521 Computer Systems Fundamentals

School of Computer Science and Engineering, UNSW

Overview

Code/Title	COMP1521 Computer Systems Fundamentals
Abbreviations	CSF, 1521
Units of Credit	6
Pre-requisites	COMP1511
Excluded	-
Equivalent	-
Offered In	S1, S2 (commencing 17s2)
Classes	3 hours lectures/week, 3 hours tute-lab/week
Assessment	Exam, Labs, Assignments, Quizzes
Technologies	C, Linux, make, gdb, git

Introduction

This course provides a programmer's view on how a computer system executes programs, manipulates data and communicates. It enables students to become effective programmers in dealing with issues of performance, portability, and robustness. It is typically taken in the semester after completing COMP1511, but could be delayed and taken later. It serves as a foundation for later courses on networks, operating systems, computer architecture and compilers, where a deeper understanding of systems-level issues is required.

The goal of the course is to expose students to:

- this
- that
- something else

Formatted: Outline numbered + Level: 1 + Numbering Style: Bullet + Aligned at: 0.63 cm + Tab after: 1.27 cm + Indent at: 1.27 cm

Assumed Knowledge

The course assumes that students have completed a first course in programming in the C programming language. Students who completed their first programming course in a higher-level language such as Java, C++ or Python are encouraged to ???

Learning Outcomes

On successful completion of this course, students should be able to ...

- Describe the layers of architectures in modern computer systems from hardware device levels upwards
- Describe the principles of memory management and explain the workings of a system with virtual memory management
- Explain how the major components of a CPU work together, including how data (including instructions) is represented in a computer
- Design, implement and analyse small programs at the assembly/machine level, including the use of I/O, interrupts and traps
- Describe the relationship between high-level procedural languages (e.g., C) and assembly/machine language in the conventional machine layer, including how a compiled program is executed in a classical von Neumann machine, with extensions for threads, multiprocessor synchronization, and SIMD execution
- Explain how input/output operations are implemented, and describe some basic I/O devices
- Describe the layered structure of a typical networked architecture

Formatted: Outline numbered + Level: 1 + Numbering Style: Bullet + Aligned at: 0.63 cm + Tab after: 1.27 cm + Indent at: 1.27 cm

Topics

- architecture of computer systems
- machine-level programming
- mapping HLLs to machine-level
- runtime representation of HLL programs (stack, heap, code)
- memory architectures: virtual memory, caching
- input/output, disk devices, interrupts
- processor/memory architecture
- parallelism, synchronisation, coordination
- overview of operating system architecture
- overview of network architecture

Formatted: Outline numbered + Level: 1 + Numbering Style: Bullet + Aligned at: 0.63 cm + Tab after: 1.27 cm + Indent at: 1.27 cm

Schedule

Week	Lectures	Labs
Week 1	introduction/overview, review of C, number representation	bit-manipulation
Week 2	data structures: stacks, queues, linked lists	accuracy of C computations
Week 3	machine-level programming	linked lists
Week 4	machine-level programming	xxx

Week 5	compilation, assembly, linking, loading; runtime memory, stack, heap	xxx
Week 6	memory hierarchy, caching, locality	xxx
Week 7	virtual memory, file systems, operating systems	xxx
Week 8	I/O, disk, exceptions (interrupts & traps)	file manipulation
Week 9	computer systems as a series of layers, Flynn's taxonomy, shared vs distributed, SMP, SIMD (incl. GPUs), embedded processors, desktops, servers, mobile devices	xxx
Week 10	parallelism: thread-level, instruction-level, data-level, task-level	xxx
Week 11	synchronisation, coordination, communication	parallel programming
Week 12	I/O revisited, networks	

COMP1511 Introduction to Programming

School of Computer Science and Engineering, UNSW

Overview

Code/Title	COMP1511 Introduction to Computing
Abbreviations	ITP, 1511
Units of Credit	6
Pre-requisites	none
Excluded	COMP1917, COMP1911, COMP1921
Equivalent	COMP1917
Offered In	S1, S2 (commencing 17s1)
Classes	3 hours lectures/week, 3 hours tute-lab/week
Assessment	Exam (theory+prac), Labs, Assignments, Quizzes
Technologies	C, Linux, gcc, make, git, gdb

Introduction

This course aims to introduce students to the practice of developing software solutions to (simple) problems. It would typically be taken in a student's first semester of study. It forms a critical introduction to both computing and the CSE community, and leads on to all of the other courses offered by CSE.

The goal of the course is to expose students to:

- fundamentals of programming
- problem-solving via software
- data structures
- how objects are represented in memory
- issues of code quality
- pair programming and teamwork
- software development as an engineering discipline

Formatted: Outline numbered + Level: 1 + Numbering Style: Bullet + Aligned at: 0.63 cm + Tab after: 1.27 cm + Indent at: 1.27 cm

Assumed Knowledge

We assume minimal background in computing, but do assume that students have solid HSC maths and can speak reasonable english. We assume no background in programming or computing, but offer familiarisation labs prior to and early in semester for students who want to learn more about the CSE lab environment.

Learning Outcomes

On successful completion of this course, students should be able to ...

- Design software solutions for simple problems

Formatted: Outline numbered + Level: 1 + Numbering Style: Bullet + Aligned at: 0.63 cm + Tab after: 1.27 cm + Indent at: 1.27 cm

- Design software solutions for larger problems using abstraction and interfaces
- Distinguish between well-written programs and poorly written programs
- Write programs using good programming style
- Understand and appropriately use abstraction
- Effectively use memory and pointers in C
- Understand the low-level functioning of computers (memory, instructions)
- Create and use simple dynamic data structures such as linked lists and trees
- Know a range of sorting algorithms and be able to compare their performance
- Explain the complexity of simple algorithms
- Test and debug programs
- Work in a team to develop software

Topics

- Introduction to Programming
- Abstraction
- Variables and storage
- Control flow: if, while, functions
- Structured data: arrays, structs, linked lists
- Problem analysis, Design principles
- Craftsmanship and style
- Code review & writing codes to be read and modified
- Searching: linear, binary, simple hashing
- Sorting: selection, insertion, quicksort
- Introduction to complexity
- Programming in the large - programming and design principles
- Unit testing and debugging
- Professionalism: time management, teamwork, quality
- Issues in large projects - practical issues and shipping working product
- Agile development
- Simple networking (web based)
- Major group project (involving agile development and unit testing)

Formatted: Outline numbered + Level: 1 + Numbering Style: Bullet + Aligned at: 0.63 cm + Tab after: 1.27 cm + Indent at: 1.27 cm

Schedule

Week	Lectures	Labs
Week 1	introduction, about university, learning, abstraction, estimation, programming, C, problem solving	introduction, lab/workstation orientation, first program
Week 2	types, variables, memory, design with functions, top down design, arithmetic expressions, layout, programming style	program style, version control
Week 3	control structures (choice (if), repetition (while)), more functions, scope, defining vs declaring, pass by copy, unit testing	defining, using and testing functions
Week 4	chars, strings, memory, addresses, pointers	BMP file format, version control
Week 5	arrays, using arrays, pointers and arrays, passing arrays, time management	fractals

Week 6	run-time stack, frames, typedef	team project, group formation
Week 7	heap, malloc, structs, abstraction, abstract data types (ADTs), teamwork	buffer overflow, stack frame hacking
Week 8	more ADTs, standards, interfaces, concrete vs abstract types	Quiet Week (no labs)
Week 9	dynamic structures, linked lists, realloc	implementing an ADT
Week 10	searching in arrays, sorting algorithms, complexity	linked lists
Week 11	project management, testing, unit tests, software development methodologies	sorting
Week 12	professionalism, ethics	practice Prac Exam

COMP2521 Data Structures and Algorithms

School of Computer Science and Engineering, UNSW

Overview

Code/Title	COMP2521 Data Structures and Algorithms
Abbreviations	DSA, 2521
Units of Credit	6
Pre-requisites	COMP1511
Excluded	COMP1927
Equivalent	COMP1927
Offered In	S1, S2 (commencing 17s2)
Classes	3 hours lectures/week, 3 hours tute-lab/week
Assessment	Exam (theory+prac), Labs, Assignments, Quizzes
Technologies	C, gcc, make, git, gdb

Introduction

The goal of this course is to deepen students' understanding of data structures and algorithms and how these can be employed effectively in the design of software systems. We anticipate that it will generally be taken in the second year of a program (and it was originally assigned a level-2 course code), but since its only pre-requisite is ITP, is it possible to take it in first year. It is an important course in covering a range of core data structures and algorithms that will be used in context in later courses.

Assumed Knowledge

On entry to the course, we assume that students can:

1. implement software in the procedural paradigm up to several 1000's LoC
2. employ a range of fundamental data types in developing software solutions (e.g. arrays, structs, matrices, sets, lists, ...)
3. can design and implement simple abstract data types

Formatted: Outline numbered + Level: 1 + Numbering Style: Bullet + Aligned at: 0.63 cm + Tab after: 1.27 cm + Indent at: 1.27 cm

4. reason about the behaviour (correctness and efficiency) of programs
   (e.g. defining pre- and post-conditions, efficiency of sorting algorithms)
5. explain how programs work at the machine level (stack, heap, ...)
6. work effectively in teams, following a systematic development process
7. develop and use test suites for functions and programs
8. work with a range of tools for program develoment
   (editors, compilers, debuggers, profilers, version control systems)
9. effectively use structures from discrete mathematics
   (e.g. sets/relations/functions, basic logic, proof techniques from MATH1081)

Learning Outcomes

On successful completion of this course, students should be able to ...

1. implement software in the procedural paradigm up to several 10,000's LoC
2. use a range of algorithmic strategies in problem-solving
3. reason about a wide range of data structures and their algorithms
4. analyse the performance characteristics of algorithms
5. measure the performance behaviour of programs
6. reason about the correctness of programs
7. choose/justify/implement an appropriate data structure for a given problem
8. choose/analyse/implement appropriate algorithms to manipulate this data structure
9. describe a range of fundamental concepts in parallelism

Formatted: Outline numbered + Level: 1 + Numbering Style: Bullet + Aligned at: 0.63 cm + Tab after: 1.27 cm + Indent at: 1.27 cm

Topics

1. fundamental data structures: lists, trees, graphs
2. algorithm and program analysis
3. techniques for sorting, searching, traversing

Formatted: Outline numbered + Level: 1 + Numbering Style: Bullet + Aligned at: 0.63 cm + Tab after: 1.27 cm + Indent at: 1.27 cm

Schedule

Week	Lectures	Labs
Week 1	Introduction; Revision of data structures + ADTs + O(n)	Linked-list revision
Week 2	Sorting Review, Parallel Sorting	External merge-sort
Week 3	Algorithmic Strategies: recursion, divide-and-conquer, brute-force	Sorting Detective
Week 4	Graphs: Representation, Traversal, Paths, Tours	Debugging with gdb
Week 5	Graph Algorithms: Shortest Path, MSTs	Web crawling and directed graphs
Week 6	Fundamentals of Tree Structures	Minimum Cost Paths
Week 7	Searching (trees): Balanced Trees	Tree Construction and Traversal
Week 8	Searching (tables): Hashing	Balanced Trees

Week 9	Searching (files): Files, B-Trees, Linear Hashing	Hashing Performance Experiment
Week 10	Searching (text): substring, regular expressions, LCS	B-Tree Performance Analysis

COMP2041/9041 - Software Construction - Course Outline

Course Staff

Staff Name	Role	Email
Andrew Taylor	Lecturer & Admin	andrewt@cse.unsw.edu.au

Class Details

Day	Time	Room
Tuesday	13:00-15:00	Rex Vowels (EE LG1)
Thursday	17:00-18:00	Rex Vowels (EE LG1)

You will have chosen a 3 hour tut-lab slot when you enrolled.
Consultations times vary through session and are listed on the course home page.

Distance Stream

A distance (WEB) stream is available. Students in this stream will need to rely on lecture recordings and the material placed on the web. Student should consider carefully whether this is sufficient for them to successfully complete the course.

Communication

Sometimes urgent information may be sent to you by email. Make sure you pay careful attention to any email you receive.
All official email will be sent to your CSE email address which is by dfefault forwarded to your UNSW address. If you redirect your mail please do so carefully & check that your redirection works.
Additional information will be provided in the Course Forum (linked to the class home page). You should read it regularly. The forums is the best place to ask questions about the course.

Course Summary

The following is a summary of the topics that will be covered in this course.

1. Tools for software construction
    - Programming languages (C)
    - Scripting languages (Perl, Shell, brief intro to python)
    - Filters (sort, sed, grep, tr, ...)
    - Analysis tools (debuggers, profilers)
    - development tools (make, git, ...)

2. Techniques for software construction
    - Analysis, design, coding, testing, debugging, tuning
    - Interface design, documentation, configuration
3. Qualities of software systems
    - Correctness, clarity, reliability, efficiency, portability, ...

The focus for the practical work will be on C and Perl. However, you would be well advised to acquaint yourselves with the facilities provided by the Unix shell.

Course Aims

This course is designed for students who have mastered the basics of programming. It aims to broaden your knowledge of techniques and tools for software construction.

Learning Outcomes

By the end of the course, you should have these attributes which will be useful to you for the remainder of your studies and after graduation:

- have practical experience in programming the scripting languages Perl, the Unix shell and optionally some Python.
- have a broader & deeper knowledge of building software systems
- more appreciation of the use of specific technologies and strategies during software development
- exposure to tools for version control, performance improvement, configuration and debugging,
- improvement of your ability to articulate & communicate concepts related to programming & systems

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Assumed Knowledge:

COMP2041/9041 assumes that you have a sound understanding of a procedural programming language such as C and can:

- produce a correct procedural program from a spec
- understand fundamental data structures + algorithms (char, int, float, array, struct, pointers, sorting, searching)
- appreciate use of abstraction in computing
-

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

For undergraduate (COMP2041) students, the above material will have been covered in first year courses such as COMP1917 Higher Computing 1 and COMP1927 Higher Data Str. & Algos.

For postgraduate (COMP9041) students, the above material will have been covered in COMP9021 Principles of Programming and COMP9024 Data Structures/Algorithms, or similar material will have been covered in their undergraduate degree.

A limited amount of specific knowledge of the C programming language may be assumed during the course.

Students who are not competant C programmers should discuss with the lecturer at the first lecture the impact this might have. Typically students who are competant in a similar languages such as C++ and Java only need to do some extra reading.

Teaching Strategies

Lectures: Lectures will be used to present the theory and practice of the techniques and tools in this course. There will be extensive use of case studies and practical demonstrations during lectures. Lecture notes will be available on the course web pages before each lecture. **Tutorials** From week 2 you will also be expected to attend a one-hour tutorial session to clarify ideas from lectures and work through exercises based on the lecture material. You should make sure that you use them effectively by examining in advance the material to be covered in each week's tute, by asking questions, by offering suggestions and by generally participating. The tutorial questions will be posted on the Web in the week before each tute. There are no marks for tutorial attendance. **Laboratory Classes**: following the tute class each week, there will be a two-hour lab class, during which you will work on a variety of small practical problems involving the tools introduced in lectures. Because this course is practical in nature, lab class are a very important component, and you should make every effort to attend the labs and complete the exercises diligently. In particular, **keep up-to-date** with the Lab work; if you fall behind it affects your ability to understand later material in the course.
To obtain a mark for a lab exercise you must both demonstrate the completed lab exercise to your tutor during a lab class and submit it using give.

If you don't complete a lab exercise during the scheduled class, you can still obtain the mark if you both submit the completed exercises before midnight Monday and you demonstrate it to you tutor in the follow week's lab.

COMP9041 students are recommended to attend tutorials and labs, but may opt to complete the work in their own time and not have it formally assessed. To do this, they must advise the lecturer by email by the end of week 2.

Assignments

In the assignment work, you will work through the process of building and/or modifying software systems, using the tools and techniques described in lectures. The assignment work will focus on Perl and Perl+CGI.
There will be two assignments the first will be a Perl application due week 7/8, the second due will be Perl/CGI due week 12. They will be of roughly equal weight.

Exam

There will be a three-hour primarily practical exam, to be held in the CSE labs during the exam period. It consists of five small implementation tasks and one written section. During

this exam you will be able to execute, debug and test your answers. The implementation tasks will be similar to those encountered in lab exercises

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

It is a hurdle requirement for this course that you pass the exam.
It also is a hurdle requirement for this course that you perform satisfactorily on the implementation tasks in the exam. This is defined as successfully at least two of the five implementation tasks.

Teaching Rationale

This course has a heavy practical orientation. Lectures will revolve around live demonstrations of programming and use of tools. Labs & assignments form a key part.

Assessment

Component	Value
Lab Work	10%
Assignments	30%
Exam	60%

These assessment weights might be varied by a few percent when the assignments have been chosen.
If your final exam mark is less than 50% then your overall mark will not be allowed to exceed your exam mark. In other words you must pass the final exam to pass the course.
As mentioned above, your performance on the practical component of the final exam must also be satisfactory to pass the course.
The lecturer may scale overall marks, or individual components, up or down to obtain a desired mark distribution.
You may be excluded from the prac exam if you have < 10/40 for assignments+labs.

```
ass    = mark for assignments     (out of 30)
labs   = mark for assessed labs   (out of 10)
exam   = mark for exam            (out of 60)

okExam =  two implementation tasks solved on exam

mark = ass + labs + pexam + texam
grade    = HD|DN|CR|PS  if mark >= 50 && okExam
         = FL           if mark < 50 && okExam
         = UF           if !okExam
```

Academic honesty and plagiarism

What is Plagiarism?

Plagiarism is the presentation of the thoughts or work of another as one's own.*
Examples include:

1. direct duplication of the thoughts or work of another, including by copying material, ideas or concepts from a book, article, report or other written document (whether published or unpublished), composition, artwork, design, drawing, circuitry, computer program or software, web site, Internet, other electronic resource, or another person's assignment without appropriate acknowledgement;
2. paraphrasing another person's work with very minor changes keeping the meaning, form and/or progression of ideas of the original;
3. piecing together sections of the work of others into a new whole;
4. presenting an assessment item as independent work when it has been produced in whole or part in collusion with other people, for example, another student or a tutor; and
5. claiming credit for a proportion a work contributed to a group assessment item that is greater than that actually contributed.

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

For the purposes of this policy, submitting an assessment item that has already been submitted for academic credit elsewhere may be considered plagiarism. Knowingly permitting your work to be copied by another student may also be considered to be plagiarism.
Note that an assessment item produced in oral, not written, form, or involving live presentation, may similarly contain plagiarised material.
The inclusion of the thoughts or work of another with attribution appropriate to the academic discipline does not amount to plagiarism.
The Learning Centre website is main repository for resources for staff and students on plagiarism and academic honesty. These resources can be located via:www.lc.unsw.edu.au/plagiarism

The Learning Centre also provides substantial educational written materials, workshops, and tutorials to aid students, for example, in:

- correct referencing practices;
- paraphrasing, summarising, essay writing, and time management;
- appropriate use of, and attribution for, a range of materials including text, images, formulae and concepts.

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Individual assistance is available on request from The Learning Centre. Students are also reminded that careful time management is an important part of study and one of the identified causes of plagiarism is poor time management. Students should allow sufficient time for research, drafting, and the proper referencing of sources in preparing all assessment items.

All work submitted for assessment must be your own work. Lab exercises and assignments must be completed *individually*. We regard copying of assignments or lab exercises, in

whole or part, as a very serious offence. We use plagiarism detection software to search for multiply-submitted work.

1. Submitting part or all of other students' work, with or without acknowledgement, is not acceptable.
2. Submitting solutions written by other persons is also not acceptable.
3. Building on ideas and partial solutions obtained from public sources, such as web resources, may be acceptable, provided full acknowledgement is made. However, the final mark will take into account the starting point and how much development work would have been required. Failing to acknowledge web or other resources is unacceptable.
4. Discussing approaches to solutions with other students is quite appropriate, but any discussions should remain at the design level, and must not include program text. Comparison tools will detect any common code across the student body.
5. The safest approach is to work diligently on your own, seeking help from the forum or course staff.
6. Submission of work derived from another person, or jointly written with someone else will, may result in automatic failure for COMP2041/COMP9041 with a mark of zero.
7. Allowing another student to copy from you will may result in a mark of zero for your own assignment or lab exercises. Do not provide your work to any other person, even people who not UNSW students. You will be held responsible for the actions of anyone you provide your work to.
8. Severe or second offences will result in automatic failure, exclusion from the university, and possibly other academic discipline.

Formatted: Numbered + Level: 1 + Numbering Style: 1, 2, 3, ... + Start at: 1 + Alignment: Left + Aligned at: 0.63 cm + Indent at: 1.27 cm

Refer also to the Yellow Form material on plagiarism and the Learning Centre website

Course schedule

The anticipated course sequence is: shell scripting (weeks 1-3), Perl (weeks 3-6), web applications (6-9), programming tools(9-12) We may need to vary this to some degree as we update material in the courses.
The lectures are timetabled for weeks 1-12. It is possible the week 13 lecture slots will be used for a remedial revision lecture or other optional presentations so please keep them free.

Resources for Students

There is no required textbook for the course. Useful reference books include the following:

- Kernighan & Pike, The Practice of Programming,
  Addison-Wesley, 1998.
  (Inspiration for 2041 - philosophy and some tool details)
- McConnell, Code Complete (2ed),
  Microsoft Press, 2004.
  (Many interesting case studies and practical ideas)

- Wall, Christiansen & Orwant , Programming Perl (3ed),
  O'Reilly, 2000. (Original & best Perl reference manual)
- Schwartz, Phoenix & Foy, Learning Perl (5ed),
  O'Reilly, 2008. (gentle & careful introduction to Perl)
- Christiansen & Torkington, Perl Cookbook (2ed),
  O'Reilly, 2003. (Lots and lots of interesting Perl examples)
- Schwartz & Phoenix, Learning Perl Objects, References, and Modules (2ed),
  O'Reilly, 2003. (gentle & careful introduction to parts of Perl mostly not covered in this course)
- Schwartz, Phoenix & Foy, Intermediate Perl (2ed),
  O'Reilly, 2008. (good book to read after 2041 - starts where this course finishes)
- Sebesta, A Little Book on Perl,
  Prentice Hall, 1999. (Modern, concise introduction to Perl)
- Orwant, Hietaniemi, MacDonald, Mastering Algorithms with Perl,
  O'Reilly, 1999. (Algorithms and data structures via Perl)
- Kochgan & Wood 2003, UnixÂ® Shell Programming,
  Sams Publishing 2003 (Careful intoduction to Shell Programming)
- Peek, O'Reilly, Loukides, Bash Cookbook,
  O'Reilly, 2007. (Recipe(example) based intro to Shell programming)
- Powers, Peek, O'Reilly, Loukides, Unix Power Tools (3ed),
  O'Reilly, 2003. (Comprehensive guide to common Unix tools)
- Loukides & Oram, Programming with GNU Software,
  O'Reilly, 1997. (Tutorial on the GNU programming tools (gcc,gdb,...))
- Robbins, Unix in a Nutshell (4ed),
  O'Reilly, 2006. (Concise guide to Unix and its toolset)
- Kernighan & Pike, The Unix Programming Environment,
  Prentice Hall, 1984. (Pre-cursor to the textbook, intro to Unix tools)

For pointers to other useful reading material, including documentation for all of the tools used in the practical work, see the course Web pages.

Course evaluation and development

Student feedback on this course will be obtained via electronic survey at the end of session, and will be used to make continual improvements to the course. Students are also encouraged to provide informal feedback during the session, and to let the lecturer in charge know of any problems, as soon as they arise. Suggestions will be listened to very openly, positively, constructively and thankfully, and every reasonable effort will be made to address them.
This feedback is used to improve the course materials & their delivery. In the most recent session feedback was very favourable probably as results of changes based on previous session's feedback. Some lab exercises and lecture topics will be updated to better reflect current practice.

Other matters

- Occupational Health and Safety policies

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

- Information for students with disabilities Contact the lecturer ASAP if you have any disabilities that may affect this course.

COMP2111 Course Information for Session 1 2016

(The present document ($Revision: 1.3 $) is available online at http://www.cse.unsw.edu.au/~cs2111/16s1/admin/intro.html).

Assumed Knowledge

set theory, propositional logic, first-order logic (all from from MATH1081); general mathematical concepts from MATH1131/1141; programming concepts from COMP1917 & COMP1927; requirements concepts from SENG1031. COMP2111 is a co-requisite for SENG2011 and a prerequisite for SENG2021.

Update Policy

This course introduction should be relatively stable at the beginning of session. Any updates after the initial publication will be discussed in class and announced under the header **Fresh stuff** in the menu on the left hand side of the COMP2111 web site.

Course Aims

The first basic aim is to re-enforce the understanding of *requirements analysis* formed in SENG1031, that is, to understand how to progress from an informal (written or spoken) requirements document to a clear understanding of what the requirements entail, and from there to a *design* or *formal specification*. In particular, this course teaches how rigorous modelling can be used to develop designs that reflect the requirements.

The second basic aim is to learn how to systematically derive implementations from formal specifications using simple mathematics. This can be done with pencil and paper or with tool support. The latter will be demonstrated occasionally in class. The skills acquired feed directly into the SE workshops SENG2011 and SENG2021.

The third basic aim is to develop an understanding of the rules for deriving implementations from specifications. Why do they work? What do we have to prove and how? This is crucial for later industrial software engineering practice.

Course Objectives

After successfully completing this course, you will have increased your understanding of fundamental SE concepts including: *specification, implementation, abstraction, and refinement*.

Graduate Attributes

are important because the disciplinary knowledge that students develop at university is not adequate in itself as the basis for their future lives. Instead, graduates need qualities and skills that equip them for lifelong learning. These include critical thinking and problem-solving skills, communication skills, information literacy skills, and group work skills. says a UNSW policy.

- **Critical thinking and problem-solving skills** will be fostered by tutorial exercises and assigments.
- **Communication skills**, both oral and written, are sharpened by writing and presenting mathematical thought in tutorials and by writing mathematical text for assignments.
- **Group work skills** are developed while solving problems with peers even if this is more so the domain of the companion course SENG2011.

Course Evaluation and Development

I take student feedback seriously. I run my own more detailed online course surveys to find out what works and what doesn't. Fill ratios of these surveys are much higher because students can score a bonus mark for filling in. The feedback is more detailed because the questions are tailored toward the course. Student feedback for the previous sessions can be found here.
The 14s1 edition was a marked improvement over 13s1 yet there's still ample room to make this work better.
Over the last 3 years the focus of COMP2111 has shifted entirely from tool training to understanding the concepts implemented by such tools, in particular the crucial role of assertional reasoning.

Teaching "Strategy" and "Rationale"

Learning in this course happens in lectures (where the white-board is used more than slides), tutorials (where the exploration of questions is guided by the tutor), and assignments (where concepts are reinforced and translated into problem-solving and coding practice).

Keeping Informed and Staying in Touch

Course Web Site

Important notices related to this course will be displayed on the *course home page*, which is located at http://www.cse.unsw.edu.au/~cs2111/. It is your responsibility to check this page regularly.

Email

Only in rare cases urgent information may also be sent to your CSE/UNSW email account. Only requests of a confidential nature should be sent to the course account, cs2111. Anything else should go onto the message board described next. If in doubt, send email and be prepared to receive the one-line answer: *"This can be asked on the forum."* Emails from non-UNSW accounts are ignored.

Message Board

A message board has been set up for this subject. All but the truly confidential questions should be posted there.

RSS Feed

Notifications about anything deemed noteworthy and related to this subject are maintained as an RSS feed. The newest items of that feed are also displayed first on the subject website.

Staff

Name	Role
Kai Engelhardt	lecturer-in-charge

Consultations

If you need the services of a consultant, make an appointment via email and come to Kai's office (217c in K17) at the agreed time. Ring using the phone next to the glass door (x54497).

Text Book

There is no single required textbook for this course. Relevant material will be made available during the session as needed.

Assessment Summary

The final mark is determined as the sum of the marks for the assessable components and bonus marks of the course, *capped by 45 if the exam mark is less than 25*:

Assessable Components

- 5% tutorial participation mark
- 15% assignment 1: due at the end of week 5
- 15% assignment 2: due at the end of week 8
- 15% assignment 3: due at the end of week 11
- 50% exam: a final exam (2 hours) worth up to 50 marks
- 1 bonus mark for filling in an online course survey at the end of the session.

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

SENG2011 Software Engineering Workshop 2A

Course Code:	SENG2011
Course Title:	Software Engineering Workshop 2A
Units of Credit:	6
Course Website:	http://www.cse.unsw.edu.au/~se2011/
Handbook Entry:	http://www.handbook.unsw.edu.au/undergraduate/courses/2016/SENG2011.html
Hours per week:	3 plus a mentor meeting
Prerequisite:	SENG1031
Corequisite:	COMP2111
Lectures and location:	Consult the course timetable.

Communication with students

The course web page will be the main source of announcements and information. You *must* check the course web page regularly, in particular the notices on the home page of the course website. If you miss an important announcement because you did not check the notices web page, no special dispensation will be granted.Urgent announcements may also be sent to your *CSE email address*, and you are equally accountable for such communications. If you have a different email account that you prefer, you should redirect your School account to there: otherwise you might miss important announcements. (Use the mlalias command for that.)

Course Staff

Contact Details

Staff Name	Role	Email	Phone
Albert Nymeyer	Lecturer in Charge	a.nymeyer@unsw.edu.au	none yet

Course Summary

This course teaches practical techniques for computer program development that help us to proceed from informal but precise *requirements,* via more formal and precise *specifications* through to *implementations* that are correct and easy to understand and maintain. Parallel instruction in the underlying methods used in SENG2011 will come from COMP2111.

In the first part of the course, the lectures, presentations and mentor-meetings will be organized around a sequence of from-requirements-to-program examples that will become increasingly demanding. The problems studied in this part will be relatively small in scale, and assignments will be done individually and by hand: the emphasis is on understanding the techniques both in theory and in practice.
The second part of the course will consist of a group work project that will require the application of the techniques studied in the first part to a larger scale problem. This part of the course will use a tool that automates some of the work required to establish that the program developed satisfies its specification.
There will also be lectures on Project Management. Students will be expected to apply project management techniques in their project work, and document their application of these techniques.

Course Aims

- Engineering: learn how to construct an abstract specification/model of a software system, applying both informal and formal approaches, and how to take it towards an implementation.
- Project Management: to learn the basics of working in a software-project team so that you can deploy effectively the engineering techniques above while working in collaboration with others.

Student Learning Outcomes

After completing this course, students should:

- be able to build a rigorous specification of a program or program-component's required behaviour;
- have an appreciation of the role of software engineering in software-system development;
- be able to reason abstractly about requirements and be able to model them using various informal/formal methods; and
- have learned that both behavioural (human based) and technical views (computer based) of design and implementation are important issues.

Assumed Knowledge

- elementary discrete mathematics
- programming to an introductory level in a simple procedural language (e.g. C, Python or similar)
- competence with Unix (e.g. Linux) commands, and the operating-system structure as presented to its users.

Teaching Strategies

SENG2011 is a workshop in which the teaching strategy is primarily to communicate techniques and to give experience. There are fewer than usual formal lectures in the style of explaining knowledge and facts.

The lectures that *are* in this course are to provide students with support and guidance in developing their practical skills. Students will be expected and encouraged to stand before the class to explain and justify the approach they have taken to developing particular programs from their requirements.

The lecture schedule is tentative and may vary during the semester as the need arises. Changes to the lecture schedule will be announced on the course website.

The remaining contact with teachers will primarily be with group *mentors*, who will give detailed guidance on understanding how to describe rigorously what a program should do and then to guarantee that the program does it. Students will be organised into groups of 7â€“8, and as groups are *required* to spend an average of 30 minutes with their group mentor per week, at a time suitable to the mentor but arranged in constellation with her/his students.

It is recommended that these mentor sessions be held in one of the small seminar rooms in the CSE building, but this will depend on availability. It is essential to find a time suitable for all members as attendance at (mentor) group meetings is compulsory.

Assessment

The total course mark will be combined from marks for these components:

- Specification/programming assignments (90% of total mark): There will be a number of these assignments, no more than 4 but possibly fewer. Their due dates will be announced during the semester, and their marks are expected to be as follows:
    - Assignment 1: 20 marks (individual)Â
    - Assignment 2: 20 marks (individual)
    - Assignment 3: 20 marks (individual)
    - Assignment 4: 30 marks (groupwork)

  The groupwork project includes an assessable project management component.
- Participation (10% of total mark): These marks will be awarded based on mentorsâ€™ reports from their meetings during the term.

  Note that, as a special case, if a student attends fewer than half of his/her mentor sessions, then the mark for the *whole course* can be capped at 64%, that is just less than a credit.

- Submissions handed-in late by up to one day (i.e. 24 hours) will be scaled by 85%; for two days it is 75% (that is total scaling, not in addition to the 85%); for three it is 65%; for four it is 50%; and after four days the submission will not be accepted at all (i.e. is scaled by 0%). If you think you have sound reasons to request a waiver of

these rules, e.g. illness or misadventure, you must submit an official request for special consideration, with supporting documentation (e.g. medical certificates) through the formal UNSW central channels (*not* by direct request to the lecturer.)

Academic Honesty and Plagiarism

Plagiarism is defined as using the words or ideas of others and presenting them as your own. UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Learning Centre: Plagiarism and Academic Integrity
- MyUNSW: Plagiarism and Academic Misconduct
- CSE: Addendum to UNSW Plagiarism Guidelines
- CSE: Yellow Form (whose terms you have agreed to)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism.

Resources for Students

The main resources for out-of-lecture study will be lecture summaries, and literature (often online) that will be identified as the course progresses.

In-depth references are as below, but they adopt in general a much more formal approach than the one we will be using:

- Edsger Dijkstra: A discipline of programming
- David Gries: The science of programming
- Edward Cohen: Programming in the 1990_s
- Roland Backhouse: Program construction and verification
- Edsger Dijkstra and Wim Feijen: A method of programming
- Anne Kaldewaij: Programming, the derivation of algorithms

See the above as an indication of what is possible in later years, if you should specialise in programming for its own sake; do not take them as what is required this year.

Course Evaluation and Development

The course was originally structured around a single large groupwork project using the B-method and associated Rodin toolset. It was felt by staff that students needed a deeper understanding of specification and the development of correct programs on a smaller scale before undertaking such a large-scale project. In 2014 there was a change of emphasis to individual work targeting the development of an understanding of specification-in-the-small and a de-emphasis on groupwork and tool use, although there was a still a project management component. This presentation was loosely based on the course COMP 6721: Informal methods that ran in the years 2010-12.Â The disadvantage of this presentation was that it left the project management component disconnected from the rest of the

course and not applied. To better incorporate the project management component, the 2015 edition restored a larger scale problem to be done as a group, applying project management techniques, while retaining an initial focus on developing individual understanding.

SENG2021 Course Details

Course Code:	SENG2021
Course Title:	Software Engineering Workshop 2B
Units of Credit:	6
Course Website:	http://www.cse.unsw.edu.au/~se2021/
Handbook Entry:	http://www.handbook.unsw.edu.au/undergraduate/courses/2015/SENG2021.html

Course Summary

This course represents the third of a series of software engineering workshops designed to teach students to work in teams and apply their knowledge to solve real-life problems. In this workshop, students will be getting experience on different aspects of designing a Web application with a major focus on the front-end. Activities in this course include developing user requirements, producing design documents, designing user interfaces and producing a prototype. The requirements for this project will be given by an industry partner and will relate to designing a realistic infographic application. Most of the teaching will be conducted via mentoring of the teams. At the beginning of the course, some lectures will give background on some key technologies and on how to document designs in general. The course has a number of industry sponsors that incude Powerhouse Museum, Fairfax Media and Macquarie Bank.

Course Aims

To develop:

- a practical appreciation of the software design process;
- an understanding of the relation between specification concepts and implementation considerations;
- an understanding of Web systems software design approaches;
- design documents describing how a specified system will be implemented;
- an executable prototype using Web development tools.
- a report summarising all the activities and experiences of the workshop.

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Student Learning Outcomes

After completing this course, students will:

- reinforce existing knowledge about the concepts and principles of designing quality software within an organisational context.

- learn about the processes of converting requirements to design in a realistic context
- acquire practical design skills, particularly in architectural design and software component integration
- experience the process of implementing a prototype Web system by choosing appropriate languages, libraries and frameworks.
- acquire additional skills involved in working as part of a project team working within strict time constraints.
- learn the process of writing reports and documentation for specific needs.
- get introduced to a new business application domain (infographics in this case)

This course contributes to the development of the following graduate attributes:

Graduate Attribute	Where Acquired
the skills involved in scholarly enquiry	yes
an in-depth engagement with relevant disciplinary knowledge in its interdisciplinary context	yes
the capacity for analytical and critical thinking and for creative problem solving	yes
the ability to engage in independent and reflective learning	yes
the skills to locate, evaluate and use relevant information (Information Literacy)	yes
the capacity for enterprise, initiative and creativity	yes
an appreciation of and respect for, diversity	no
a capacity to contribute to, and work within, the international community	no
the skills required for collaborative and multidisciplinary work	yes
an appreciation of, and a responsiveness to, change	yes
a respect for ethical practice and social responsibility	no
the skills of effective communication	yes

Assumed Knowledge

Before commencing this course, students should have:

- The ability to develop requirements documents
- The ability to design and implement general algorithms
- Basic knowledge of essential design concepts and techniques (equivalent to UML class diagrams and ER)
- Basic knowledge of scripting and Web technologies

- Writing and communication skills

These are assumed to have been acquired in the previous software engineering courses and workshops

Teaching Rationale

In this workshop, we will follow a product-based framework to the project-based learning. A set of intermediate deliverables leading to a product are specified by the stakeholder, a role assumed by the lecturer in charge. Some weekly lecture slots will be used to elaborate on the deliverables and answer general questions. During these meetings, teams are encouraged to discuss their progress and demonstrate work-in-progress. Teams can also arrange additional meetings with the stakeholder if required. A tutor will be available to assist with technical matters and answer queries related to the case study.

Teaching Strategies

Early weeks will consist of lectures; afterwards, all teams will meet weekly with their mentors. The Schedule specifies the activities for each week. Teams are offered the possibility to hold additional mentoring sessions if the need arises. Students can also ask for lectures on particular topics.
The Macquarie Second Year Software Engineering prize is awarded to one team from SENG2021 in a particular year. A number of teams usually three are chosen on the basis of their final demonstration and are asked to prepare a 20 to 30 minute presentation explaining their design and prototype implementation of the current project. The presentation is to be made to members of Macquarie Bank.

Assessment

The assessable components for the course are:

- Three design reports: Week 5 (10%) + Week 7 (10%) +Week 12 (25%)=45%.
- Two software prototypes Week 9 (15%) +Week 12 (30%)=45%.
- Mentoring sessions participation: 10%.
-

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

For more information on these deliverables, see the Course Web page.

It is assumed that all students will have read and understood the regulations outlined in the myUNSW Assessment Policy If you have not familiarised yourself with these it is strongly suggested that you do so. This course will be administered using those regulations.

Academic Honesty and Plagiarism

Plagiarism is defined as using the words or ideas of others and passing them off as your own. UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Plagiarism
- Academic Integrity and Plagiarism
- CSE: Addendum to UNSW Plagiarism Guidelines
- CSE: Yellow Form (whose terms you have agreed to)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism.

Course Schedule

Weekly Schedule (subject to change)

Week	Date	Mentoring (9-11am)	Lecture (11-12noon)	Project
1	26 July	Introduction to the Course, Guest Lecture		Specs available, Form groups
2	2 August	Open Consultation, Guest Lecture		Finalise groups
3	9 August	Mentoring Meeting (project ideas)		-
4	16 August	Mentoring Meeting (design ideas/report 1) + Guest Lecture		-
5	23 August	Mentoring Meeting	Guest Lecture	Design Report 1 Due
6	30 August	Mentoring Meeting	Guest lecture ?	-
7	6 September	Mentoring Meeting		Design Report 2 Due
8	13 September	Mentoring Meeting		-
9	20 September	Public Demonstrations Prototype 1	Mentoring Meeting	-
10	4 October	Mentoring Meeting		-
11	11 October	Mentoring Meeting		-
12	18 October	Public Demonstrations Prototype 2		Design Report 3 Due
13	25 October	Macquarie Second Year Software Engineering Prize		-

Additional details and changes will be posted on the course's noticeboard.

Resources for Students

For domain knowledge (i.e. infographics), students are encouraged to research appropriate sources of information depending on their needs. They are also expected to learn about the basic concepts using Web data sources.

Course Evaluation and Development

This course is evaluated each session using the CATEI system.

During the last CATEI evaluation, students have raised many issues related to the clarity of the specifications given. Although every effort is made to produce good specifications, students must appreciate that most workshops projects are open ended and leave room for innovation by students. Therefore, it is important that they seek information about the project requirements stakeholder (this role being played by LIC) on a continuous basis during mentoring sessions. There were many issues with the marking scheme which will be resolved in the next offering.

SENG3011 Software Engineering Workshop 3

Course Details

Course Code:	SENG3011
Course Title:	Software Engineering Workshop 3
Units of Credit:	6
Course Website:	http://www.cse.unsw.edu.au/~se3011/
Handbook Entry:	http://www.handbook.unsw.edu.au/undergraduate/courses/2016/SENG3011.html

Course Summary

The purpose of the 3rd year software engineering workshop is to give students experience with a group-based large-scale software development project involving a realistic application in the business domain (finance). In this session, teams will be developing a complex software application and the focus is to learn about a new application domain, study the requirements, manage the project, liaise with the stakeholder and deliver high quality working solutions. Another aspect of the workshop is to reinforce skills in software design, testing, reporting and the use of support tools around these activities.

Course Aims

This third year workshop focuses on the issues of designing and implementing a quality software system that conforms to the requirements of a stakeholder. Besides sharpening their design and coding skills, students will have to get immersed in a complex application domain, learn the basic concepts to be able to understand the requirements and continuously communicate with the stakeholder to discuss issues arising from the project such as design trade-offs, additional functionalities, new interface features etc. In addition, students will develop interpersonal communication skills by preparing correctly formatted and structured reports, negotiating technical, management and interpersonal issues within their teams and resolving problems within their development teams using effective conflict resolution techniques.

Unlike previous projects with rigid requirements, groups are encouraged to be creative in their implementation by focusing on delivering the best quality product in consultation with the stakeholder. Consequently, they have the complete freedom in choosing the implementation technologies of their choice e.g. (Java, C++, J2EE, .NET or hybrids) providing they meet the requirements of the stakeholder.

The application domain selected in this session is in the finance area. In this workshop, students will be issued with an initial requirements document outlining the essential features of a product. Additional information will be given on a needs basis throughout the workshop in a variety of ways (emails, regular meetings, class lecture). Many guest lectures will be organised with industry speakers and access to relevant data for the project will be facilitated.

Student Learning Outcomes

After completing this course, students will:

- reinforce existing knowledge about the concepts and principles of software development associated with the implementation of quality software within an organisational context.
- learn about the processes of requirements elicitation and engineering in a realistic context
- acquire practical design skills, particularly in architectural design and software component integration
- experience the process of implementing a quality software system by choosing appropriate languages, libraries and frameworks.
- acquire additional skills involved in working as part of a project team implementing a quality software system within strict time constraints.
- learn the process of writing reports and documentation for specific needs.
- get introduced to a new business application domain (financial market operations in this case)

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

This course contributes to the development of the following graduate attributes:

Graduate Attribute	Where Acquired
the skills involved in scholarly enquiry	yes
an in-depth engagement with relevant disciplinary knowledge in its interdisciplinary context	yes
the capacity for analytical and critical thinking and for creative problem solving	yes
the ability to engage in independent and reflective learning	yes
the skills to locate, evaluate and use relevant information (Information Literacy)	yes
the capacity for enterprise, initiative and creativity	yes
an appreciation of and respect for, diversity	no
a capacity to contribute to, and work within, the international community	no
the skills required for collaborative and multidisciplinary work	yes
an appreciation of, and a responsiveness to, change	yes

a respect for ethical practice and social responsibility	no
the skills of effective communication	yes

Assumed Knowledge

Before commencing this course, students should have:

- The ability to develop complex programs from stated requirements
- The ability to design and implement algorithms for text and data processing
- Good knowledge of design concepts and techniques (equivalent to UML class diagrams and ER)
- Good knowledge of database and Web technologies
- Writing and communication skills

These are assumed to have been acquired in the previous software engineering courses and workshops

Teaching Rationale

This course adopts a project-based approach to Learning and Teaching where students learn through applying their knowledge in situations inspired from real-life. Software development in a group situation is encouraged with the lecturer and other external evaluators guiding and providing continuous feedback to each group.

Teaching Strategies

In this workshop, a set of intermediate deliverables leading to a product are specified by the stakeholder, a role assumed by the lecturer in charge. Some weekly lecture slots will be used to elaborate on the deliverables and answer general questions. During these meetings, teams are encouraged to discuss their progress and demonstrate work-in-progress. Teams can also arrange additional meetings with the stakeholder if required. A tutor will be available to assist with technical matters and answer queries related to the case study. Assistance can also given over email by the lecturer in charge.

Assessment

The assessable components for the course are:

- Functionality and technical quality of the four prototypes (5%+10%+15%+35%=65%).
- Quality of 3 written reports: Initial, Testing and Final Reports (5%+10%+20%=35%).

Details about these assessment components and the deadlines will be given via the course's on-line system.

Academic Honesty and Plagiarism

Plagiarism is defined as using the words or ideas of others and presenting them as your own. UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Learning Centre: Plagiarism and Academic Integrity
- MyUNSW: Plagiarism and Academic Misconduct
- CSE: Addendum to UNSW Plagiarism Guidelines
- CSE: Yellow Form (whose terms you have agreed to)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism.

Course Schedule

Weekly Schedule

- Week1: Introductory lecture. Group Formation.
- Week2: Guest lecture by Optiver. Finalising Groups.
- Week3: Mentoring.
- Week4: Mentoring. Management Report due.
- Week5: Guest lectures by industry.Prototype 1 due.
- Week6: Mentoring Meeting.
- Week7: Mentoring. Prototype 2 due.
- Week8: Public Holiday.
- Week9: Mentoring Meeting. Testing Report due.
- Week10: Public Demonstrations. Prototype 3 public demonstrations
- Week11: Mentoring Meeting.
- Week12: Final demonstrations. Prototype 4 and Final Report due.

Additional details and changes will be posted on the course's noticeboard.

Resources for Students

For domain knowledge, students are encouraged to research appropriate sources of information depending on their needs. They are also expected to learn about the basic concepts using Web sources (more information will be given).

Course Evaluation and Development

This course is evaluated each session using the CATEI system.

In the previous offering of this course, feedback was generally positive mainly because of the industry nature of the project and the involvement of industry partners. The course sponsor (Optiver) gave at the end of the workshop a very clear message that this course

supports software development skills being put into practice and provides a great foundation for working with industry.

Most suggestions for improvements relate to how deliverables were assessed. In response, every effort will be made at explaining the criteria used for marking in the documentation and during mentoring sessions with students.

Having said that, students must also appreciate that workshop projects are not programming assignments. They deliberately have some ambiguities in the requirements and whenever unsure, students should seek advice during mentoring sessions. In industry, it is rare that all requirements are clearly written down in advance for every project and workshops are designed to teach students to cope with uncertainty in business environments.

COMP3311: Database Systems

Course Details

Course Code:	COMP3311
Course Title:	Database Systems
Units of Credit:	6
Course Website:	http://www.cse.unsw.edu.au/~cs3311
Handbook Entry:	http://www.handbook.unsw.edu.au/undergraduate/courses/2016/COMP3311.html

Course Aims

This course aims to explore in depth the practice of developing database applications and the theory behind relational database management systems (RDBMSs). This course focuses on Database Design. It will also give an overview of the technologies used in implementing database management systems and the past, present and future of database systems and database research.

Large data resources are critical to the functioning of just about every significant modern computer application, and so knowledge of how to manage them is clearly important in industry. In the context of further study, understanding how to use databases effectively is essential for courses such as COMP9321 Web Applications Engineering and COMP9322 Service-Oriented Architectures. COMP3311 also provides a foundation for further study in advanced database topics, such as COMP9315 Database Systems Implementation and COMP9318 Data Mining. Database concepts are also relevant in courses such as COMP9319 Web Data Compression and Search and COMP6714 Information Retrieval and Web Search

Student Learning Outcomes

By the end of the course, you should be able to:

- develop accurate, non-redundant data models
- realise data models as relational database schemas
- formulate queries via the full range of SQL constructs
- use stored procedures and triggers to extend DBMS capabilities
- understand principles and techniques for administering RDBMSs
- understand performance issues in relational database applications
- understand the overall architecture of relational DBMSs
- understand the concepts behind transactions and concurrency control
- appreciate query and transaction processing techniques within RDBMSs

- appreciate the past, present and future of database technology

Glossary:

- **DBMS**: DataBase Management System ... software system to support database manipulation
- **RDBMS**: Relational DBMS ... the most popular style of DBMS (refers to underlying data model)
- **SQL**: Structured Query Language ... the ANSI standard language for manipulating RDBMSs

Teaching Strategies

- Lectures : deliver the basic concepts and explain with detailed examples
- Lab Work : help students implement basic database components with real-life database instance
- Consultation: weekly consultation to provide personalized advice to students on their progress in the course.

Teaching Rationale

The learning focus in this course is primarily lectures (theoretical knowledge) and projects (practical knowledge). The course will have an emphasis on problem solving for real applications.

Assessments

Number	Name	Full Mark
1 *	Assignment 1: Data Modeling	10
2 *	Assignment 2: Relational Algebra + Normforms	20
3 *	Assignment 3: DBMS	20
4 **	Project 1	25
5 **	Project 2	25
6	Final Exam	100

Later Submission Penalties:
* : zero marks
** : 10% reduction of your marks for the 1st day, 30% reduction/day for the following days

The final mark is calculated by the harmonic mean:
Final Mark= 2 * (ass1 + ass2 + ass3 + proj1 + proj2) * FinalExam / (ass1 + ass2 + ass3 + proj1 + proj2 + FinalExam)

Course Staff

Name	Office	Phone	E-mail	Role
Xuemin Lin	K17-503	56493*	lxue@cse*	lecturer in charge
Long Yuan	K17-201	56206*	longyuan@cse*	course admin for even weeks
Xubo Wang	K17-201	56206*	xwang@cse*	course admin for odd weeks

Note: You are invited to meet us **in person** during consultation time slots and during the lab periods. You are also welcome to contact us via e-mail if something is urgent.

Lectures Time

This course has a 3-hour lecture per week, held on each Mon 09:00 - 12:00 at Mathews Theatre B (K-D23-203).

Course Resources: Textbooks

Author(s)	Title	Edition	Publisher/Year
Elmasri & Navathe	Fundamentals of Database Systems	6th edition	Addison-Wesley, 2010

Course Resources: References

Author(s)	Title	Edition	Publisher/Year
Jeffery D. Ullman, Jennifer Widom	A First Course in Database Systems	Recent Edition	Prentice Hall
R. Ramakrishan	Database Management Systems	3rd	McGraw-Hill, 2003
D. Maier	The Theory of Relational Databases	1st	Computer Science Press, 1983

Academic Honesty and Plagiarism

Plagiarism is defined as "using the words or ideas of others and presenting them as your own". UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Learning Centre: Plagiarism and Academic Integrity
- MyUNSW: Plagiarism

- CSE: Addendum to UNSW Plagiarism Guidelines
- CSE: Yellow Form (whose terms you have agreed to)

Course Evaluation and Development

This course is evaluated each session using the CATEI system. In this session, we will use more concrete examples to demonstrate difficult concepts.

More Information

For further information on this course, and to keep up to date with any changes, please consult the course web site (frequently):
http://www.cse.unsw.edu.au/~cs3311/

COMP3141 Software System Design and Implementation

Course Overview

This course presents semi-formal and formal methods for the design and implementation phases of software system development. It introduces approaches to testing informed by formal designs, and it discusses trade-offs between static and dynamic approaches to improving software correctness. Throughout the course, the discussed methods are supported by software tools that assist in managing design, implementation, and testing. The course content is illustrated by case studies and practical exercises. Central topics are the use of logical properties and types to inform program design, implementation, validation, and verification.

The course will introduces students to the strongly-typed Haskell programming language. No previous knowledge of Haskell is assumed.

The course web page is at

- http://www.cse.unsw.edu.au/~cs3141/

This is a 6 UoC course. Please see the course web page for a detailed course schedule.

Parallel Teaching

There is no parallel teaching.

Prerequisites/Assumed Knowledge

You need to have successfully completed the core programming, algorithm, and software development courses. You should be a confident coder and be prepared to study the concepts of a new programming language in directed self-study.

Course Prerequisites

COMP1921 or COMP1927

Course Exclusions

None.

Objectives

After completing this course, you should

- understand the how to use logical program properties to characterise aspects of the functional specification of programs.

- understand the difference and trade-offs between static methods (such as formal methods and type systems) and dynamic methods (such as testing) in assisting software design and implementation.
- understand the role of types in program design, implementation, validation, and verification.
- be able to use basic formal methods, strong type systems, and property-based testing to design and implement software while minimising software defects.
- be able to use a variety of tools based on formal specifications of logical properties.

The course exposes students to a mathematically founded approach to specifying and implementing software systems. It develops basic skills required to engineer software with high confidence in the correctness of the final product. The whole course encourages critical examination and analysis of existing solutions.

Staff

The Lecturer-In-Charge is Manuel Chakravarty. The course administrator is Liam O'Connor

Constituents

Lectures

The lectures will introduce you to new material, which is being reinforced and practised in weekly exercises and larger assignments. The course follows no particular textbook, but reading material covering specific topics will be identified throughout the course. Students are required to study reading material as advised during the lecture and/or on the course web page.

There are three hours of lectures each week. The lecture times and locations are the following:

- Tuesday, 11:00-13:00 @ Webst ThA
- Thursday, 10:00-11:00 @ CLB 8

Exercises

Weekly marked exercises start in **Week 3.** There will be 10 exercises, and these exercises will reinforce the material discussed in the lecture. Students have approximately one week to submit a solution. Submitting solutions to the exercises is compulsory, and solutions will be automatically marked. Solutions to exercises will not be accepted late. Automarking results are binding. Exercises will generally not be manually remarked. There are no extensions to exercises.

Assignments

There will be **two** practical assignments. They will be due approximately around Week 6 and Week 11. Students will have between one and two weeks to understand each individual assignment and to develop a solution.

Unless otherwise stated if you wish to submit an assignment late, you may do so, but a late penalty **reducing the maximum available mark** applies to every late assignment. The maximum available mark is reduced by 10% if the assignment is one day late, by 25% if it is 2 days late and by 50% if it is 3 days late. Assignments that are late 4 days or more will be awarded zero marks. So if your assignment is worth 88% and you submit it one day late you still get 88%, but if you submit it two days late you get 75%, three days late 50%, and four days late zero.

Assignment extensions are only awarded for serious and unforeseeable events. Having the flu for a few days, deleting your assignment by mistake, going on holiday, work commitments, and so on do not qualify. Therefore aim to complete your assignments well before the due date in case of last minute illness, and make regular backups of your work.

Assignments are being marked automatically. You need to make sure to follow the instructions closely. Failure to follow the details of the assignment specification is no reason for re-marking, even if small mistakes lead to a substantial loss of marks.

Course philosophy and teaching strategies

The learning focus in this course is primarily on lectures and assignments. While the assignments are graded and contribute to the final mark, their primary purpose is to facilitate learning by hands-on experience.

Assessment

Assessment consists of a practical component and a final examination. The break down of the practical component is as follows:

Assignment 1	20 marks
Assignment 2	20 marks
Weekly Exercises	60 marks

Both the practical component and the final examination are out of 100. The final grade for the course is determined by the **harmonic mean** between the practical component and the final examination.

Exam marks and final marks may be scaled to ensure that the course Pass/Fail boundary and the Distinction/High Distinction boundary reflect a consistent standard from session to session.

Assignments

You can earn varying amounts of marks for each of the two assignments (see above table). Each assignment is individual; i.e., **no team work of any kind is permitted.** Completing and submitting **all** assignments is **compulsory;** i.e., each assignment has a core component and you will not be permitted to pass the course unless you have made a reasonable effort to solve the *core component.* A "reasonable effort" means that there may be bugs in your solution, but you must submit an at least partially working piece of adequately structured code.

Exercises

There will be 10 weekly exercises (each valued at 6 marks). Each exercise is individual; i.e., **no team work of any kind is permitted.** You will not be allowed to pass the course unless you have submitted **at least** 8 of the weekly exercises (the submitted code can be only partially correct).

Final examination

The final exam is a three hour written exam. Requests for a supplementary exam will only be considered where students (a) have completed all other course components to a satisfactory standard, (b) have been absent from the final exam, (c) and have submitted a fully documented request for special consideration to NSQ within three working days of the final exam.

Examination hurdle

To achieve a passing final mark, a student needs to pass the exam. To pass the exam, a student needs to achieve 50% of overall marks awarded in the exam. Any student who fails the exam will automatically fail the entire course. No re-assessment will be awarded in this case.

Assignment and exercise work

Assignments and exercises are an important part of the course. They are an essential way of learning the practical skills you need to acquire. Any plagiarism in assignments or exercises will be **severely punished** and may result in an automatic Fail for the whole course. Read the plagiarism warning below for more details.

For each assignment, you will have *approximately* one week from release of the specification until the submission deadline. The specifications will be posted on the course web page.

Assignment work can be completed on the workstations at UNSW or on a computer at home. Your assignment must properly run on the computers at UNSW; so, test them at UNSW if you develop them at home. Unless otherwise stated, assignments must be submitted on-line from a school terminal using the give command. It is in your best interest

to make regular backup copies of your work and (because of machine loads on deadline days, for example) to complete assignments well before their deadlines. Moreover, the electronic submission system give allows you to submit an assignment multiple times; only the last submission will be marked. We suggest that you submit a version once you have a partially complete solution and repeatedly submit whenever you improved your solution significantly. In particular, make sure that you submit your solution once you have completed the core component of each assignment. The core component of each assignment must be submitted to be able to pass the course.

Texts

There is no textbook for the course.

As tutorial and reference for Haskell, you can use either of these three books:

- Haskell Programming From First Principles by Christopher Allen and Julie Moronuki, pre-release.
- Real World Haskell by Bryan O'Sullivan, Don Stewart, and John Goerzen, O'Reilly Media.
- Learn You a Haskell for Great Good! by Miran Lipovača, No Starch Press.

Both books are available as ebooks.

Further reading material will be announced in the lecture and/or on the course web page.

Getting Help

Questions regarding the course material, assignments, exercises, and general administrative questions should be asked on the course forum (accessible from the course web page), where answers benefit the whole class. Alternatively, approach the lecturer after class.

To discuss matters concerning your personal performance, please send an email to the course account <cs3141@cse.unsw.edu.au>.

For identification purposes, if you wish to send email concerning the course, you must:

1. Send the mail from your CSE or UNSW student account (not from GMail, Yahoo, Bigpond or similar).
2. Include your student id and your full name.

Plagiarism

Many students do not appear to fully understand what is regarded as plagiarism. The university's plagiarism procedure is documented here [https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf] and the CSE Plagiarism policy here [http://www.cse.unsw.edu.au/~chak/plagiarism/plagiarism-guide.html].

All work submitted for assessment must be entirely your own work. We regard unacknowledged copying of material, in whole or part, as an extremely serious offence.

In this course, submission of any work derived from another person, or solely or jointly written by and/or with someone else, without clear and explicit acknowledgement, will be severely punished and may result in automatic failure for the course and a mark of zero for the course. Note that this includes including unreferenced work from books, the internet, etc.

Do not provide or show your assessable work to any other person. Allowing another student to copy from you will, at the very least, result in zero marks for that assessment. If you knowingly provide or show your assessment work to another person for any reason, and work derived from it is subsequently submitted you will be penalised, even if the work was submitted without your knowledge or consent. This will apply even if your work is submitted by a third party unknown to you. You should keep your work private until submissions have closed.

If you are unsure about whether certain activities would constitute plagiarism ask us before engaging in them!

Copying without consent, severe, or second offences will result in automatic failure, exclusion from the university, and possibly other academic discipline.

These are not idle threats, we search the internet and use plagiarism detection software and a range of search engines to hunt for non-original work.

See also the latest version of the Unix Primer and the Yellow Form and the school and the faculty and university plagiarism policies for additional information. If the penalties set out on this page, the Unix Primer, the Yellow Form, the school, faculty, or university plagiarism policies differ for any situation, the more severe penalty applies.

Note that we have experienced cases of plagiarism where the code has been copied from printouts or CDs/USB sticks that have been lost in the lab or stolen from the computer or printer. Generally it is your responsibility to prevent other students from accessing your files, but if you lose work in this way, email your tutor immediately.

Final Examination

The final examination in this course will be held during the end of session examination period; it may examine any material covered in lectures, exercises, assignments, and any reading you have been given.

Special Consideration

Students whose exam performance is affected by serious and unforeseeable events outside their control can apply at the student centre for special consideration. If special consideration is granted you will be able to sit the supplementary exam.

Special consideration does not mean we adjust your marks, it means that we permit you to sit the supplementary examination. If you apply for special consideration after the cut-off date set by the University or after the supplementary exam has been held, then it will not be granted. Special consideration will only be granted where students (a) have completed all other course components to a satisfactory standard, (b) have been absent from the final exam, (c) and have submitted a fully documented request for special consideration to the student centre within three working days of the final exam.

Supplementary Exam

A supplementary examination will be held soon after the results have been released — see the UNSW calendar for details. If you think that you may be eligible for the supplementary examination, make sure you are available around during that time. Be careful not to plan any overseas travel at that time. If you can't attend the supplementary exam you will not be offered a second chance. Supplementary exams will be oral exams.

Check Your Marks

You can inspect the current state of your mark record online at https://cgi.cse.unsw.edu.au/~give/Student/sturec.php

Course Evaluation and Development

This course is being continuously improved and we will conduct a survey at the end of session to obtain feedback on the quality of the various course components. Your participation in the survey will be greatly appreciated. Student feedback over the last years has generally been positive.

COMP3331 Course Details

Course Code	COMP3331
Course Title	Computer Networks and Applications
Units of Credit	6
Course Website	http://cse.unsw.edu.au/~cs3331
Handbook Entry	http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP3331.html

Course Summary

This course is an introductory course on computer networks, aimed at students with a background in computer science / electrical engineering. We will focus on common paradigms and protocols used in present data communication. Through lectures, in-class activities, labs and assignments, you will learn the theory and application of (1) medium access control, congestion control, flow control, and reliable transmission, (2) addressing and naming, (3) routing and switching, (4) widely used protocols such as Ethernet, IP, TCP, UDP, HTTP, etc. (5) security threats and common defensive techniques, and (6) special purpose networks such as content delivery networks, peer-to-peer networks and wireless networks. This is a combined undergraduate and postgraduate course. The written exams for the postgraduate students will contain some questions, which are different from the undergraduate exam, and will more challenging.

Course Timetable

There will be 3 hours of lectures every week: (i) 2-hour lecture on Monday 16:00 - 18:00 in Ritchie Theatre and (ii) 1-hour lecture on Wednesday 14:00 - 15:00 in Rex Vowels Theatre. There will be 2-hour labs during 9 weeks (starting in Week 2). The detailed lab schedule will be posted on lab exercises page. The detailed course timetable is available here.

Course Aims

To provide an in-depth introduction to a wide range of topics in the field of computer networks including the Internet. To get a hands-on understanding of the working on network protocols. To gain expertise in network programming, designing and implementing network protocols, evaluating network performance and problem-solving skills. To build the necessary foundational knowledge required in subsequent networking courses (COMP4335-4337, COMP9332-9337).

Student Learning Outcomes

After completing this course, students will:

1. have a working knowledge of computer networks, and will be able to demonstrate their knowledge both by describing aspects of the topics and by solving problems related to the topics
2. have a solid understanding of the current architecture of the Internet and the entities involved in its operations
3. be able to identify soundness or potential flaws in proposed protocols
4. be equipped with the necessary skills to design networked applications and protocols
5. implement and write protocols and applications in C, Java or Python
6. analyse and evaluate the performance of computer networks
7. be able to capture and analyse network traffic
8. be able to understand and explain security and ethical issues in computer networking

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

This course contributes to the development of the following graduate capabilities:

Graduate Capability	Acquired in
scholarship: understanding of their discipline in its interdisciplinary context	lectures, labs, assignments
scholarship: capable of independent and collaborative enquiry	labs, assignments
scholarship: rigorous in their analysis, critique, and reflection	lectures, labs, exams, sample problems
scholarship: able to apply their knowledge and skills to solving problems	labs, assignments, exams, sample problems
scholarship: capable of effective communication	labs, assignments, lectures, exams
scholarship: information literate	all aspects of the course
scholarship: digitally literate	all aspects of the course
leadership: collaborative team workers	labs, assignments
professionalism: capable of independent, self-directed practice	all aspects of the course
professionalism: capable of lifelong learning	all aspects of the course
professionalism: capable of operating within an agreed Code of Practice	labs, assignments
global citizens: culturally aware and capable of respecting diversity and acting in socially just/responsible ways	labs, course forums

Assumed Knowledge

Before commencing this course, students should:

- have a good understanding of data structures and algorithms, basic probability theory.
- be able to write working programs in C, Java or Python. The course will include programming assignments and labs.

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

These skills are assumed to have been acquired in the courses: COMP1921 or COMP1927 or MTRN3530 (for undergraduates) and COMP9024 (for postgraduates)

Teaching Rationale

This course takes a top-down approach to teaching computer networks. The rationale behind this is that most students have first-hand experience using applications running over the Internet. This allows them to relate to each layer of protocol stack as we travel down the layers. Once they are committed, they participate in appropriate cognitive aspects such as learning the details with a focus to understand them. Students get mentally prepared to answer questions where very often there is no single answer or the answers can be unexpected. This results in deep learning and gives students a sense of accomplishment and confidence.

Learning will be largely facilitated through the delivery of lectures. The hands-on laboratories will provide an opportunity to gain deeper understanding of the concepts discussed in the lectures. The sample problems, homework problem set and tutorials will help in the development of problem-solving skills and in preparing for the exams. The programming assignments are mainly geared to allow students to gain familiarity with basic network programming and designing network protocols.

Teaching Strategies

- Lectures: introduce theory and concept and demonstrate how they apply in practice
- Lab Work: reinforce concepts taught in lectures by conducting hands-on experiments and analyse network performance
- Assignments: allow students to design and implement network protocols and evaluate network performance
- Sample Problems: allow students to solve problems based on content from lectures, develop problem-solving skills, assist with exam preparation
- Consultations and Course Forum: allow students an opportunity to ask questions and seek help.

Assessment

There will be four assessment components as listed below:

Component	Weight
Lab Exercises	20%
Programming Assignments	25%

Component	Weight
Mid-semester Exam	20%
Final Exam	35%

To pass the course a student MUST receive at least 40% marks in the final exam. The following formula outlines precisely how the final mark will be computed:

```
lab = marks for lab exercises (scaled to 20)
assign = marks for the two programming assignments (scaled to 25)
midExam = mark for the mid-semester exam (out of 20 marks)
finalExam = mark for the final exam (out of 35 marks)
mark = lab + assign + midExam + finalExam
grade = HD|DN|CR|PS if mark >= 50 && finalExam >= 14
      = FL          if mark < 50 || finalExam < 14
```

Academic Honesty and Plagiarism

Plagiarism is defined as *using the words or ideas of others and presenting them as your own* . UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Learning Centre: Plagiarism and Academic Integrity
- MyUNSW: Plagiarism and Academic Misconduct
- CSE: Addendum to UNSW Plagiarism Guidelines
- CSE: Yellow Form (whose terms you have agreed to)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism.

Course Schedule

The following table lists the tentative weekly schedule. Students will be informed of any changes during the lecture and by announcements on the notices page.

Week	Lecture Dates	Lecture Topics	Labs	Assessment Tasks
1	25 & 27 July	Course Logistics Introduction: What is a network made of? How is it shared? How is it organised?	No Lab	

Week	Lecture Dates	Lecture Topics	Labs	Assessment Tasks
		How does communication happen?		
2	1 & 3 August	Introduction: How do we evaluate a network? How did the Internet come about? Application Layer: Client-server and P2P architectures The Web & HTTP E-mail	Lab 1	
3	8 & 10 August	Application Layer: Domain Name Service (DNS) Content Distribution Networks Socket Programming	Lab 2	Assignment 1 Released
4	15 & 17 August	Application Layer: Peer-to-Peer Networks and DHT Transport Layer: Transport services Multiplexing & Demultiplexing UDP	Lab 3	
5	22 & 24 August	Transport Layer: Principles of reliable data delivery TCP	Tutorial 1 for Exam Prep	
6	29 & 31 August	Transport Layer: Congestion control Fairness	No LAB	**Mid-semester Exam on 29th August**
7	5 & 7 September	Network Layer: Network services Datagram vs Virtual Circuits IP Addressing	Lab 4	
8	12 & 14	Network Layer:	Lab 5	Assignment 1 Due

Week	Lecture Dates	Lecture Topics	Labs	Assessment Tasks
	September	Router internals Routing algorithms Link Layer: Services Error detection		Assignment 2 Released
9	19 & 21 September	Link Layer: medium access control link layer addressing Ethernet switches Wireless and Mobile Networks Wireless characteristics CDMA	Lab 6	
		Mid-semester Break		
10	5 October 3 October is public holiday	Wireless and Mobile 802.11 Network mobility Network Security Basic Cryptography	Remedial Marking for Assignment 1 (optional)	
11	10 & 12 October	Network Security Message integrity & Digital signatures Authentication Secure E-mail Firewalls SSL	Lab 7	
12	17 & 19 October	Optional (one of the following): Multimedia Networking SDN, Virtualisation Internet of Things Problem Solving and Revision	Tutorial 2 for Exam Prep	
13	24 October	Remedial Lecture if necessary		Assignment 2 Due
Exam	4th November -	Exam Period		**Final Exam**

Week	Lecture Dates	Lecture Topics	Labs	Assessment Tasks
Period	22nd November			

Resources for Students

Course Textbook:

- Computer Networking - A Top-Down Approach Featuring the Internet, J. Kurose and K. Ross, Addition Wesley , Sixth Edition, 2012.

Reference Texts:

- Unix Network Programming Volume 1 - Networking APIs: Sockets and XTI, W. Richard Stevens, Prentice Hall, Second Edition, 1998.
- Java Network Programming, E. R. Harold, O'Reilly, Third Edition, 2004.
- Learning Python, Mark Lutz, O'Reilly, Fifth Edition, 2013.
- Computer Networks: A Systems Approach, Larry Peterson and Bruce Davie, Morgan Kaufmann, Fifth Edition, 2011.
- Introduction to Computer Networks and Cybersecurity, John Wu and J. David Irwin, CRC Press, 2013.
- Computer Networks, Andrew Tanenbaum and David Wetherall, Fifth Edition, Pearson, 2010.

Links to additional reading material will be available on the lecture notes page.
Software:
For the labs, we will be using several Unix-based network utility programs. The purpose of these programs and information on how to use them will be provided in the lab handouts. We will also use a packet sniffing tool called Wireshark , which has been widely deployed on CSE machines. In addition, we will also use ns-2 , a widely used network simulator for a few labs. Ns-2 is installed on the CSE lab machines. The simulator is written in C++. However, it uses OTcl as its command and configuration interface. In the lab exercises, we will use scripts written in OTcl. We will provide the OTcl scripts for the lab exercises . You will expected to run the scripts, make some minor changes in the scripts, and analyse certain performance metrics. You will not be required to write C++ code. Detailed resources for all tools used will be made available on the lab exercises page.

Programming assignments are expected to be developed in C, Java or Python. Students are assumed to have sufficient expertise in one of these programming languages. Links to network programming in C, Java and Python will be available under the assignments link of the course webpage. Sample code demonstrating a simple client/server application will also be supplied as a starting point for students.

Course Evaluation and Development

In the previous offering of this courses, we had introduced a new set of labs using a network emulator called Mininet which provides the ability to simulate large network topologies and observe protocol behaviour and evaluate network performance using these topologies. Unfortunately, several students had problems getting Mininet to run. Moreover, the lab exercises provided to be difficult to configure and run. As a result, we have scraped the Mininet labs in this semester and introduced a new set of labs which we believe will be easy to run and still very instructive.

Supplementary Examination/Re-assessment

You can view the CSE Supplementary exam/Special consideration policy at the following link: https://www.cse.unsw.edu.au/about-us/organisational-structure/student-services/policies/yellow-form/index.html#dates

Re-Assessment Policy: Due care is taken to mark all assessment components fairly and appropriately. Therefore, it is unlikely that marks will be changed after a re-assessment. You should contact the LiC to discuss this further. However, students who still feel that the mark they received does not reflect their performance have the right to apply for re-assessment. Students MUST apply for re-assessment via Student Central within 15 days after notification of results of assessment. Please note that re-assessment or re-marking of a piece of work may result in marks to go up or down. Further details can be found from UNSW student guide at following site: https://student.unsw.edu.au/results

COMP4920/SENG4920 Management and Ethics

Contact Details

Name	Role	E-Mail	Phone
Wayne Wobcke	Lecturer in charge	w.wobcke@unsw.edu.au	9385 6475
Bruno Gaeta	Lecturer	bgaeta@unsw.edu.au	9385 7213

Course Details

Units of Credit: 6

Web Site: http://www.cse.unsw.edu.au/~cs4920/

Timetable:

Lecture Time	Room	Lecturers
Wed 12.00-2.00	Rex Vowels	Wayne Wobcke, Bruno Gaeta

Seminar Time(**)	Room	Facilitator	E-Mail
Mon 10.00-12.00	Square House 208	Sandeepa Kannangara	s.kannangara@unsw.edu.au
Mon 12.00-2.00	Square House 208	Bruno Gaeta	bgaeta@unsw.edu.au
Tue 10.00-12.00	Mathews 123	Rob Everest	robertce@cse.unsw.edu.au
Tue 12.00-2.00(*)	Mathews 125	Wayne Wobcke	w.wobcke@unsw.edu.au
Tue 3.00-5.00(*)	Square House 207	Wayne Wobcke	w.wobcke@unsw.edu.au
Wed 10.00-12.00	Quad G054	Sandeepa Kannangara	s.kannangara@unsw.edu.au
Wed 2.00-4.00	Mathews 226	Rob Everest	robertce@cse.unsw.edu.au
Wed 4.00-6.00	Mathews 309	John Calvo-Martinez	jcalvo@cse.unsw.edu.au
Thu 9.00-11.00	Ainsworth G01	John Calvo-Martinez	jcalvo@cse.unsw.edu.au

Note: (*) Class for SENG students only
Note: (**) Seminars may be cancelled depending on enrolments

Preamble

COMP4920 is taken by two groups of students: **SENG students** who have completed software project management as part of Software Engineering workshops, and **COMP students** (Computer Science, Computer Engineering and Bioinformatics students) who have not previously studied software project management. Lectures are common, but seminars and assessment differ between the two groups. Differences are highlighted in blue.

Course Aims

COMP4920 covers practical aspects of both *software project management* and *professional issues and ethics*, and as such is critical preparation for graduates about to enter the workforce, in addition to being essential for accreditation of the Software Engineering, Computer Science, Computer Engineering and Bioinformatics degree programmes. **Students enrolling should be in the final year of study or nearing completion of their degree.**

There are two specific themes and objectives.

- *Software Project Management.* To gain practical experience in all phases of the planning and execution of a software project, including requirements scoping, choice of software process methodology, project planning and scheduling, teamwork and communication, and risk and change management. SENG students study project management only in seminars: there is no practical component.
- *Professional Issues and Ethics.* To appreciate the responsibilities of a professional software engineer and understand the ethical dimensions of the IT industry as applied to specific issues such as software correctness, privacy and security, intellectual property and legal obligations of IT practitioners. SENG students study professional issues to a greater depth and breadth than COMP students.

Student Learning Outcomes

On successfully completing this course, students should be able to:

- *Software Project Management.* (COMP students only)
    - Understand the range of activities involved in a large software project.
    - Be able to plan and successfully execute a team-based software project.
    - Take on different roles in a team to contribute to team success.
    - Understand and appropriately apply software engineering processes.
    - Understand the importance of teamwork and communication in a software project.
- *Professional Issues and Ethics.*
    - Understand the responsibilities of a professional software engineer.
    - Appreciate and apply ethical frameworks to make professional judgements.
    - Develop critical thinking in relation to professional issues.
    - Gain in-depth understanding of several topical professional issues.
    - Appreciate different ways of managing intellectual property.
    - Understand the societal context of technology developments.
    - Improve communication skills needed to present reasoned arguments.

This course contributes to the following UNSW graduate attributes.

- *The skills involved in scholarly enquiry.* The course emphasizes professional codes of ethics and conduct that oblige engineers to keep up to date and rigorously apply the latest methods and technology.

- *The capacity for analytical and critical thinking and for creative problem solving.* Critical thinking is developed through analysis of professional issues and case studies in seminars and through writing an analytical essay involving in-depth research on a topical issue.
- *Capacity for enterprise, initiative and creativity.* The course covers management of intellectual property and innovation including aspects specific to the software industry, such as software patents and open source software.
- *An appreciation and respect for diversity.* Societal benefits and drawbacks of software engineering are discussed explicitly.
- *Skills required for collaborative and multidisciplinary work.* The software project requires the ability to work within technical teams in the development of a product that requires teamwork and management of a project. (COMP students only)
- *Respect for ethical practice and social responsibility.* Explicit discussion of ethical theories and professional codes provides students with frameworks to make ethical judgements and knowledge of what society expects of professional engineers.
- *Skills of effective communication.* Effective communication skills are encouraged through preparation and delivery of a student-led seminar and through writing a critical essay on a topical issue.

Assumed Knowledge

Students are assumed to be in their final year of study (or nearing graduation) **and** completed around half of their Stage 3 or 4 courses, so are assumed to have reasonable knowledge and maturity in Software Engineering, Computer Science, Computer Engineering or Bioinformatics.

COMP students only: Students are assumed to be competent computer programmers with knowledge of agile software processes and experience of working in agile teams based on the Scrum methodology. The level of experience is assumed to be sufficient to undertake (from conception and planning to completion) a moderately large software development project. **Note that seminar facilitators play the role of clients, and cannot provide technical advice on programming languages or platforms.**

Teaching Rationale

COMP4920 is an important course to help students become "job ready". The course covers both software project management from a practical point of view, and professional issues and ethics as related to the IT industry. Students benefit by having the opportunity to interact with industry experts, hence **attendance at lectures is very important**, especially as the course focuses on "soft skills".

For *software project management*, the teaching philosophy is that much of this knowledge cannot be "taught" but rather is gained through experience and reflection. Hence the approach taken in this course is that students learn about software project management through working in and managing a team-based software project, encompassing all phases of the project from requirements scoping, application of a software process methodology, project planning and scheduling, teamwork and communication, risk management and

change management. Students develop a project plan in the first half of semester **and will be held to that plan for the second half of semester** (subject to any agreed modifications).

For *professional issues and ethics*, teaching is based on seminar-style discussion groups, encouraging students to express their ideas and form their own judgements on specific issues relating to the IT industry on the basis of rational arguments. Seminars also promote team organization and presentation skills through a team-based student-led seminar. Critical thinking is developed through researching and writing an in-depth analytical essay on a specific professional issue of relevance to the industry. SENG students also conduct a debate and have an essay-style written exam.

Time management is an important aspect of this course. It is expected that each student attends **all** lectures and seminars, prepares for each seminar by reading the relevant material **in advance**, contributes actively to seminar discussion, and spends roughly 10-15 hours on the student-led seminar and roughly 15-20 hours on the essay and 15-20 hours on the lecture summaries.

SENG students only: Students should spend 10-15 hours on the debate and a further 25-35 hours on exam preparation.

COMP students only: The software project is a major component of the course and should take 40-50 hours **per person** for planning and execution.

Teaching Strategies

The course has a mixture of guest lectures on topics of interest and seminars focusing on particular professional issues, case studies and the software project.

Lectures provide an overview of one particular aspect of software project management or a professional or ethical issue. Guest lecturers are able to provide expertise in a variety of areas and the lectures provide an essential foundation to apply in seminars and essays. **Note that due to the commercially sensitive nature of the material, it is not possible to provide lecture recordings.**

Seminars provide students an opportunity for more in-depth discussion on particular topics and, in student-led seminars, enable students to develop skills in expression, critical analysis and presentation.

Essays enable students to study in-depth a topic of interest and promotes the development of critical thinking and analytical abilities and written communication skills.

Software projects are the means through which students develop an understanding of software project management and the ability to apply software processes in all phases of planning and executing a software project.

Assessment

The assessment for this course consists of the following weighted components.

- Seminar participation (10%)
- Student-led seminar (10%)
- Essay (20%)
- Lecture summaries (10%)

SENG students only:

- Debate (10%)
- Written examination (40%)

COMP students only:

- Software project plan – presentation and document (20%)
- Software product (30%)

The student-led seminar and debate combine an individual and team mark, and in addition include a peer assessment component. Seminar participation and the essay, lecture summaries and written examination are assessed individually.

The *seminar participation* mark is based on *active* and *relevant* contributions over all seminars (including student-led seminars), which requires attendance at all lectures and reading the seminar material **in advance**. It is not an attendance mark. See the UNSW Teaching website for more details on why and how seminar participation is assessed.

The *student-led seminar* mark is based on coverage of the chosen topic, including identification of relevant ethical issues, presentation style and engagement of the audience in discussion.

The *essay* mark is based on the originality and depth of ethical analysis applied to the chosen topic, drawing on the main ethical theories discussed in the course, and on the quality of the writing and appropriate use of evidence in developing a reasoned argument.

The *lecture summaries*, due in Week 13, should consist of summaries *and short reflections* on any 5 guest lectures relating to professional issues and ethics (i.e. not software processes). Each summary (at most 1 page) should contain an overview of the main points of the lecture, identify a professional/ethical issue discussed in the lecture, and include a short reflection on how that issue can be addressed by applying ethical reasoning.

The assessment of the *software project* includes the presentation and written submission of a project plan (in Week 8) and a presentation/demonstration of the final system (in Week 13). Two seminars will be devoted to the informal presentation and class discussion of the projects (sprint reviews) to enable the group to provide feedback on the progress of each project. The mark for the software product consists of an individual component and a team component; each team member generally receives the same mark for the team component.

The individual mark will in part be based on a project diary shown to the facilitator at various intervals. **Part of the assessment is based on project management and teamwork, including the appropriate use of project management tools.**

The final mark for the course is determined by adding together these component marks according to the above weighting to give a result out of 100, which is subject to further scaling.

Late submission policy for assignments: Assignments (project plan document, essay and lecture summaries) submitted late are subject to the penalty that the mark reduces by 20% per (calendar) day late, for up to three days, after which a mark of 0 is received. Projects submitted late will receive a 0 mark.

Academic Honesty and Plagiarism

UNSW has instituted severe penalties for academic plagiarism. Therefore it is important for students to understand what is acceptable and unacceptable in the presentation of work for assessment.

You should **carefully** read the UNSW policy on academic integrity and plagiarism. Note, in particular, that *copying* (taking ideas and/or text from other students or the Internet and presenting them as your own), and *collusion* (working together on an assignment, or sharing parts of assignment solutions) are forms of plagiarism. That is, giving your assignment solution to another student counts as collusion, regardless of the originality of your work. In COMP4920, this applies particularly to the essay, which must be written in your own words, and with properly cited sources. General expectations of students at UNSW, and the procedures for handling student misconduct (including plagiarism), are set out in theUNSW student code.

In essence, as applied to COMP4920, copying or sharing material from the Internet such as slides, text or program code (that is, without proper citation) counts as plagiarism and is unacceptable. In a student-led seminar, essay or debate, all material must be in the student's own words (except quotations, which should be kept to a minimum and must be clearly identified as such). References must be from primary sources (i.e. do not cite *Wikipedia* articles or the like). Essays will be run through *turnitin* for plagiarism detection. In the software project, sharing code *amongst team members* is acceptable, but sharing code between (members of) different teams is unacceptable.

The penalties for plagiarism range from receiving 0 marks for the assignment, through receiving a mark of 00 FL for the course, to expulsion from UNSW (for repeat offenders). The school maintains a register of students with confirmed plagiarism offences. Note that allowing someone else to copy your work counts as academic misconduct, and makes you liable to a penalty, even if you can prove that the work is yours originally.

Course Schedule

The following is the rough sequence of lecture topics by week. Broadly speaking, there are two lectures on each of (i) ethics, (ii) agile methods, (iii) legal perspectives, (iv) software licensing, and (v) business perspectives. However, as most lectures are given by guest lecturers, this schedule is subject to change. More details will be provided during the semester as the lectures are confirmed.

Week	Topic
1	Introduction to Project Management and Ethics
2	Theoretical Underpinnings of Ethics
3	Moral Reasoning and Professional Ethics
4	Agile Software Processes in Practice
5	Legal Perspectives on System Development
6	Agile Product Management and User Experience
7	Data Privacy and Uberveillance
8	Intellectual Property and Software Patents
9	Open Source Software
10	Innovation and Entrepreneurship
11	Employment Conditions and Contracts

Resources for Students

References

Reference material for lectures and seminars will be provided on the course web pages throughout the semester. There is no set textbook for the course, however some of the introductory lecture on software project management uses material from the following book, which is an excellent reference for software engineering generally (though not specifically for agile methods).

> Sommerville, I. *Software Engineering.* Tenth Edition. Pearson Education, Upper Saddle River, NJ, 2015.

Course Evaluation and Development

Computer Science and Engineering courses are evaluated by student survey each time they are taught. The survey includes standard questions asked of all comparable courses so that it is possible to compare a course with other relevant UNSW courses, and also includes space for free-form comments. Survey responses are anonymous. The completed survey forms are analysed statistically by someone independent of the course staff, and the results, including free-form comments, are made available to the lecturer in charge *after* grades have been reported and released.

For COMP students, COMP4920 was offered for the first time in 2012 as a replacement for COMP3711 Software Project Management (taught by the School of Information Systems, Technology and Management) and a previous 3 unit course COMP2920 Professional Issues and Ethics. Feedback from these previous courses is primarily that the offering of COMP3711 treats project management at too abstract a level for Computer Science and Engineering students, which is the reason for the more practical approach to software project management taken in COMP4920. For SENG students, COMP4920 replaces and largely builds on the previous course SENG4921 Professional Issues and Ethics.

Students are generally satisfied with the current course. However, a few students commented in 2015 that the requirements for the lecture summaries were not clear. To address this concern, a more precise template for lecture summaries has been provided.

ISTM BIS Curriculum Review (June 2016)

Core Courses

INFS1602 Digital Transformation in Business

This is a foundational (Level 1) Information Systems (IS) course that introduces students to the use of IS in business and society. As an overarching theme, INFS1602 examines the issues and management of information systems in relation to human behaviour and their consequences. Through this course, students will learn to appreciate existing and emerging technologies affecting businesses, business relationships and their products and services. In taking this course, students will be provided with tasks and assignments that will aid in refining their professional business skills and the ability to evaluate the value of technology to businesses. This includes communication and group-work skills, time management and research skills.

The topics that are covered in INFS1602 include understanding the role of Information Systems and IS Professionals in Global Business, the relationship between Information Systems, Organisations, and Strategy, the Dominant Business Models Enabled by the Internet, and the emergence of Web 2.0 Technology. The course also touches on popular enterprise-level information systems such as Enterprise Systems, Supply Chain and Customer Relationship Management Systems and the emergence of Business Intelligence in Supporting Organisation Decision Making. The course also involves the discussion of the considerations behind the Acquisition and Building of Information Systems and the issues common to the Management of Information Systems Projects. Lastly, the course addresses the need to secure the Information Systems and the potential Ethical and Social Issues faced by businesses in relation to their use of Information Systems.

INFS1603 Introduction of Business Databases

This is a foundational (Level 1) Information Systems (IS) course that introduces students to the concepts, techniques and technologies relevant for creating and managing business databases. It will explain the major components of information systems, which are important to capturing, transmitting, storing, retrieving, manipulating and displaying information used in business processes. Through this course, students will be exposed to the fundamental knowledge on business databases, which are foundational for many advanced courses. Students will be given tasks and assignments to help them acquire the ability to create and manage business databases.

The topics that are covered in INFS1603 include Entity Relationship Modeling, Relational Modeling, and Normalisation. The course also introduces the topics related to creating and

managing business databases, such as SQL and PL/SQL. The course ends with the discussion of Object-Oriented Modeling and Relational Algebra.

INFS1609 Fundamentals of Business Programming

This is a foundational (Level 1) Information Systems (IS) course that introduces students to application programming. The course provides a first step towards learning the principle of object-oriented programming through the Java programming language. Programming refers to the development of software, which is also called a program. Essentially, software contains the instructions that tell computerised devices what to do. In lectures, students will be introduced to the theoretical component of the course, learning fundamental programming concepts. During weekly workshop tutorials, students will engage in the practical component of the course, learning how to write code using the BlueJ integrated development environment (IDE).

The topics that are covered in INFS1609 introduce students to the fundamentals of Java programming. This begins with an overview of data types and methods before introducing students to small problem-solving exercises that require the use of conditional statements, loops and Arrays (including Multi-Dimensional Arrays and Array Lists). Students are then introduced to the topics of modular programming, testing and debugging (using JUNIT). Finally, having gained a general understanding of these concepts, students further explore the principles of object-oriented programming, including objects, classes, abstraction, polymorphism, inheritance and encapsulation.

INFS2603 Business Analysis Using Design Thinking

This is a Level 2 Information Systems (IS) course that continues the students' study of IS by furthering their knowledge and skills in relation to the analysis and design of business information systems. This course introduces students to the contemporary method of design thinking, and the object-oriented approach to understand and solve business problems. In lectures, students will study a range of methods, tools and techniques used in planning, analyzing, designing and implementing business relevant systems. During weekly practical workshops, students will get the chance to apply design thinking principles to understand and solve real-world cases, utilizing their conceptual knowledge.

The topics that are covered in INFS2603 include understanding design thinking principles, methods and process. The course will then cover topics related to project management: including project plans, work plans and feasibility analysis, and developing analysis strategy: including requirements determination using design thinking methods, business process modelling, structural and behavioral modelling. Once the students have an understanding of how to develop project plans and system proposals, topics related to developing system specification will be covered, including: architecture design, interface design, database design and program design. The last few topics will cover the installation phase of systems

including: change management plan, test plan, training plan, support plan and migration plan.

INFS2605 Intermediate Business Programming

This is a Level 2 Information Systems (IS) course that continues the students' study of IS by furthering their knowledge and skills in relation to business application programming. The course continues the study of Java programming from INFS1609 (Fundamentals of Business Programming) and examines contemporary approaches to software development. In lectures, students will study a range of topics from advanced Java concepts, software development frameworks and practices, to user experience and design. During weekly workshop tutorials, students will engage in the practical component of the course and problem-solving exercises through the development of Java applications using the Netbeans Integrated Development Environment (IDE).

The topics that are covered in INFS2605 build on those introduced in INFS1609, providing students with a thorough review of software development processes and object-oriented programming principles. Students will then expand their Java skills and knowledge through the study of Model View Controller (MVC) architecture, event-driven programming and Graphical User Interfaces (GUI). Specifically, the course introduces students to the development of JavaFX GUI applications, using Scenebuilder. Building on this, students are then provided with an overview of exception handling and taught how to develop basic database applications using **Java** Database Connectivity (**JDBC**), an application programming interface (API), which defines how a client may access a database. This concludes with an introduction to API's that facilitate the development of reporting functionalities (e.g. exporting data to excel) from database applications.

INFS2608 Database Management & Big Data Infrastructure

INFS2608 is a Level 2 Information Systems (IS) course that continues students' study of IS by covering various advanced topics pertinent to database management, which includes both relational and analytical data system infrastructure. It will explain advanced concepts used to design and manage relational and analytical data system infrastructure. Through this course, students will learn to evaluate issues associated with enterprise database management and business data analytics, such as data quality and security. In taking this course, students will be provided with tasks and assignments that will aid in refining their ability to evaluate the value of data focused infrastructures.

The topics that are covered in INFS2608 include Database Architectures and the Web, Transaction Management and Enterprise Database Security. The course also covers emerging database infrastructure and analytics infrastructure, such as Data Warehouse Concepts and Infrastructure, Data Quality in Big Database Systems, and Big Data Business

Analytics Infrastructure Design. The course ends with the discussion of leading big data analytics infrastructure, such as Hadoop.

INFS2621 Enterprise Systems

This is a Level 2 Information Systems (IS) course that continues the students' study of IS by introducing students to Enterprise Systems (often referred to as ERP systems), specifically, how they can be used by organisations to run their operations more efficiently and effectively. The course will present the evolution, components and architecture of Enterprise Systems and help students to understand the benefits and drawbacks of implementing such systems and how they can assist organisations to improve their overall efficiency. Furthermore, the course aims to help students understand the impact of Enterprise Systems on managing complex organisational processes including procurement, order fulfilment and logistics within a supply chain. In lectures, students will learn about the challenges associated with implementing Enterprise Systems for managing complex supply chains and their impact on organisations. Students will learn to develop models for selected business process including procurement, fulfilment and logistics. Students will learn to communicate and assess an organisation's readiness for enterprise system implementation with a professional approach in written form, and describe the selection, acquisition and implementation of Enterprise Systems. In workshops, students will learn to complete a set of common business processes including procurement, fulfilment and logistics, using an Enterprise Systems package-SAP ERP ECC6. Students will also learn about the scope of common Enterprise Systems modules (e.g., MM, SD, FI and CO), and other extended Enterprise Systems solutions such as SAP HANA, SAP ERP Simulation Games. Students will develop an understanding of the issues in systems use of an Enterprise Systems package (e.g. SAP) to support business operations and decision-making through design thinking and play.

INFS3603 Introduction to Business Analytics

This is a level 3 Information Systems (IS) course and a foundational course in Business Analytics (BA). This course provides students an understanding of business needs and technology trends driving investment in business analytics and big data technologies. The course also presents the fundamentals of implementing and managing business analytics in organisations. In lectures, students will learn business analytics methods and tools as well as the challenges associated with implementing business analytics projects. Through real-world case studies, students will develop their understanding of the applications of business analytics as well as the social and ethical implications of business analytics. Students will also improve their critical thinking, problem solving, research, communication, and team-working skills through their group assignments.

Topics that are covered in this course include: decision making process; business analytics concepts, methods, and frameworks; frameworks for putting analytics to work; the governance, oversight and business value gained from business analytics within organisations; ethical and social implications of business analytics; and future directions for business analytics.

INFS3604 Business Process Management

This is a level 3 Information Systems (IS) course that considers the management of business processes and their continuous improvement including identification, analysis and redesign. A business process is a set of related activities that jointly realise a business goal in an organisational and technical environment. These processes take place in a single organisation but may need to interact with processes within other organisations. Business process management (BPM) is concerned with the concepts, methods, and techniques that support the design, improvement, management, configuration, enactment, and analysis of business processes. Modelling is a significant component of the course enabling explicit representation of processes – once they are defined, processes can be analysed, improved, and enacted. Software in the form of business process management systems can be used to manage business processes. By taking this course students will be able to understand business process from a management and process analyst perspective, learn tools, analytical frameworks and general principles for managing and improving business processes. The course will incorporate a laboratory component using BPM software.

INFS3617 Networking & Cyber Security

This is a level 3 Information Systems (IS) course that continue students' study in IS by further developing their knowledge and understanding in information technology infrastructure and security in a business environment. The course will provide students with a learning experience which encourages participation, building of ideas in regard to current issues in business data networks, telecommunications and infrastructure along with overall discussions through the topics. The course has a technical component in which students gain practical knowledge and experience in networking and IS security techniques.

Topics to be covered in this course include inter-networked data communications and distributed data processing. Topics covered include, the business imperatives for distributed systems, systems architectural design (client/server; distributed processing, etc) layered architecture models (TCP/IP, OSI, etc), key network models and technologies, security issues related to architecture, design and technology, network configuration and management techniques.

INFS3634 Mobile Applications Development

This is a Level 3 Information Systems (IS) course that continues students' study of IS by furthering their knowledge and skills in relation to mobile application programming. Continuing from INFS2605 (Business Application Programming), this course focuses on the development of software applications using the Android mobile platform. In

lectures, students will be provided with an overview of mobile programming concepts and tools, and engage in case studies with regards to mobile App development and the current mobile market. During the weekly practical workshops, students will use the Android Studio Integrated development environment (IDE) in learning how to design and develop a range of mobile applications. Students will be required to evaluate the quality of their own, and peers', coding solutions. Students will also research and analyze current trends in the mobile market and present a portfolio of their design work at the end of the course.

INFS3605 Information Systems Innovation & Transformation

This is a Level 3 Information Systems (IS) course that concludes the students' study of IS through the application, integration and synthesis of students' knowledge from previous IS courses. Specifically, INFS3605 is the 'capstone' IS course that is centrally organised around practical, experiential, group software projects. Throughout the course, students will apply programming knowledge and teamwork skills learnt in previous courses in an applied and integrated fashion. The course begins with student groups brainstorming and developing their software project ideas and then gathering requirements. Following this, student groups engage in an iterative development process in designing and refining their software application. Specifically, students will use the agile scrum framework in developing their software project, working in two-week sprints/iterations. This hands-on project course takes a blended approach to learning, mixing online content provided through the school's e-learning platform (Moodle) with weekly flipped workshop sessions. Throughout the course, students will perform various roles (including scrum master and product owner) and ceremonies (including sprint planning, stand-up sessions, sprint reviews, sprint retrospectives, and backlog refinement), as well as utilise a number of a tools (such as kanban boards, burndown charts and planning poker).

INFS3605 does not introduce 'new topics' to students. Instead, this capstone project course requires students to apply, integrate and build upon existing knowledge and skills learnt in previous IS courses. In particular, the course requires students to perform as agile scrum teams and develop complex software applications in an iterative and incremental manner.

Elective Courses

INFS2631 Innovation and Technology Management

This is a level 2 multi-disciplinary course at the intersection of information systems, entrepreneurship and operations management. The course aims to develop students' conceptual knowledge and practical skills regarding managing technological innovation through various phases of the innovation process. The course emphasizes the role of crowdsourcing, crowdfunding, social media and social networks in developing, driving, and managing innovations.

INFS3632 Service and Quality Management

This is a Level 3 Information Systems (IS) course that introduces students to the key concepts in managing service operations and quality management. This presents an in-depth coverage of topics crucial to the effective and efficient operation of a service system. In lectures, students will learn the "state of the art" of process management of service firms and the opportunities provided by information technology and data analytics in enhancing their competitiveness. Students will be engaged in simulations, where they can apply the concepts learned in the class to real-world settings and learn how to manage process variabilities and quality control. In a project which involves conducting a walk-through-audit to a real company of the students' choice, they will learn how to implement a service business to meet customer satisfaction.

In INFS3632 two main areas in services are covered: process management and quality management. In service process management, the topics covered include the introduction of the service economy, service strategy, and how to develop new services. From studying customer service encounters, this course will look into the design of supporting facilities and service processes. This course will also cover process analysis techniques and how to manage process variability and waiting lines. In the area of service quality management, this course will cover total quality management (TQM), statistical quality control, process improvement, six-sigma and lean operations.

INFS3830 Social Media and Analytics

This is a Level 3 Information Systems (IS) course that continues students' study of strategies to create and extract value from social media. In particular, the course will focus on the power of social media to influence, the effect of social media on operational matters, social media metrics and strategic aspects of social media analytics. The course will help students better understand various social media technologies, platforms and analytics, and how they are able to be applied in a business context. The course will present the purpose, function and design of social media platforms and help students to understand the benefits and drawbacks of using such technologies. The course will also present social media data analyses and how they can assist organisations to improve their overall efficiency and provide competitive advantage. In lectures, students will learn about the scope and metrics for assessing the effectiveness of social media and networking and to develop models for implementing and leveraging social media. In addition, students will, understand techniques for sentiment and text analytics and communicate and assess a firm's social media strategy with a professional approach in written form, and discuss the challenges associated with implementing social media, analytics and networking technologies and their impacts on organisations. In workshops, students will learn to apply a set of techniques to create and extract value from social media, social media metrics and strategic aspects of social media analytics. Students will work on practical examples to complement the theoretical frameworks and concepts. Some of the workshops are also intended to provide students gain basic hands-on experience and practical proficiency using SAS text mining software.

INFS3873 Business Analytics Methods

This is a level 3 course in Business Analytics (BA) that builds on the fundamentals of business intelligence presented in INFS3603. This course exposes students to applications of descriptive, predictive, and prescriptive analytics in order to develop students' ability to use analytics to drive business insights. To develop these skills, students will learn methodological theory and use SAS Enterprise Miner and SAS Visual Analytics software tools to analyse a variety of case studies describing organisational problems with real-world relevance. Emphasis will be placed on using analytics to create value for organisations and being able to communicate analytic findings to a managerial audience.

The first major topic covered in INFS3873 is on conduct segmentation to perform descriptive analytics for large datasets, students will then use visual analytics for data exploration and data presentation. The course will then teach students various regression and data mining techniques to examine relationships between variables. In addition, students will learn a variety of forecasting and optimisation techniques to make data-driven decisions

INFS3020 International Information Systems and Technology Practicum

This is a level 3 Information Systems (IS) course that continues students' study of IS by furthering their knowledge and understanding in international aspects of information systems/technology (IT) business operations (e.g. global IS/IT teams, distributed systems development, eBusiness, and localisation management). This will be attained via first-hand observations of businesses in Asian countries such as China, India, Hong Kong, and South Korea. The central components of the course include a series of seminars and a two-week study tour to one Asian country. During this study tour, students will visit a number of leading international and national organisations, including companies operating in the IS/IT sector and those in other sectors with a significant IS/IT footprint. The primary purpose of these visits will be to enable students to develop an appreciation of the ways in which IS/IT-enabled business operations and business systems differ across national boundaries. Students are required to prepare a written assignment based on the field trip at the end of the tour based on their observations of the businesses and the country. A group presentation and personal reflection report are to be delivered prior to the end of the Semester.

Honours Courses

INFS4886 Principles of Research Design

This is a Level 4 Information Systems (IS) course that continues students' study of IS by furthering their knowledge and skills in relation to research designs. This course focuses on the understanding of IS research philosophies and designs. Students will develop practical skills in developing instruments for both qualitative and quantitative methods.

Topics to be covered in the lectures include an overview of the research process, theory and theorizing, critiquing a research paper, conducting a systematic literature review, writing a literature review, conceptual modelling and research design, archival research, case studies, surveys, experimental and simulation research, action and design research, writing and defending a research proposal. During the weekly practical workshops, students will learn to develop a research proposal and be able to evaluate the quality of their own, and peers', research designs. Students will also research and analyse current trends in various IS topics and present a research proposal at the end of the course.

INFS4887 Business Research Methods

This is a Level 4 Information Systems (IS) course that continues students' study of IS by furthering their knowledge and skills in relation to research methods and analytical skills. Continuing from INFS4886 (Principles of Research Design), this course focuses on the understanding of IS research methodologies.

Topics to be covered in the lectures include overview of knowledge in research methods and techniques of data collection and analysis, SPSS, experimental research, fieldwork, grounded theory, literature review, and thesis writing. During the weekly practical workshops, students will learn from key IS literature how to design and develop a range of research designs and know-how. Students will learn to prepare an independent study including formulating research questions and selecting a research approach, applying research methodology – designing a study and selecting specific methods and techniques appropriate for answering the research questions.

Honours Elective Courses

INFS4805 Information Systems Auditing and Assurance

This is a Level 4 Information Systems (IS) course that continues students' study of IS by furthering their knowledge and skills in relation to information systems auditing. The course examines contemporary IS audit practices and draws on student's business and IS studies to-date. The course introduces students to key auditing concepts and techniques and applies those to the IT environment. In the seminars, students will study a range of topics from the role of the IS audit function to specific IS Audit tools, techniques and

methodologies for the various types of IS Audit. Students will also learn about professional standards (COBIT 5, ITAF), professional practice, ethical behaviour and the legislative and regulatory environment.

Topics to be covered in this course include contemporary IT audit standards and frameworks (including ITAF and COBIT 5) and IT audit regulatory environment. Topics such as developing Audit Programs, auditing Data centres and the physical IT environment, auditing IT security and operating systems, auditing software systems and applications, auditing e-business and mobile applications, and auditing IT projects will also be discussed. Finally, course also covers areas in undertaking risk evaluations and ethical and professional practice considerations for IS Auditors.

INFS4831 Information Systems Consulting

This is a Level 4 Information Systems (IS) course that familiarises students with the key concepts, practices and issues relevant to engaging and providing IS consulting services. The lectures cover a range of themes related to the content and practice of IS consulting, including relevant theories to illustrate how IS consultants engage with organisations and help them solve business problems as well as the challenges and opportunities in contemporary business environments brought about by technological advancements. The weekly seminars encourage the students to further reflect on and apply these concepts to examples and cases from practice.

The topics covered in the course include both the IS consulting process and content. On the process side, the course looks at topics such as style, communication, and stakeholder management. On the content side, the course considers a range of contemporary issues in IS consulting, revolving around the organisational impact of key technology trends that drive the demand for IS consulting, such as crowdsourcing, social media, business analytics, service innovation, as well as security and privacy.

INFS4848 Project, Portfolio and Program Management

This is a Level 4 Information Systems (IS) course that continues students' study of IS by further providing a comprehensive introduction to project management. The course aims to equip students with both theory and practical skills in the management and implementation of projects. The course teaches the following areas of project management knowledge: Integration Management, Scope Management, Time Management, Cost Management, Quality Management, Human Resource Management, Communications Management, Risk Management, Procurement Management, and Stakeholder Management. In addition, students gain knowledge on technical, behavioural and strategic aspects of project, portfolio and program management. During the weekly practical workshops, students work on project teams producing a comprehensive and realistic project plan, thus acquiring

knowledge on IS project management and on principles of ethical and responsible management.

INFS4854 Information Systems Strategy

This is a Level 4 Information Systems (IS) course that familiarises students with the key concepts, practices and issues in the strategic management of IS. The lectures cover theoretical and practical considerations across a variety of strategic IS management issues, which are further examined and applied in the weekly seminars. The course aims to equip the students with the foundational skills needed to be able to meaningfully participate in, or interact with, this aspect of IT management.

The course covers four key themes. It begins with a discussion of the strategic value of IT, including the role of business-IT alignment in realising that value. Second, the course looks at strategic IT decision processes, including planned and emergent strategy-making and governance. Third, the course considers strategy implementation issues, including the role of IT leadership, project and portfolio management, and sourcing decisions. The course closes with a discussion of the strategic role of IT-enabled innovation and current trends in strategy and IT.

INFS4907 Managing Security, Ethics & Privacy in Cyberspace

This is a Level 4 Information Systems (IS) course that introduces students to the awareness and knowledge of IS/IT security related issues occurring in cyberspace. It has a specific emphasis on the need for ethical viewpoints, approaches and practices from a management perspective when addressing the multidimensional challenges and solutions posed by the IS/IT related security problems. The class will be conducted in a semi-formal workshop fashion. Using business cases and scenarios addressing various cyberspace issues, students will study and discuss the ethical and related implications these issues pose to stakeholders. They will learn to manage cyber related security issues responsibly from ethical, social, corporate, responsible management, and professional perspectives. In some situations, they may encounter dilemmas which require a careful balance and trade-off in the way decisions are made.

The topics that are covered in this course include: the nature of cyberspace and ethics/definitions/frameworks and related perspectives and their relevance to cyberspace.

It also covers the importance of IS/IT Security and consequences of security breaches, key IT trends [e.g. 1) Cloud computing, IT outsourcing – client/vendor relationships, Intellectual property, privacy and confidentiality; 2) Social Media; 3) E-business/commerce; 4) E-government] that highlight the importance of security and ethics in cyberspace. Finally, the course also discuss topics in international business and globalisation, difficulties in enforcing ethical practises – cultural, legal, education and accreditation, dilemmas etc., and implications to training and education.