CITS5501 Software Testing and Quality Assurance
Introduction

Unit coordinator: Arran Stewart

Highlights

- This lecture gives a big picture view of what we will cover and why.
- The big question: what makes software high quality? And how can we repeatedly ensure we produce software of high quality?

Testing and quality assurance techniques

- Testing and quality assurance techniques range from basic procedures every developer should know (unit testing, use of test frameworks), through to techniques that are often only used for high-assurance software (formal methods).

Areas where testing and quality assurance techniques are used (2)

Some of the more interesting examples:

- *Verifying* that software meets particular safety or security properties – an example is the provably secure seL4 Microkernel
- Model checking – Microsoft uses model checking techniques to test that driver code (which runs with high privileges) is using the API correctly
- Enforcing properties with rich type systems:
  - Memory safety
  - Microsoft's research Singularity OS
  - Encoding protocols using types (session types)
- Extracting programs from proofs (using e.g. proof assistants like Agda)

Unit Information

Unit Coordinator: Arran Stewart
Contact: arran.stewart@uwa.edu.au
Phone: +61 8 6488 1850
Office: Rm G.08 CSSE Building
Consultation: Thurs 4-5pm

Contact Hours:

- Lecture: Tues 2-4pm CSSE seminar room 1.24
- Workshop: Wed 10am, starting week 2

Unit website

Unit webpage: http://teaching.csse.uwa.edu.au/units/CITS5501/

All content for the unit (with the exception of lecture recordings) will be delivered via this website, *not* the LMS.

Textbook and other readings

You will need access to the following:

- Ammann and Offutt, *Introduction to Software Testing*, 2nd ed, Cambridge University Press, 2016
- Either of:
  - Pressman, *Software Engineering: A Practitioner's Approach*, 9th ed., 2019.
  - Sommerville, *Software Engineering*, 10th ed., Addison Wesley, 2015.

  (Earlier editions should be fine as well.)

Textbook and other readings, cont'd

- The UWA library has copies of all the textboos
- They can also be bought fairly cheaply online – see the unit website for suggestions.

Lectures

It's recommended that you review the relevant textbook chapters for lectures *before* attending the lecture.

A detailed topic schedule is available on the website.

Workshop details

These will be a combination of group-work exercises, and using particular software tools covered in the lectures.

The venue is lab 2.01, but feel free to bring a laptop if you have one.

Assumed knowledge

- Completion of 12 points of programming-based units is a prerequisite for enrolling in CITS5501.
- In particular, I assume that you are familiar with programming in at least one object-oriented language (typically either Python or Java).
- Portions of the unit that require coding can be done in either language.
- If you have done CITS1001, you'll be familiar with JUnit; if not, you might want to look at the "Materials and reading" page of the website where there are links to the documentation for JUnit.
- It's assumed that you should be able to work out the meaning of simple programs written in either language.

Assessment

3 short pieces of work relating to workshop material:

- 5% each in weeks 3, 6 and 9.
- These will usually be very simple (usually less than a page of work) and marked out of 5 – they are just to ensure that you're keeping abreast of the material.
- They'll be available a day before the workshop, and you can either hand your work in at the workshop, or submit online via cssubmit.

Assessment, cont'd

Project:

- Worth 35%, consists of code and a report.
- Task is to design and execute a testing and validation process for a software system.

Assessment, cont'd

Exam:

- Worth 50%.
- 5 questions with a total value of 100 marks.
- The workshop exercises are a very good guide to what will be on the exam.

Lecture schedule

- General overview of topics:
  - Testing & testing methodology
  - Quality assurance
  - Formal methods and formal specifications

Software quality - what is it?

- What are some ways that software can be good?
  And what are some ways that it can be bad (or, less than ideal)?

Ensuring quality software

- There are multiple aspects to building quality software:
  - Organisational processes – How does the software team operate?
  - Process and software standards – Are particular standards used?
  - Process improvement – How is success in building quality software measured and improved?
  - Requirements specification – How do we work out what software we should be building? And how do we work out whether we built the right software?
  - Formal methods – Ways of proving that software is correct
  - Testing – identifying and correcting defects

The software “illities”

There are many features that contribute to the success of software, and very few relate to correctness. . . :

- Usability
- Maintainability
- Scalability
- Reliability/Availability
- Extensibility
- Securitability (sic)
- Portability

Why test?

Testing is not just about demonstrating correctness. Testing is used in several ways in modern software development:

- Unit tests – Ensuring functional units are correct
- Integration testing – Ensuring components work together
- Acceptance testing – Getting paid at the end of the day
- Regression Testing – Don't break the build!
- Test Driven Design – "Test-first" software process
- Tests as documentation – Complete test suites are often the most accurate documentation a project has.