Software Testing, Quality Assurance & Maintenance (ECE453/CS447/SE465): Project (v1)

Patrick Lam

Due: April 5, 2010

The course project allows you to demonstrate that you've achieved the objectives of this course: creating and evaluating test suites, using tools for software maintenance and verification, and working with reasonably-sized software systems.

You may test your fourth-year design project (if applicable), a software system of your choice (but check with me first), or MegaMek[1].

You may form groups of up to 4. Submit one project per group. (If you have more than 4 people in your fourth-year design group, submit 2 projects.)

Unit Tests (60%)

In the first part of the project, your task is to create a suite of unit tests for a unit; by unit, I mean one class or several related classes, containing about 500 lines. You'll need to choose a unit which is reasonable to test. Write about what you chose to include in your test suite. I expect between 1 and 2 pages of discussion about your test suite; you will probably want to use concepts from the course to defend your test suite, although I'm open to other reasonable arguments too. (A suitable class to unit test in MegaMek might be `MoveOption`—actually, this class is quite complicated, and you would do well to focus your efforts on a particular aspect of this class. Profiling might help you identify important methods.)

[1]`http://megamek.sourceforge.net`

Consider superficially covering all of the methods in your unit test and then testing the most interesting methods in more depth. However, do not take this suggestion to be mandatory; you can tackle this however you'd like.

System Tests (40%)

(10%) Briefly document the design of your system: describe the components which will interact in the three scenarios to follow. (10% each) Describe three system tests. One of the tests should be a normal system use case, while a second test should be an unusual use case. I am not constraining your third test. Include at least one case where the actual behaviour of the system differs from the expected behaviour of the system (i.e. a bug), and briefly discuss (a few paragraphs) how you might fix this bug. (Better yet, propose a patch.) You may cite and use an existing bug, as reported in the system's bug tracker, for one of the test cases.

Describe how each of your tests exercises the system components, as you've outlined previously, and state how these tests exercise the couplings between these components. Also include the input, expected output, and actual output.

Structure of Descriptions. Here are some points to include in your test case specifications:

Test case ID:	
Test case title:	
Test objective:	
Assumptions:	
Initial conditions:	(what needs to be true before the test can run)
Post conditions:	(what is true after the test succeeds)
Test steps:	
Clean up:	
Pass/fail criteria:	
Automatable:	(yes/no)

Deliverables

Please electronically hand in the following:

- unit code, executable unit tests, and descriptions;
- system structure, test specifications, execution reports, and discussions;
- patch, if appropriate.