Software Testing, Quality Assurance and Maintenance
SE465/CS447/ECE453, Winter 2017, version 1

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

Course web page:	`https://patricklam.ca/stqam`
github repo:	`git@github.com:patricklam/stqam-2017.git`
Lectures:	MWF 11:30-12:20, MC 4020 (SE465)
	MWF 15:30-16:20, EIT 1015 (ECE453/CS447)
Tutorials:	not generally used, but scheduled for F 17:30-18:20, EIT 1015

Instructor:

Prof. Patrick Lam
Office: DC2597C
Office Hours: M 12:30-13:30, or by appointment
Email: `patrick.lam@uwaterloo.ca`
Phone: Use email instead!

Lab Instructor:

Irene Huang
Email: `yqhuang@uwaterloo.ca`

Teaching Assistants:

Parsa Pourali	`ppourali@uwaterloo.ca`
Gursharan Sandhu	`g27sandh@uwaterloo.ca`
Mier Ta	`m2ta@uwaterloo.ca`
Xinye Tang	`x58tang@uwaterloo.ca`
Song Wang	`s446wang@uwaterloo.ca`
Jun Zhao	`j283zhao@uwaterloo.ca`

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. Over the past few years, we've been working to improve this course. Be aware that, when studying for exams, this year's offering differs significantly from previous offerings. Near exam time, I'll indicate the types of exam questions that will still be relevant and those that won't be.

- 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 documentations;
  - 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. List of tools may change.
  - 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 `ecgit.uwaterloo.ca`.

January 16	A1 out
January 30	A1 due, A2 out
February 17	Midterm (6:30-8PM)
February 27	A2 due, A3 out
March 24	A3 due
April 3	Last lecture; project due
Exam period	Final exam

You can also find the dates on the following Google calendar:
`juj4cn4vtkeivufdvd6iqo6h8o@group.calendar.google.com`

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/`