University of Toronto
Department of Computer Science

CSC302F – Engineering Large Software Systems

March 21, 2012
Prof. Steve Easterbrook

Assignment 4: Build and Test a New Feature

Due Date: 10:20am, Thursday, April 5, 2012
(i.e. within 10 minutes of the start of the lecture)

This assignment counts for 15% of the final grade

Build and test a new feature for the open source application you worked with in Assignment 3. You should build enough of the functionality of the new feature to offer users some additional value, but you are not required to build all of the functionality suggested in the original feature requests. However, you will be expected to fully test and debug the functionality that you do build, and to demonstrate that your verification process was *thorough*. You are encouraged to identify and use automated tools to help: e.g. code coverage analysis tools, automated testing tools, static analysis tools, etc.

The project is to be carried out in your assigned teams. Each team will submit one report.

I. Doing the Assignment

This assignment has 8 steps. They are:

1. *Review the analysis you performed for the previous assignment*. Check that your understanding of the requirements is correct, and that your effort estimates still seem reasonable. Then select an appropriate set of functions (related to the feature request) to implement for this development cycle.
2. *Decide on your development process*. How will you organize and keep track of the implementation effort? Decide whether any of the software process models mentioned in the course will help you (e.g. SCRUM, XP, ICONIX, RUP). For example, consider whether to use the following strategies: test case first, pair programming, daily scrum meetings, burndown charts, daily smoke test, etc.
3. *Build and verify the functions*. Use any appropriate verification strategies (informal and formal reviews, static and dynamic analysis, etc). Keep track of the number and types of defects discovered through each technique.
4. *Build an automated test suite your new functions*. Note: depending on your development strategy, you might do this before, during or after implementing the functions themselves.
5. *Select one of most complex of the classes you have implemented (or modified) for additional white box testing*. You should demonstrate 100% code coverage of this class, by using a suitable testing strategy (e.g. structured basis testing, MC/DC, inheritance coverage, etc)
6. *Write a short user guide to explain how to use your new feature*.
7. *Write a report* that describes your implementation and verification activities. Be sure to document your development process, and comment on how well the process worked for you.
8. *Document your teamwork* using the online peer evaluation system on the course website.

II. Change requests

You should continue to work on the same feature you selected in assignment 3. If you encounter difficulties that lead you to conclude the implementation is infeasible, then change your project plan accordingly. Making sensible decisions about what (and how much!) to implement is part of this assignment.

III. What to Hand In

Hand in your report at the start of your lecture on the due date. *Reports not handed in within the first ten minutes of the lecture will be treated as late.*

The report should not exceed twenty (20) pages (not counting cover pages and appendices). Do not include source code or detailed test cases in the report, other than selected excerpts needed for illustration purposes – the source code and test suites should be checked into your DrProject code repository. Be sure to indicate clearly which version of the code in your repository is the final version of the program, and also indicate the location of the final version of the test suite. Your TAs may need to run your code and your test suite during the marking of this assignment.

Your submitted report should include the following items:

1. A description of the software development process you chose to use, including all tools and techniques. Be sure to justify your choices.
2. An analysis of your implementation and verification procedures. Indicate which techniques you applied, and comment on their effectiveness. You should include data on the numbers and types of defect found through each verification technique, and a commentary on the resulting quality of your implementation.
3. A ***brief*** user guide for the new feature(s) you have added. You do not need to repeat material from the original user guide, but should clearly show where your new material should be inserted into the existing guide.
4. A post mortem on the course work for all four assignments this term, including a review of the tools and techniques you have applied, how well they worked (or didn't!), and lessons learned. Be sure to identify both positive and negative lessons: i.e. both things that worked well and things that went wrong, and identify techniques you are likely to want to use again on other software development projects.

Written Presentation Requirements

Be sure to include a cover page indicating the name of your team, the names of all team members, title of work, course, date and tutor's name. Assignments will be judged on the basis of visual appearance, the grammatical correctness and quality of writing, and the visual appearance and readability of diagrams, as well as their contents. Please make sure that the text of your report is well-structured, using paragraphs, full sentences, and other features of a well-written presentation. Use itemized lists of points where appropriate. Text font size should be either 10 or 12 point.

IV. Marking Scheme

Your tutor will mark your assignment. If you have questions about a marked assignment, you should first ask your tutor before/after a tutorial. If you don't get satisfactory answers, you should talk to your instructor.

Marks for this assignment will depend on the following factors:

Description of your choice of process, techniques and tools (20%): Did you clearly describe the criteria you used for your decision? Are the criteria appropriate? Is the choice sensible, given the criteria? Did you investigate existing resources (e.g. free testing tools) that you could use to help you?

Quality of your new version of the application (20%): Do the set of functions you have added offer some significant extra value to the users? Is it clear which version in your repository is the finished one?

Was it easy to build and install? Do the new functions work? Are they usable? Are they robust? Is the user guide clear?

Quality of your verification process (20%): Did you build an automated test suite? Does it work? Does it cover the software adequately? Did you use additional verification strategies beyond developer testing? Did you demonstrate 100% coverage for at least one of your classes? Is the coverage criteria you applied appropriate? Did you collect and analyse data on defects found by your various verification strategies? Did you draw appropriate lessons from this data?

Quality of your post mortem (20%): Did you review the development processes you used on the various assignments throughout the course? Did you review all the tools and techniques? Did you identify lessons learnt, both positive and negative? Are the lessons authentic and relevant?

Presentation (20%): The style of your presentation, including language, grammar, clarity of the presentation, layout and legibility of the diagrams, etc. (10% - Language; 10% - Style and clarity)
