SE Degree Handbook

(Draft 11/11/2016)

SE Degree General Information 2
SE Degree Learning Outcomes 3
SE Degree Structure 4
SE Degree Proposed Specialisations 7
COMP1531 Software Engineering and Data Modelling 11
COMP2511 Object-Oriented Design & Programming 14
COMP1521 Computer Systems Fundamentals 17
COMP1511 Introduction to Programming 20
COMP2521 Data Structures and Algorithms 23
COMP2041 - Software Construction 26
COMP2111 Formal Modelling and Specification 35
SENG2011: Reasoning about Programs: from Specification to Implementation 37
SENG2021 Requirements and Design Workshop 38
SENG3011 Implementation Workshop 43
COMP3141 Design and Software Quality 48
COMP4920/SENG4920 Management and Ethics 50
COMP Courses List 58
ISTM BIS Curriculum Review (June 2016) 63
COMP3331 Computer Networks and Applications 75
COMP3311: Database Systems 82
COMP3511 Human Computer Interaction 85

SE Degree General Information

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 CSE 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

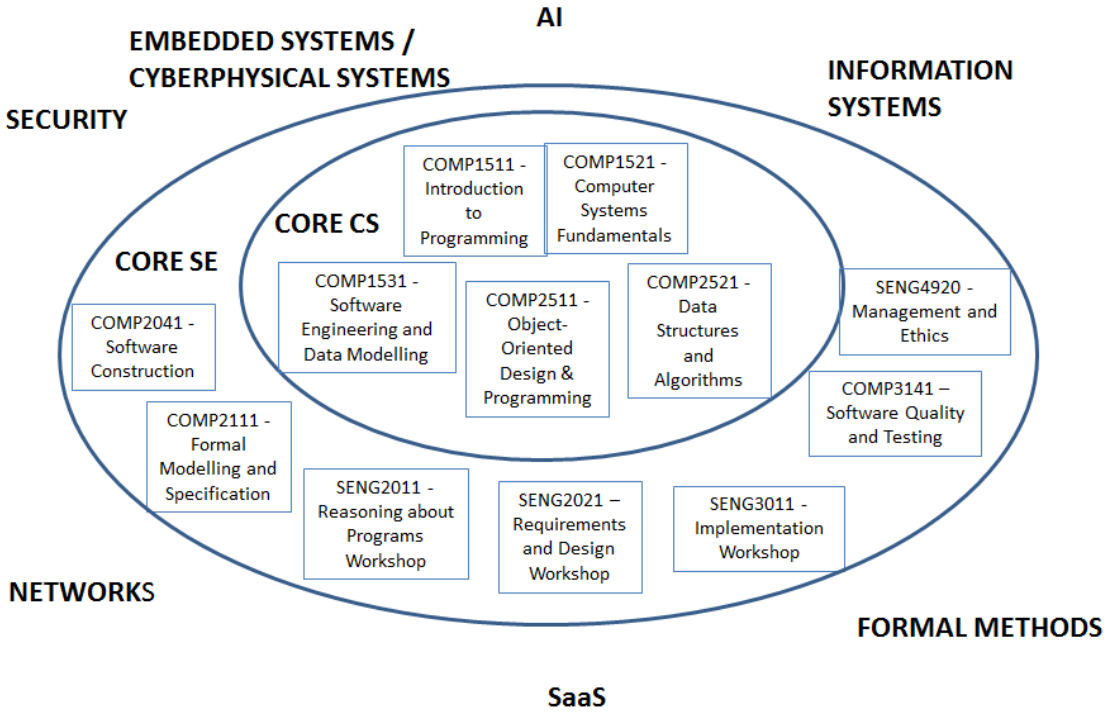
1. Demonstrate a solid understanding of the software engineering knowledge and skills, necessary to begin practice as a software engineer.
2. ability to appropriately define and apply relevant abstractions from algorithmics, computer science, and mathematics to complex software system development.
3. 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

The current program structure described in:
http://webapps.cse.unsw.edu.au/cse/new/

In the new structure, students must complete 192 Units of Credit (UoC) including:

- 168 UOC from SE Stream
- 12 UOC of General Education courses
- 12 UOC electives (Foundational Disciplinary or Disciplinary Knowledge Courses



SE Stream Core courses[1] (114 UOC):

- COMP1511 Introduction to Programming (6 UOC)
- COMP1521 Computer Systems Fundamentals (6 UOC)

[1] For latest information on core courses syllabuses, see
http://webapps.cse.unsw.edu.au/cse/core/index.php

- COMP1531 Software Engineering and Data Modelling (6UOC)
- COMP2041 Software Construction (6 UOC)
- COMP2111 Formal Modelling and Specification (6 UOC)
- COMP2511 Object-oriented Design & Programming (6 UOC)
- COMP2521 Data Structures and Algorithms (6 UOC)
- COMP3141 Software Quality and Testing (6 UOC)
- SENG4920 Ethics and Management (6 UOC)
- COMP4930 Thesis Part A (6 UOC)
- COMP4931 Thesis Part B (6 UOC)
- ENGG1000 Engineering Design (6 UOC)
- MATH1081 Discrete Mathematics (6 UOC)
- MATH1131 Mathematics 1A (6 UOC) **or** MATH1141 Higher Mathematics 1A (6 UOC)
- MATH1231 Mathematics 1B (6 UOC) **or** MATH1241 Higher Mathematics 1B (6 UOC)
- MATH2400 Finite Mathematics (3 UOC)
- MATH2589 Probability and Statistics (3 UOC)
- SENG2011 Reasoning about Programs Workshop (6 UOC)
- SENG2021 Requirements and Design Workshop (6 UOC)
- SENG3011 Implementation Workshop 3 (6 UOC)

Plus:

- Professional Electives (48 UOC): Any course from
    - level 3, 4, 6 or 9 COMP courses, or
    - level 3 or 4 INFS/MATH/ELEC/TELE courses
- Free Elective (6 UOC)
- 60 days Industrial Training

Professional Elective Definition:

- 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

A number of specialisations are proposed with “packages” of electives.

Limit Requirements

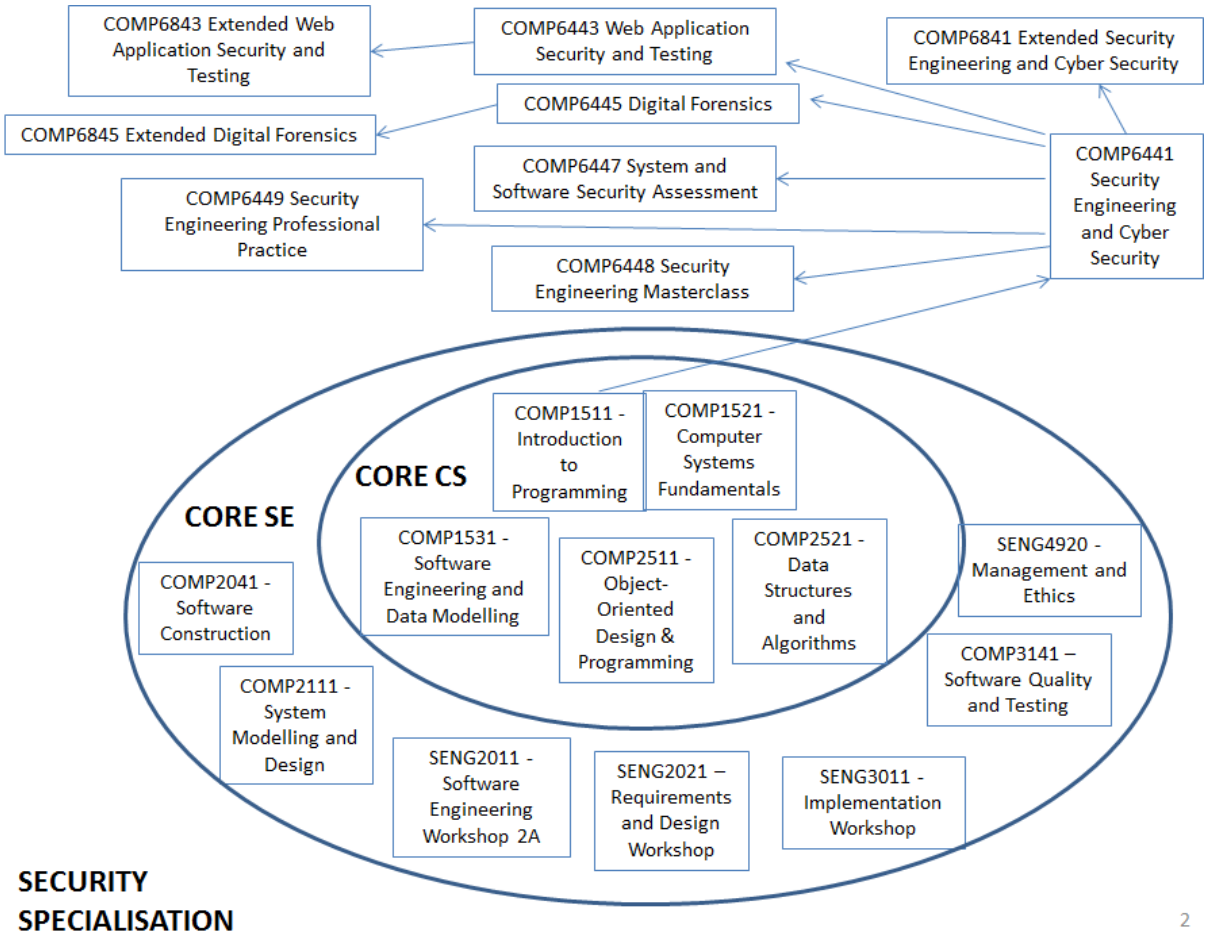
Level 4 UOC Minimum Students must complete a minimum of 30 UOC of the following courses.

- 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

SE Degree Proposed Specialisations

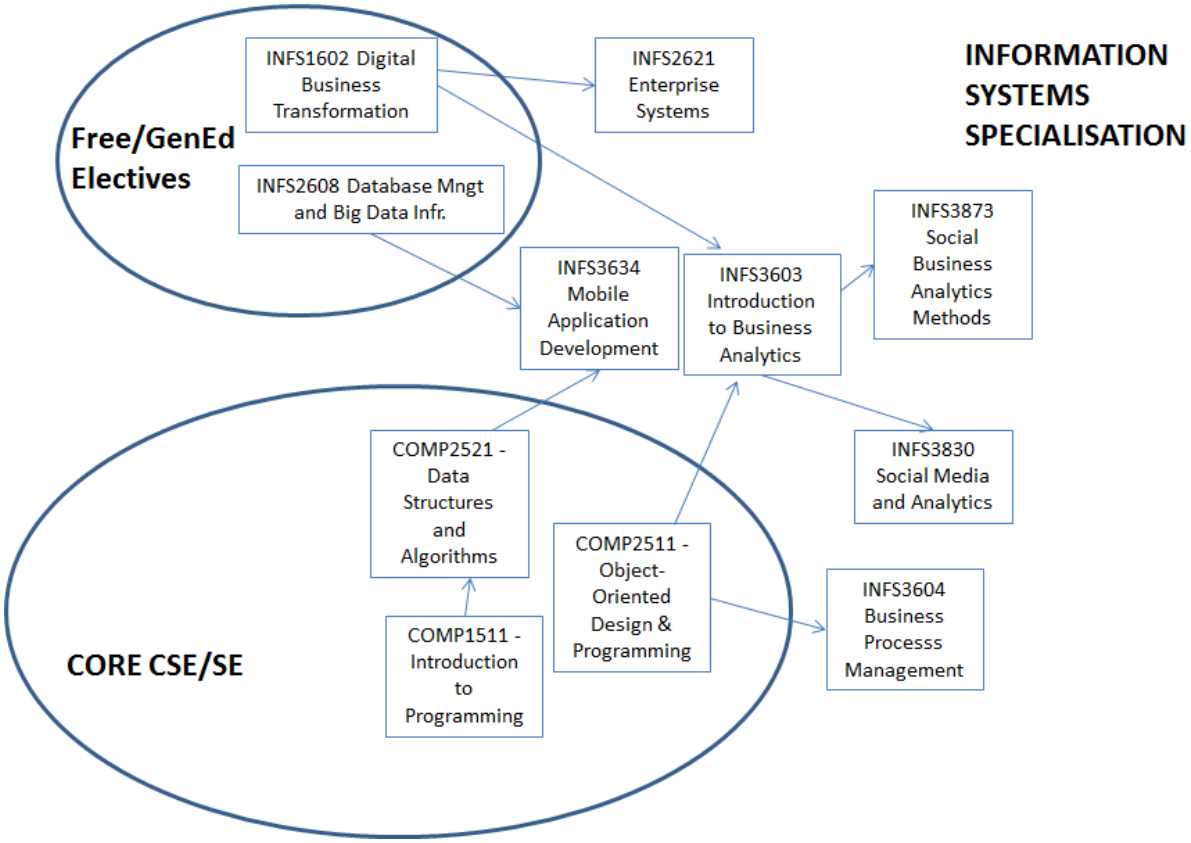
Security specialisation

Defined around new courses funded by CBA initiative.



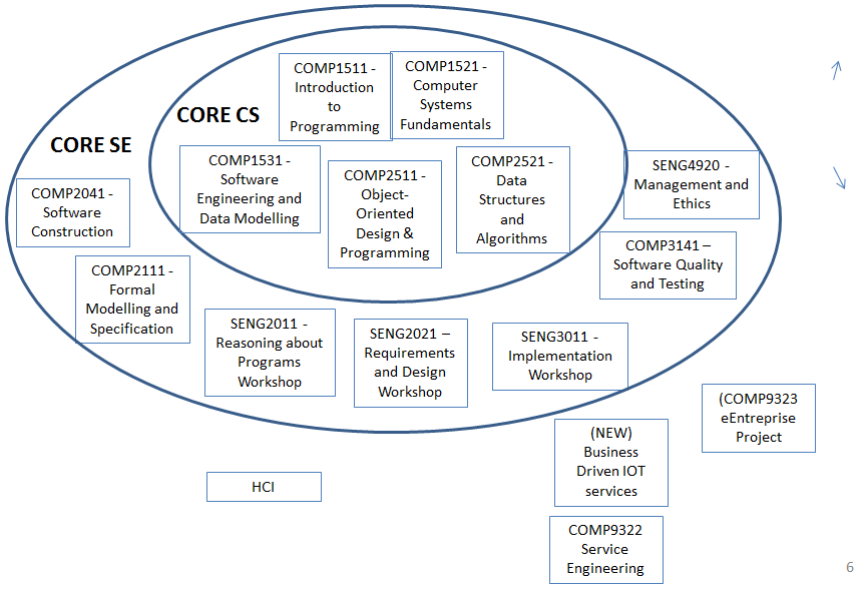
Information Systems Specialisation

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

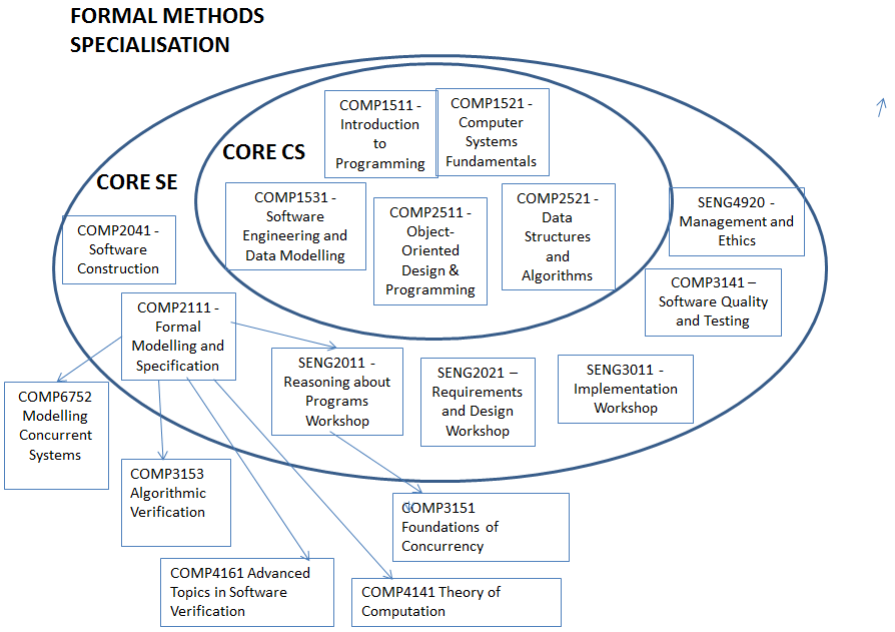


Software as a Service Specialisation

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



Formal Methods Specialisation

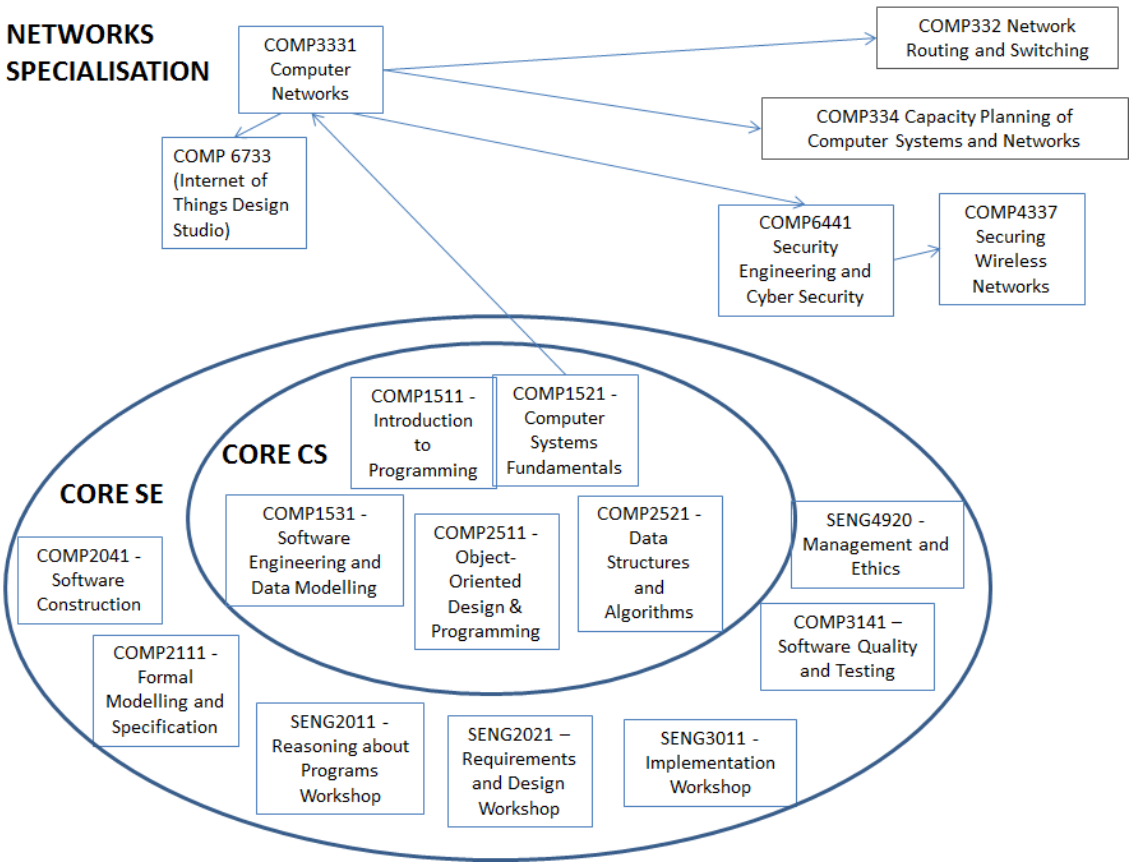


Data science Specialisation (awaiting information)

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

Networks Specialisation

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



Embedded Systems Specialisation (awaiting information)

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

AI Specialisation (awaiting information)

Defined around the work of AI Research Group

COMP1531 Software Engineering and Data Modelling

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 and development practices on Web platforms

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 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
- understand agile methods and the principles of testing, code development and validating software
-
- understand the architecture of simple Web systems

Topics

- software processes and project management
- software tools and development environments
- web technologies and web application architectures
- 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	Software Requirement Engineering - requirement elicitation, describing system data using UML or ER, functional/non-functional requirements	xxx
WeeksWeek 3-6	CData modelling and introduction to databases, E-R model, E-R diagrams	
Week 4	Relational model, SQL schemas, E-R-to-SQL	
Week 5	SQL: tables, select, from, where, aggregation	
Week 6	SQL: join, group-by, views	
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 version control, build/deployment management, tool integration	xxx
Week 10	Group project: Web front-ends, Javascript/CSS Introduction	xxx
Week 11	Group Project Demonstrations	-
Week 12	Group Project Showcase	-

COMP2511 Object-Oriented Design & Programming

School of Computer Science and Engineering, UNSW

Overview

Code/Title	COMP2511 Object-Oriented Design & Programming
Abbreviations	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 tut-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 ...

- design appropriate solutions to medium-scale problems using an object-oriented approach
- apply a systematic object-oriented design process as a part of following software engineering best practices
- apply an agile software development method to organize team-based projects
- create medium-scale object-oriented programs using appropriate design principles
- use appropriate software engineering tools for the development of medium-scale software systems

Topics

- object-oriented design
- object-oriented programming
- agile software processes
- design patterns
- introduction to user interface design and programming
- introduction to concurrency

Schedule

Week	Lectures	Tut-Labs
Week 1	Principles of Design; Software Engineering; Java basics, Eclipse	
Week 2	Object-Oriented Programming (Objects vs ADTs, inheritance, encapsulation, polymorphism, Java object model, equality, cloning)	Basic Java programming
Week 3	Design by Contract (Pre- and Postconditions, Class invariants, exceptions, Javadoc, Unit testing, JUnit)	Programming by Contract
Week 4	Object-Oriented Design (Use cases, CRC Cards, UML)	Object-Oriented Design
Week 5	Generic Types and Polymorphism (Java type system, generic types, Arrays and Lists, Interface types)	Generic Types
Week	Introduction to Design Patterns (Iterator Pattern, Strategy	Search Algorithms

6	Pattern)	
Week 7	Problem-Solving Algorithms (A* search, heuristics)	Problem-Solving Algorithms
Week 8	Agile Software Processes (Scrum, Extreme Programming, Agile Planning and Estimation)	Agile Software Processes
Week 9	User Interface Design and Programming (Java GUI programming, event-driven programming, Observer Pattern)	User Interface Programming
Week 10	Complex Design Patterns (Decorator Pattern, Composite Pattern)	Sprint Reviews
Week 11	Introduction to Concurrency (multi-threaded systems, race conditions, synchronization, locks)	Design Patterns
Week 12	Review	Concurrency

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

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

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

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

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
- 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)

Schedule

Week	Lectures	Labs
Week 1	introduction, about university, learning, abstraction, estimation, programming, C, problem solving	introduction, lab/workstation orientation, first program
Week	types, variables, memory, design with functions,	program style, version

2	top down design, arithmetic expressions, layout, programming style	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:

- implement software in the procedural paradigm up to several 1000's LoC
- employ a range of fundamental data types in developing software solutions (e.g. arrays, structs, matrices, sets, lists, ...)
- can design and implement simple abstract data types
- reason about the behaviour (correctness and efficiency) of programs (e.g. defining pre- and post-conditions, efficiency of sorting algorithms)
- explain how programs work at the machine level (stack, heap, ...)
- work effectively in teams, following a systematic development process
- develop and use test suites for functions and programs
- work with a range of tools for program develoment (editors, compilers, debuggers, profilers, version control systems)
- 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 ...

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

Topics

- fundamental data structures: lists, trees, graphs
- algorithm and program analysis
- techniques for sorting, searching, traversing

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 - Software Construction

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

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
-

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:

- 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;
- paraphrasing another person's work with very minor changes keeping the meaning, form and/or progression of ideas of the original;
- piecing together sections of the work of others into a new whole;

- 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
- claiming credit for a proportion a work contributed to a group assessment item that is greater than that actually contributed.

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.

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.

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

COMP2111 Formal Modelling and Specification

Learning Objectives:

- develop an appreciation of the relevance of discrete mathematics to computing
- improve facility in the use of discrete mathematics concepts
- improve capacity for rigorous reasoning
- learn to use a toolkit of formal modelling approaches frequently used in computing

The course as a whole addresses MAA.md.2, DES.dd.4

Content areas:

- Propositional Logic. Validity, Satisfiability.

  Applications to hardware design, refactoring/simplification of conditional statements, scheduling problems, SAT as a basis for verification.
- Natural deduction for propositional logic.

  Reasoning about propositional specifications.
- Set theory: Relations, transitive closure

  Application: relational databases
- Predicate Logic: terms, substitution, term rewriting, pattern matching

  Operational semantics of a simple imperative language using relations, transitive closure

  Hoare logic rules for assignment statements, if statements

- Predicate Logic: quantification

  Applications: database integrity constraints, program interface specifica- tions (preconditions and postconditions)

  Exercise: Drill in writing predicate logic specifications with a proper interaction between quantifiers and propositional operators, understanding subtleties of universal quantification in the trivial case.

- Set Theory: functions and types

  Introduction to functional programming, functional programs as high level specs.

- Inductively defined sets, and structural induction.

  Application: recursive data type definitions, recursive functions defined over these, and reasoning about these.

  Practical Exercise: write some simple Haskell programs, prove properties of them

- Hoare Logic for loops (MAA.md.2)

  Simple examples, e.g. summation, max (More complex examples left for SE2011)

- Reasoning: natural deduction for predicate logic (MAA.mf.2)

  Applications: reasoning about equivalent formulations of a specification, examples showing that a postcondition implies a precondition (examples of this for loop exit conditions)

- State Machine models (MAA.tm.2)

  Invariants (MAA.md.2)

  Manual proof of invariants.

  Model checking of invariants (MAA.af.2)

- Formal Languages, Deterministic and Nondeterministic Finite State Automata, Regular expressions (FND.mf.7)

  Application: exercises with unix tools using regexp, simple concurrency questions

  (Leave determinization, pumping lemma for COMP4141)

- Context Free Grammars (FND.mf.8)

  Application: grammar for a simple expression language, grammar for a fragment of HTML

  Practical exercise: Given a Haskell parser (generated from a CFG using happy) and evaluator for a simple expression language, extend it to a more expressive language

SENG2011: Reasoning about Programs: from Specification to Implementation

Learning Objectives:

- Gain practical experience in rigorous modelling and specification
- Learn patterns of reasoning by which programs implementing a specifica- tion can be derived through a compositional process.
- Learn patterns of reasoning for showing program correctness.
- Understand decomposition, abstraction and refinement.

The course uses formal methods, but the focus is on acquiring an understand- ing of the patterns of reasoning treated, which can also be applied informally to improve students' program development ability – this will be the take-away message for many students.

Topics:

- Hoare logic, pre-conditions, post-conditions and invariants, revisited through larger applications (MAA.md.1)
- Termination: variants, well-founded orders, lexicographic order.
- Refinement Calculus and program derivation (MAA.md.1)
- Replacement of an inefficient datatype implementation by an equivalent, more efficient implementation – coupling invariants
- Use of a formal verification tool (e.g., Dafny/Alloy) (MAA.mf.4)

The emphasis is on developing an understanding of these ideas through many worked examples of correct program development.

SENG2021 Requirements and Design Workshop

Course Code:	SENG2021
Course Title:	Requirements and Design Workshop
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 is part the series of software engineering workshops designed to teach students to work in teams and apply their knowledge to solve real-life problems. This workshop will offer students the opportunity to concentrate on software requirements analysis and design issues including artefacts produced as well techniques and tools to support this process (brainstorming, problem statements, requirements elicitation, producing design documents and prototyping). In addition, it aims to provide students with some of teamwork skills, requirements engineering and design techniques that an engineer would use in the early stages of the development process. The students will also be getting experience on different aspects of designing a Web application with a major focus on the front-end. The requirements for this course will be determined in collaboration with industry partners and will relate to developing a realistic 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 concepts and technologies and on how to produced artefacts in general. The course has a number of industry sponsors that include Fairfax Media and Macquarie Bank.

Course Aims

To develop:

- a practical appreciation of the software requirements and design process;
- an understanding of the relation between user requirements, design concepts and implementation considerations;
- an understanding of the quality of project management and the role of managers, users, designers, programmers and analysts throughout the system's development process
- an understanding of Web systems requirements, design and prototyping approaches;
- an ability to produce key requirements and design documents describing how a specified system will be implemented;
- an appreciation of basic usability and Human Computer Interaction (HCI) issues

Student Learning Outcomes

After completing this course, students will:

- reinforce existing knowledge about the concepts and principles in the early stages of the software development life cycle
- experience with the development of project plans, brainstorming, requirement documents, prototyping techniques, issues and tasks management, peer reviews
- 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.
- the recognition that production of quality software is a task demanding a disciplined approach to all stages of its development
- an appreciation of the many and varied issues involved in the development of software systems and the role and the importance that Software Engineering review processes play in producing quality systems
- develop an awareness of the community of engineering professions and the importance of keeping current through life-long learning and through interacting with that community. Students will also be encouraged to develop their research skills as one of the means of acquiring the necessary knowledge and skills to solve engineering problems

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 course, students will learn and apply generic skills including requirement elicitation techniques, the design review process, developing team skills, working creatively through group work and brainstorming and managing the varying levels of uncertainty involved in these processes. A primary goal of this course is to teach students the importance and process of group work. Students will learn that both human and technical views of design and implementation are equally important issues. In order to develop long-lasting and efficient software systems—in essence a quality product—Software Engineers need to be aware of and address both these factors in the development process. Less tangible outcomes are for students to also see that specifications are not the panacea of correctness, specification can be wrong and more importantly they can be subtly wrong. In addition, students will see first hand how assumptions and bias can sometimes be embodied in a specification unintentionally.
The workshop follows 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:

- Brainstorming
- Software specification artefacts (problem statements, stories, class
- Requirements diagrams, use cases, low-fidelity prototypes, high-fidelity prototypes)
- Design reports: architectural design, GUI design, class and sequence diagrams
- Group presentations: pitching, prototype demo
- Peer reviews

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 (TO BE DONE)

Weekly Schedule

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 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 on a continuous basis during mentoring sessions.

SENG3011 Implementation Workshop

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)

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.

COMP3141 Design and Software Quality

- Software correctness
- Functional programming
- Introduction to functional programming
- Types and software quality
- Ensuring software quality (industry guest lecture)
- Unit testing versus property based testing
- Logical program properties
- Static versus dynamic checking
- Controlling effects
- Machine checked program properties
- Concurrency and parallelism

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 the UNSW 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.

COMP Courses List

Semester 1

Code	Course
COMP2121	Microprocessors & Interfacing
COMP3121 COMP9801 COMP3821 COMP9101	Algorithms & Programming Tech
COMP3131 COMP9102	Programming Languages & Compil
COMP3141	Software Sys Des&Implementat'n
COMP3153 COMP9153	Algorithmic Verification
COMP3211 COMP9211	Computer Architecture
COMP3231 COMP9283 COMP3891 COMP9201	Operating Systems
COMP3311	Database Systems
COMP3331 COMP9331	Computer Networks&Applications
COMP3411 COMP9814 COMP9414	Artificial Intelligence
COMP3441 COMP9441	Security Engineering
COMP4128	Programming Challenges

COMP4141	Theory of Computation
COMP4337 COMP9337	Securing Wireless Networks
COMP6441 COMP6841	Security Engineering and Cyber Security
COMP6443 COMP6843	Web Application Security and Testing
COMP6752	Modelling Concurrent Systems
COMP9020	Foundations of Comp. Science
COMP9021	Principles of Programming
COMP9024	Data Structures & Algorithms
COMP9243	Distributed Systems
COMP9311	Database Systems
COMP9313	Big Data Management
COMP9318	Data Warehousing & Data Mining
COMP9319	Web Data Compression & Search
COMP9321	Web Applications Engineering
COMP9322	Service-Oriented Architectures

COMP9334	Systems Capacity Planning
COMP9417	Machine Learning & Data Mining

Semester 2

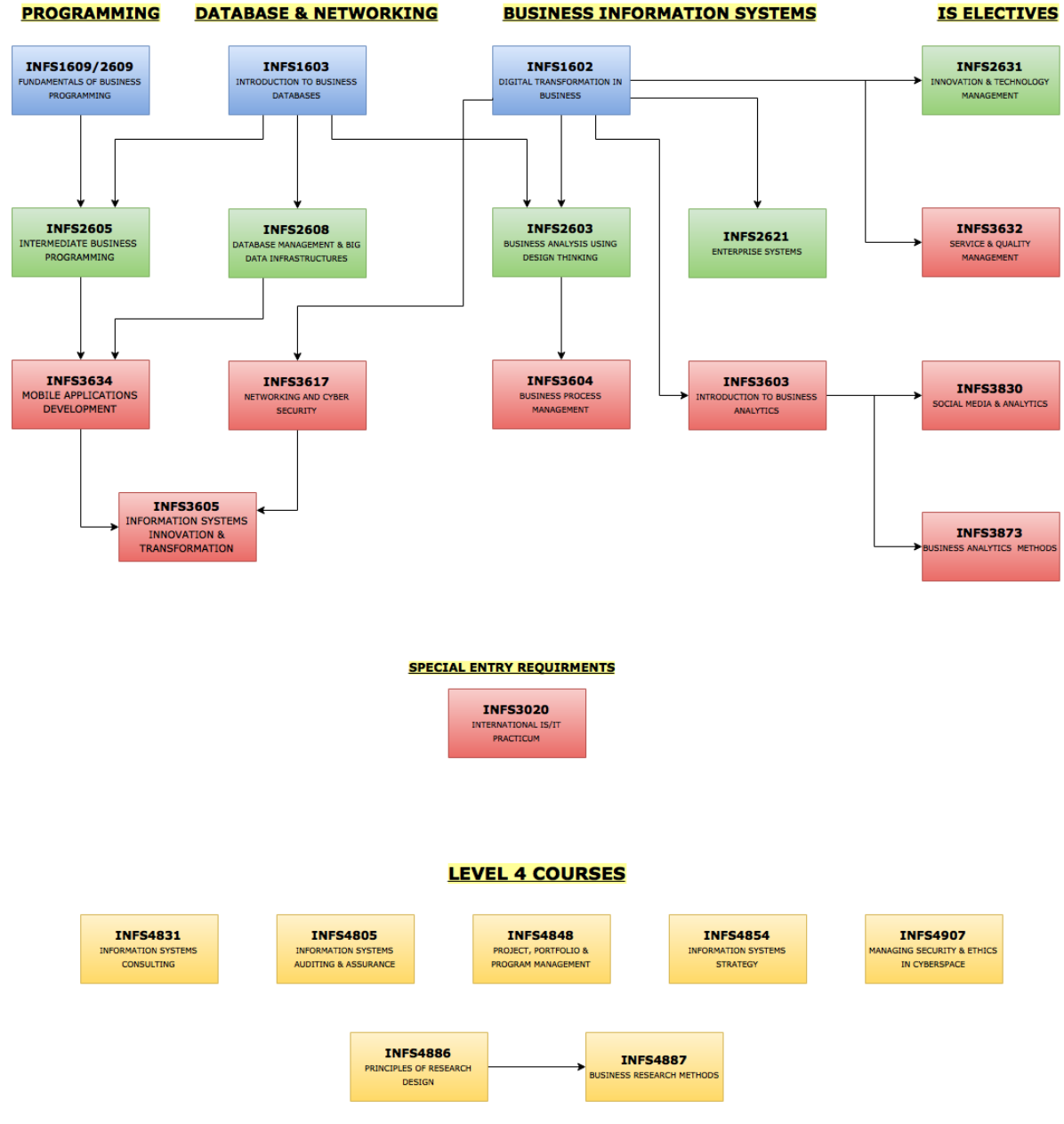
Code	Course
COMP2121	Microprocessors & Interfacing
COMP3151 COMP9151	Foundations of Concurrency
COMP3161 COMP9161	Concepts of Programming Lang.
COMP3222 COMP9222	Digital Circuits and Systems
COMP3331 COMP9331	Computer Networks&Applications
COMP3421 COMP9415	Computer Graphics
COMP3431 COMP9431	Robotic Software Architecture
COMP3511 COMP9511	Human Computer Interaction
COMP4121	Advanced & Parallel Algorithms
COMP4161	Advanced Verification
COMP4336 COMP9336	Mobile Data Networking

COMP4418	Knowledge Representation
COMP4920	Management and Ethics
COMP6447	
COMP6714	Info Retrieval and Web Search
COMP6733	
COMP6741	Parameterized & Exact Comp.
COMP6771	Advanced C++ Programming
COMP9000	Special Program
COMP9020	Foundations of Comp. Science
COMP9021	Principles of Programming
COMP9024	Data Structures & Algorithms
COMP9032	Microprocessors & Interfacing
COMP9242	Advanced Operating Systems
COMP9311	Database Systems
COMP9313	Big Data Management
COMP9321	Web Applications Engineering

COMP9322	Service-Oriented Architectures
COMP9323	e-Enterprise Project
COMP9418	
COMP9444 COMP9844	Neural Networks
COMP9517	Computer Vision

ISTM BIS Curriculum Review (June 2016)

Overview



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.

COMP3331 Computer Networks and Applications

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:

- 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
- have a solid understanding of the current architecture of the Internet and the entities involved in its operations
- be able to identify soundness or potential flaws in proposed protocols
- be equipped with the necessary skills to design networked applications and protocols
- implement and write protocols and applications in C, Java or Python
- analyse and evaluate the performance of computer networks
- be able to capture and analyse network traffic
- be able to understand and explain security and ethical issues in computer networking

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.

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.

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

COMP3511 Human Computer Interaction

Course Details

- 6 units of credit (UoC)

- Pre-requisites
    - 48 units of credit from any program (undergraduates)
    - No pre-requisites for postgraduates
- This course is a pre-requisite for COMP4511 User Interface Design and Construction and any HCI related thesis.
- The lectures (Tuesday 4-7pm, in CLB 6) are common for undergraduates and postgraduates.
- Each student should be enrolled in one of the designated 2 hour tutorial/laboratory time slots
- Tutorial/laboratory will start in Week 2 and go through until Week 12.
- Tutorial/laboratory and assignment checkpoints will take place every week in G13-K17 (Mac laboratory) also known as the "CHIL" (Computer Human Interaction Laboratory)
- Postgraduates and undergraduates will have different assignment contexts.

Course Summary

- Lecture topics are summarised in the Course Schedule below.
- The course includes topics relating to Requirements, Design, Prototyping and Evaluation within the User Centred Design process.
- You will also be given the skills to conduct a basic Usability Evaluation.
- Other topics covered within the course allow you to understand your users and their needs. This includes an overview of basic Cognitive capacities, Designing for Accessibility, Internationalisation, levels of Expertise, and Collaboration.
- You will also be looking at the differences between Scientific Data Gathering and User Studies, with a consideration for Human Ethics.
- Other topics include Visual Design principles, and looking at different Input/Output devices and their potential impact on Design.

Course Aims

- to develop your skills in the area of user-centred design
- to provide background knowledge about how people think and process information
- to demonstrate techniques/heuristics necessary to evaluate systems for their usability
- to give you the capability of executing a user-centred design process
- to give you experience in using paper-based design techniques
- to give you experience in the formal evaluation of user interfaces
- to give you exposure to developing electronic prototypes of user interfaces
- to ensure that your design work includes user needs analysis
- to give you an awareness of user centred design tools, methods, and techniques
- above all, maintain a real-world perspective to applying this knowledge in industry

Learning Outcomes

10 Core Learning Outcomes

- Be able to prepare a project plan that is based on user-centred design principles and then carry out activities to design, evaluate and refine user interaction based on iteration.
- To develop the skills necessary to create a user interface evaluation report (written and oral) that critiques a user interface.
- Understand the strengths and limitations of human cognition and memory and apply these to the design of more usable interfaces that do not cognitively overload users.
- To develop design skills, primarily using paper for rapid solutions, and consolidate individual designs in small groups to understand the importance of design decisions and the selection process.
- Prepare and carry out usability walkthroughs to evaluate both paper and electronic based designs for their usability, and then create structured reports that quantify the issues discovered from evaluation activities.
- To ensure that your design work includes user needs analysis and is not just a reflection of what you believe your users need.
- Construct questionnaires/surveys to obtain pre- and post-test information from users, and to understand the importance of ethics and privacy in order be able to carry out appropriate user- centred design activities.
- Understand the relationship between the scientific method and the user-centred design approach and be aware of the scientific and research approaches used in user interface design research.
- Understand how user centred design processes should be inclusive of all users, including international audiences, those with special needs, such as disabilities, as well different levels of user experience, and use this knowledge to design interfaces appropriate to a particular group of users.
- To develop an awareness of user-centred design tools, methods, and techniques and maintain a real-world perspective in order to be able to apply this knowledge in industry.

Broader Learning outcomes

- Through the use of a design diary, develop an understanding of design conceptualisation, technical and creative thinking
- Distinguish (user-centred) design from (code) implementation
- Design a project plan that includes the important role of the user in the software design lifecycle
- Critique a user interface basing your evaluation on design principles, usability goals and user experience goals
- Be able to use the heuristic evaluation technique for evaluating user interfaces
- Describe the characteristics of human cognitive and perceptual capacities and their relationship to user interaction
- Understand the different methods people use to solve problems
- Describe the basic human cognitive architecture
- Be able to define and describe (with examples) cognitive load theory principles including the redundancy effect, split attention effect, worked example effect and modality effect
- Be able to apply cognitive load theory to the design of more usable interfaces that do not cognitively overload users
- Develop an understanding of the nature of human expertise, including an understanding of novices' capabilities and needs
- Use the knowledge of experts and novices to be able to design interfaces appropriate to a particular group of users

- Understand the difference between quantitative and qualitative research methodologies
- Understand the different phases of the user-centred design approach
- Be able to identify and distinguish users and stakeholders for a particular design situation
- Create scenarios and personas and apply them throughout the design and evaluation process
- Be able to deconstruct a system design into information, interaction and visual design components
- Appreciate the complexities of visual design and the role of graphic and visual designers
- Apply data analysis techniques to understand and refine information architecture and system requirements
- Carry out design activities to design, evaluate and refine user interaction
- Design and sketch primarily with paper to obtain rapid solutions to design questions
- Design on your own and in small groups, consolidating individual designs to understand the importance of design decisions and the process by which selection is made
- Understand the user interface design issues surrounding web design
- Develop an understanding of conventional and future input and output devices that extend the user experience beyond the graphical user interface
- Understand how to construct non-functioning visual electronic prototypes based on previous paper based design activities
- Appreciate the special needs of other people, being able to define the goals of Universal Access and understand how user-centred design processes should also be inclusive of special needs
- Understand the broader issues that technology and user interfaces play in the area of occupational health and safety
- Become aware of the design issues for preparing user interfaces for international audiences (those other than English speaking), and considerations that need to be made in the implementation phase
- Understand the issues surrounding the design of social and collaborative software, and the need for this type of software
- Be able to quantify user interaction in terms of low level interactions, and understand some of the mathematical techniques used to measure that interaction
- Become aware of the scientific and research approaches used in user interface design research

Assumed Knowledge

The assumed knowledge for this course is that you know how to write a report and/or essay for your assignments, and that you are familiar with the technology in the Mac lab. Because students come from a variety of backgrounds, with different knowledge bases, the assumed knowledge is not extensive. The course does, however, involve extensive reading.

Teaching Rationale

Failing to take into consideration the needs of your software user audience will lead to costly disaster. People will become frustrated because the application does not work the way that they expect. You know it yourself – you have encountered web sites that are difficult and non-intuitive to use. We aim to show you a design process that helps reduce such user interface difficulties before users are unleashed on your software. This design process starts with understanding people. The process involves an on-going working relationship with potential users during the entire design of a system; not just in the software-testing phase.

Engineers have created many software applications without consultation with the immediate user audience. They may have talked to the managers of the software (those that will pay the

development cost bills) but have not talked to the end users. The end users have valuable insight into the workflow of organizations, and this is complimented with knowledge from other stakeholders.

The intention is not for lectures to reiterate the text material but to re-activate it, re-represent it, elaborate it, and demonstrate the application of it to design. This implies, and it will be assumed, that you have done the reading prior to lecture. If you have questions about the reading, the lectures, or the interrelation between the two, make sure that you ask in lectures or via the various consultation methods described below.

Teaching strategies

Tuesday 4-7pm is a common lecture that will have lecture material, design diary exercises and some small group activities. Given the later time slot we will endeavour to make this more engaging than a typical lecture format. The lecture period will need your participation to make this work successfully. **You will need to bring your design diary.** Your participation in classes may count towards your participation component. For those enrolled in the webstream and watching lectures online, you will be expected to have listened to the relevant lecture in advance of tutorials and to come prepared to class. Failure to do so may result is associated penalties.

Each week you will be required to participate in your timetabled tutorial/laboratory class. This will be held in the CHIL (Computer Human Interaction Laboratory) G13-K17, ground floor Mac lab. Bring your design diary to tutorial class and remember to date each page. It will act as evidence of your original design and assignment work.

Regular progress on assignment 2 group work is required and will be checked with weekly or fortnightly deliverables. This is designed to keep you working regularly on your assignments so that you don't leave things until the last minute. During some scheduled tutorial classes (see web site and assignment pages for dates) there will be assessable in-class activities and checkpoints (due at the beginning of the class) relating to assignment milestones. Late penalties will be applied if you have not adequately prepared for these activities.

This will also be a time for you to ask questions of your tutor, and for your tutor to give you some feedback on your work.

The practical periods in the tutorial/laboratory are intended to facilitate group discussion and to give you the ability to work through practical examples.

Your design diary will be marked periodically in tutorials and will be collected at the end of the semester for assessment and review. Your tutor will date stamp the diary in tutorial class. You are encouraged to find your own design examples of bad user interaction experiences. This may involve you taking a photograph, as an example, and gluing a print of that photo into your diary and writing up your ideas as to why the interaction is poor and solutions to improve.

This course appears to some as being "easy" but the reality is that it isn't. (This comment comes from student feedback). Many unfortunately don't make this realisation until the final weeks.

- There is a lot more reading than other courses
- Unlike code, you cannot hack out a solution the night before
- Design takes a lot more thinking and conceptualisation to explore the problem space
- The process is iterative and you have to demonstrate improvements that evolve from iteration

- Your design work involves discussing issues with potential users
- Your design work involves discussing and working with others in your group

Assignments

Assignment 2 context will differ between postgraduates and undergraduates to cater for the different experiences and learning approaches. This strategy has been formulated based on our own observations and feedback from students.

All students (COMP3511/COMP9511) will complete 2 assignments.

- Assignment 1 – Individual Website Design Critique
- Assignment 2 – Group User Interface Design

Assignment 1 focuses on heuristic evaluation, design principles and usability principles. For Postgraduates and Undergraduates, you will apply your understanding of these concepts when evaluating a series of websites.

Assignment 2 is a group design activity where the group will carry out a full user centred design process to create a series of paper prototypes of a system. The process starts with design conceptualisation, analysing user needs and goals, through a number of design iterations, with on- going evaluation. You will discover through your testing that your first design will have flaws and not work the way the user expects. Iteration becomes an essential technique to improve the situation. Iteration is combined with an evaluation process to formally analyse whether improvements are being made.

Assignment 2 is heavily focused on paper design and introduces the formal evaluation process. The first phase will be based on individual design work, whilst the second phase will be carried out with a team of 3-4 students to consolidate individual designs. Group members must be from the *same* tutorial class because assessable exercises are carried out in tutorial time – so all group members must be present.

In week 9, a formal usability evaluation will be run by your group and observed and assessed. The outcomes of the evaluation and the subsequent design discussion will be written up and added to the final group report. This provides an opportunity to incorporate feedback from experienced tutors. In addition to the report, a final group presentation of the design will be presented in tutorial class in week 12.