Teaching an Active-Participation University Course in Software Reliability and Testing

Laurie Williams
williams@csc.ncsu.edu

Abstract

A large percentage of software development costs are spent on software reliability and testing. However, many practicing software engineers and graduate students in computer science have never taken a course in software reliability or software testing. A graduate-level software engineering course at North Carolina State University provides instruction in these topics to better prepare current and future software engineers for the software reliability and testing challenges. The course takes place in a laboratory setting such that students can learn testing and reliability theory and apply this theory immediately.

1. Perspective

Computer science graduate students and professional software engineers should understand the principles behind software reliability and testing. Such knowledge will aid them in building solid, testable code, in validating and verifying this code, and in engineering reliability into their projects. As such, North Carolina State University (NCSU) offers a graduate level class that instructs in software reliability and testing. The resources for this class can be found at the OpenSeminar in Software Engineering[1].

2. Course content

The first part of the course focuses on software testing. The students learn and apply fundamentals in black and white box testing and automation tools of the same. As the language of the course has been Java, students use JUnit[2] for white box test automation and FIT[3] for black box test automation. Using these automation tools, student write test cases explicitly using strategies such as equivalence class partitioning, boundary value analysis, data flow testing, control flow testing, scenario-based testing, security testing, and diabolical testing. Students learn to write the minimum number of test cases to find the maximum number of faults. They assess the quality of their test suites via mutation testing and coverage tools

In the second part of the course, students learn software reliability engineering and reliability estimation. In this portion of the course, we use John Musa's *Software Reliability Engineering*, second edition. Lecture slides based upon this book are posted on the OpenSeminar as well.

3. Active learning

The students apply the testing and reliability theory learned in the class in several ways. First, the course takes place in the NCSU Laboratory for Collaborative System Development[4]. Therefore, the twice/week classes are intermingled between lecture-style instruction and hands-on exercises on the computers. For example, instruction is provided on unit testing, coverage principles, and JUnit. Then, a short exercise for developing JUnit with high coverage is completed.

Additionally, the students apply their software reliability and testing knowledge via a semester-long project. In five iterations, students develop the project. Code is synchronously developed with automated and "strategized" black and white box test cases and through manual test plans. The programs are written such that the operations are logged, enabling a retrospective analysis of the actual operational profile.

In the final phase of the project, students chose a subset of their manual tests that can be run in one 70-minute class period, based upon an estimated operational profile. During two class periods, student teams swap completed projects and run their 70-minute test on other teams' projects. Test failures are documented to provide feedback to student teams on defects that have escaped all testing.

[1] http://openseminar.org/se/courses/41/index/screen.do
[2] http://junit.org/index.htm
[3] http://fit.c2.com/fit/
[4] http://collaboration.csc.ncsu.edu/laurie/LCSD.htm