Software Testing, Quality Assurance and Maintenance
SE465, Winter 2019, version 0

Patrick Lam

Brief Overview

As you have no doubt discovered, software never works right from the start. A key technique for getting more acceptable software is testing. Organized testing can help identify problems in software systems, enabling developers to fix these problems. This course will introduce software testing techniques; while it's not my goal to produce testers, you should at least be conversant with up-to-date testing methodologies and techniques. And easier-to-test software is better software.

In this class, we will also touch on software maintenance. While we greatly (over?) emphasize design in engineering school, maintenance consumes a large fraction of today's software development resources.

General Information

github repo: `git@github.com:patricklam/stqam-2019.git`
Lectures: MW 2:30-3:50, MC 4020
Tutorials: not generally used, but scheduled for M 10:30-11:20, MC 4020

Instructor:

Prof. Patrick Lam
Office: DC-2539
Drop-in Hours: W 11:30-12:20 (or by appointment)
Email: `patrick.lam@uwaterloo.ca`
Phone: Use email instead!

Teaching Assistants:

Meet Bhavsar	`m4bhavsa@edu.uwaterloo.ca`	SE 2019
Jason Milasincic	`jmilasincic@edu.uwaterloo.ca`	SE 2019
Parsa Pourali	`ppourali@uwaterloo.ca`	PhD student
Michael Socha	`msocha@edu.uwaterloo.ca`	SE 2019

Course Description

Objectives. We hope that you will learn how to test as a developer, thus making you a better developer. Good software systems are easy to test and we'll talk about how testability affects design.

- You will be able to create and evaluate test suites for reasonably-sized software systems.
- You will learn how to use and write tools for software maintenance and verification (particularly automated testing tools).

Topics. This course is a moving target and this may be the last time the course is taught like this. I think this course will be quite similar to the 2017 offering, although I would like to go more SE 212-like in the tools unit, so that might change. As usual, exams may be different from other offerings, but 2017 is a good bet.

- Introduction and definitions (defects, faults, failures) (1 week)
- *Defining Test Suites*—finding interesting inputs and knowing when to stop looking (4 weeks)
  How do you generate test suites?
  - open-ended exploratory testing;
  - statement/branch coverage;
  - graph-based models of program state; integrating design documentation;
  - automatically generating inputs: grammars, fuzzing.

  How good is your test suite?
  - coverage
  - mutation
- *Engineering Test Suites* (3 weeks)
  - JUnit (unit testing, end-to-end testing, regression testing)
  - Selenium and other web-based testing
  - mock objects
  - refactoring tests
  - refactoring code to be testable
  - continuous integration
  - flaky tests
- *Tools for Verification & Validation* (3 weeks)
  I plan to include some intuition for why these tools work. I hope to go more formal-verification-ish here than the things listed below.
  - concept: static vs. dynamic analysis
  - static approaches: type systems, immutability, compiler warnings, software model checking, JML tools. Coverity, Facebook Infer, FindBugs, aComment
  - dynamic approaches: valgrind/Clang address sanitizer, assertions, concurrency detectors: races, missing locks
  - human-based approaches: code review, bug reporting
- Bonus: debugging and the scientific method (1 week)
  (reference: Andreas Zeller. *Why Programs Fail: a Guide to Systematic Debugging.*)

Reference Material

Course notes (posted to the git repository) are your best bet.

Evaluation

This course includes assignments, a midterm, a course project, and a final examination.

3 individual assignments	20%	(6 2/3% each)
Course project (in groups, up to 3/group)	15%	
Midterm	15%	
Final exam	50%	

The midterm and final exams will be open-book, open-notes.

Schedule. Assignment handin will be done through the git server at `git.uwaterloo.ca`.

January 14	A1 out
January 28	A1 due, A2 out
February 25	A2 due, A3 out
February 28	Midterm (6:30-8:20PM, RCH 301/307)
March 25	A3 due
April 5	Last day of lectures; project due
Exam period	Final exam

Policies

Group work. The project will be done in groups. You may discuss assignments with others, but I expect each of you to do the assignment independently. I will follow UW's Policy 71 if I discover any cases of plagiarism. I will not use turnitin.

Lateness. You have 2 days of lateness to use on assignment submissions throughout the term. Each day you hand in an assignment late consumes one of the days of lateness. If you consume all of your late days, assignments that are still late will get 0. Missed assignments get 0. e.g. you may hand in A1 one day late and A2 one day late if you hand in A3 on time. Or you can hand in A1-A2 on time and A3 two days late.

Academic integrity:	`http://uwaterloo.ca/academicintegrity/`
Petition & Grievance:	`http://secretariat.uwaterloo.ca/Policies/policy70.htm`
Discipline:	`http://secretariat.uwaterloo.ca/Policies/policy71.htm`
Penalties:	`http://secretariat.uwaterloo.ca/guidelines/penaltyguidelines.htm`
Appeals:	`http://secretariat.uwaterloo.ca/Policies/policy72.htm`
AccessAbility:	`https://uwaterloo.ca/disability-services/`