SOFTENG 254:
Quality Assurance

Lecture 1a: Introduction

Paramvir Singh
School of Computer Science

Agenda

- Admin
  - Labs are neither assessed nor compulsory, but are examinable
  - Start Thursday this week ⇒ ON (JUnit Review)
- What is Software Engineering
- What is Software Quality Assurance
- The software lifecycle (details in Part-2)

People

Paramvir Singh Course Coordinator, Lecturer

p.singh@auckland.ac.nz

Office hours: Zoom 🙂

Aiden Burgess Tutor

Sreeniketh Raghavan Tutor

Details on Canvas

PART II ECSE ASSISTANCE CENTRE

Semester Two starts on Week 2

EEE and COMPSYS: Wednesdays 12-1pm & 3-4pm | Thursdays 1-3pm

Software: Mondays 3-4pm | Thursdays 10am-12pm

Leech Study Space, during teaching weeks only

PART II ECSE ASSISTANCE CENTRE

Leech Study Space

Semester Two hours

EEE and COMPSYS:

- **Wednesdays 12-1pm**
- **Wednesdays 3-4pm**
- **Thursdays 1-3pm**

Software:

- **Mondays 3-4pm**
- **Thursdays 10am-12pm**

FREE support is provided to help you:

- Prepare for tests or exams
- Go through weekly tutorial problems
- Prepare for assignments by working through similar model questions

Group or one-on-one tutoring is available so come along and bring a friend!

Communication

- Meetings — lectures, tutorials, labs, office hours, Zoom, Discord.
- Resources
  - Canvas — course outline, assignment handouts, possibly other stuff
- Course marks — Canvas
- Email — Electronic Mail is an official and the primary means of communication with students
- Journal Construction

Communication

- Email — students ⇐⇒ course staff
- Canvas Announcements/notifications — course staff ⇒ students
- Piazza (`piazza.com`) is available via Canvas — invitations already sent (if not, contact me by email)
  - asynchronous broadcast by all
  - ask questions that can be answered by anyone in the class (including course staff)
  - Use for general discussion on course matters
  - Do not post answers to assignment questions

Course schedule

Week	Wed Lecture - 1	Wed Lecture - 2	Thurs Lecture - 1	Thurs Lecture - 2	Lab	Assignment
1	Course Introduction	Software Quality	Overview of Testing	Statement Coverage	Drop-In (Thursday only)	
2	Paths	Practicalities	Data flow testing - 1	Tutorial	Drop-in	
3	Data flow testing - 2	Input Space Partitioning	Logic Coverage	Tutorial	Drop-in	Iteration 1 (A1) due
4	Decision Tables	Testing the whole system	Impact of Language features on testing	Tutorial	Drop-in	
5	Mock Testing	Measurement Theory	Software Metrics	Discussion	Drop-in	
6	**Test1**		Other Topics	Discussion	Drop-in	
7	Part-2 Introduction	Software Development Process - 1	Software Development Process - 2	CMMI	Drop-In (Thursday only)	Iteration 2 (A2) due
8	Revisiting Object-Oriented Modelling	Modelling with UML - I	UI Modelling	Tutorial	Drop-in	
9	Software Design	Smells and Refactoring	Reflection	Tutorial: Software Modelling and Design	Drop-in	Iteration 3 (A3) due
10	Source and Version Control	Git and GitHub	Git and GitHub - 2	Tutorial	Drop-in	
11	Bug Tracking Systems	Automated Build Scripts	Continuous integration	Build Environments and Dev-Ops	Drop-in	
12	**Test2**		Code Comprehension	Discussion	Drop-in	Iteration 3 (A4) due

Examinable Material

- Lecture slides
- Lecture content (what is said, not just what is on the slides)
- Lab content
- Assignment content
- Suggested readings
- Piazza comments
- Announcements

Journal

- May be taken into tests and exams
- Can be used to record: lecture notes, thoughts, questions, work log
- Cannot contain material pasted in except,
  - where specifically stated
  - for pasting prints of digital handwritten notes (annotated lecture slides are not allowed)
- Used for official lab write-ups
- Journals will be checked for rule compliance during the tests and exam

Full Guidelines and Resources

- *Note taking and Journal Writing* page on Canvas

At the end of each lecture

- You should spend some time revising your notes, and sharing and discussing the same with your peers.

What is Software Engineering?

Responsible application of theories of software development to provide cost effective construction of software systems that provide value to other people

- “other people” — not software systems that at most the builder will use
- “value to other people” — not software systems that deliver functionality that is of no use (e.g., faulty, hard to use, not actually useful) to the customer
- “systems” — big things and things that integrate with big things, not little things

What is Software Engineering?

Responsible application of theories of software development to provide cost effective construction of software systems that provide value to other people

- “cost effective” — not software systems that cost a lot to build or cost a lot over their lifetimes (design has relevant quality attributes)
- “theories of software development” — not theories of artificial intelligence, theories of computation, theories of signal processing
- “application of theories” — not development of theory, not (just) presentation of theory
- “responsible” — the engineers involved in the construction accept responsibility for the resulting system and the consequences of its use

What is Software Engineering?

Responsible application of theories of software development to provide cost effective construction of software systems that provide value to other people

- “other people” — not software systems that at most the builder will use
- “value to other people” — not software systems that deliver functionality that is of no use (e.g., faulty, hard to use, not actually useful) to the customer
- “systems” — big things, not little things, or things that integrate with big things

What is Software Engineering?

Responsible application of theories of software development to provide cost effective construction of software systems that provide value to other people

- "cost effective" — not software systems that cost a lot to build or cost a lot over their lifetimes (design has relevant quality attributes)
- "theories of software development" — not theories of artificial intelligence, theories of computation, theories of signal processing
- "application of theories" — not development of theory, not (just) presentation of theory
- "responsible" — the engineers involved in the construction accept responsibility for the resulting system and the consequences of its use

Software Quality Assurance

- What is Software?
- What is Quality?
- What is Assurance?

What is Software?

- Agenda
- People
- Communication
- Course Schedule
- Examinable Material
- Journal
- Software Engineering
- Quality Assurance
- Software
- Quality
- “ilities”
- Software Lifecycle
- Software Development Process

What is Software?

- source code
- compiled (or object) code
- executable code
- configuration files
- images/media
- other input data
- user documentation
- developer documentation
- test code
- design documentation
- requirements documentation
- business case

What is Software?

- anything not hardware in a computer system that is of value to a stakeholder of the system
- stakeholders include:
  - users (of different kinds)
  - customer or whoever pays the bill
  - developers (current and future)
  - managers of developers/users/customers
  - regulators, those who check for legal compliance
  - customers of users

What is Quality?

- Agenda
- People
- Communication
- Course Schedule
- Examinable Material
- Journal
- Software Engineering
- Quality Assurance
- Software
- Quality
- “ilities”
- Software Lifecycle
- Software Development Process

What is Quality?

- “contains no bugs” (whatever “bug” means)
- “works properly” (whatever “properly” means)
- does what the customer wants
  - which customer? the person paying the bill?
  - not the user?
  - what about other stakeholders?
  - what about what the stakeholder needs?

What is Quality?

- “contains no bugs” (whatever “bug” means)
- “works properly” (whatever “properly” means)
- does what the customer wants
  - which customer? the person paying the bill?
  - not the user?
  - what about other stakeholders?
  - what about what the stakeholder needs?

What is Quality?

- “contains no bugs” (whatever “bug” means)
- “works properly” (whatever “properly” means)
- does what the customer wants
  - which customer? the person paying the bill?
  - not the user?
  - what about other stakeholders?
  - what about what the stakeholder needs?
- Our meaning —

Everyone* associated with the software system has as good experience as possible

*including developers

“ilities”

- fitness for use
- correctness — does it contain faults
- reliability — how likely is it to fail
- performance — is it fast enough/doesn’t need too much space
- maintainability — how easy is it to do maintenance in the future
- modifiability — how easy is it to change
- availability — how likely is it to be around when the user wants it
- reusability — how easy is it to reuse in different systems
- usability — how painful is it to use

What is Quality Assurance?

- “book” meaning — Making sure that the *process* for developing software has good quality (fitness for purpose) — tries to reduce mistakes
- common interpretation — Making sure that the *product* has good quality

What is Quality Assurance?

- “book” meaning — Making sure that the *process* for developing software has good quality (fitness for purpose) — tries to reduce mistakes
- common interpretation — Making sure that the *product* has good quality
- Our meaning — Making sure that the client is satisfied with the result

The Life of Software

The Software “Lifecycle”

Requirements	Determine what the system should be (functional and non-functional)
Specification	Describe requirements precisely
Architecture	Make high-level decisions about design
Detailed Design	Make low-level decisions about design
Implementation	Create executable system
Testing	Check created system is “correct”
Deployment	Install system in customer’s environment
Operation	Support day-to-day usage as contracted
Maintenance	Keep system useful to customer
Retirement	

The Life of Software

The Software “Lifecycle”

Requirements (“Analysis”)	SOFTENG 750
Specification	SOFTENG 752 (SOFTENG 211)
Architecture	SOFTENG 325 (SOFTENG 370, SOFTENG 364, COMPSYS 201)
Detailed Design	SOFTENG 251 (SOFTENG 370, SOFTENG 364, COMPSYS 201)
Implementation	SOFTENG 250, SOFTENG 211 (SOFTENG 251, SOFTENG 370, SOFTENG 364, COMPSYS 201)
Testing	SOFTENG 254 (SOFTENG 251, SOFTENG 325)
Deployment	(SOFTENG 206, 306, 700)
Operation	(SOFTENG 206, 306, 700)
Maintenance	(SOFTENG 206, 306, 700)
Retirement	

Software Development Process

- *process* — an ordered series of steps to accomplish a set of tasks
- activities that use resources, are subject to constraints
- descriptive or prescriptive
- supports *risk management* by capturing “best practice”
- provides natural units on which to base quality assurance (activities)
- Often-mentioned examples:
  - Waterfall — idealised view of software development based on lifecycle
  - Spiral — early process that emphasised *iteration* and risk management (Barry Boehm)
  - Rational Unified Process (RUP) — more recent (and different) also emphasising iteration and risk management, often associated with UML (Philippe Kruchten)
  - Agile — plan to react to change, e.g. eXtreme Programming (XP) takes iteration (and other things) to the extreme (Kent Beck)