COSC 442

Software Quality Assurance & Testing

Spring 2015

**Final Project – Improving the Quality of a Large, Open-Source Software Project**

*"It makes me feel guilty that anybody should have such a good time doing what they are supposed to do."*

- Charles Eames

**Directions:** This final project will provide you the opportunity to independently improve the quality of a large, open-source software project. For this project, you may work in groups of up to 3 students, of your choosing. Unlike previous projects, this final project will be more of a self-directed project (i.e., I won’t tell you exactly what to do). For this project, your group is expected to create and maintain a ***new*** repository as well as an issue tracker that will document your team’s actions in improving the software quality of the open-source software project. **Please read through all instructions prior to starting**.

**Goals:** The intention of this project is to allow you to demonstrate that you have achieved the objectives of this course:

* Understand and communicate basic software testing terminology, principles and techniques
* Develop unit, module, subsystem, integration and system test cases for a variety of systems
* Apply quantitative, technical and practical testing methods and tools for testing and modifying evolving software
* Design and implement software for easier testing, maintainability and reuse
* Evaluate software for change impact and validate software changes
* Recognize the roles and responsibilities of a software test engineer

in the context of a large software project. Throughout this project, you will be utilizing many of the techniques (e.g., refactoring, documenting, static analysis, code complexity analysis, coverage, testing, etc.) and tools (e.g., Eclipse, JUnit, Subversion, EclEmma, Google CodePro Analytix, Javadoc, Fitclipse, etc.) that were covered separately, as well as other techniques and tools that you find on your own, in a self-directly manner. In doing so, you should further develop skills in program comprehension, testing and software quality assurance as well as practical experience in the role of a real quality assurance expert in a real-world project setting.

**Deadline:** Tuesday, May 12, 2015 11:59 pm with all documentation and source code checked into your repository, except as noted.

**Language/Environment Requirements:** Java within the Eclipse IDE and a Subversion version control system of your choice. Any other tools / approaches used are to be decided and documented by your group.

**Task 1 – Setting up a repository server and issue tracker.** As we have done throughout the semester, the final project will utilize a Subversion repository server specifically dedicated to this project. Since Google Code is [shutting down](http://google-opensource.blogspot.com/2015/03/farewell-to-google-code.html) and no longer allowing the creation of new repositories, your group will need to set up a Subversion repository using a different service. There are several options available, including SourceForge (<https://sourceforge.net/>), Assembla (<https://www.assembla.com/subversion/>) RiouxSVN (<https://riouxsvn.com/>) and GitHub, kind of, (<https://help.github.com/articles/support-for-subversion-clients/>) among others. Whichever you select, you should add me ([cosc442spring2015@gmail.com](mailto:cosc442spring2015@gmail.com)) as a project committer/member so that I can access your project from my computer. Although we have used a repository all semester with a single user, you will want to familiarize yourself with the team collaboration aspects (e.g., branching, merging and synchronizing) at <http://realsearchgroup.org/SEMaterials/tutorials/subclipse/> and <http://php.sabscape.com/blog/?p=380>. Additionally, your group is to set up and utilize an issue tracker to identify, document and assign quality assurance / testing tasks. There are a number of issue tracking online applications, including Google Code, SourceForge and others listed at <http://mashable.com/2014/02/16/bug-tracking-apps/>. Your group may select any of these. Again, if needed, you should add me ([cosc442spring2015@gmail.com](mailto:cosc442spring2015@gmail.com)) as a project member so that I can access your issue tracker.

*Once you complete this task, someone from your group should email me at* [*jdehlinger@towson.edu*](mailto:jdehlinger@towson.edu)  *with the following information: group member names, repository home and subversion URL, and issue tracker home and URL.*

**Task 2 – Project Selection.** The open-source project that your group will work on will be solely your choice, based on the following conditions: it must be an open-source game from SourceForge.net that is 1. in either alpha/beta status; 2. has been recently updated; 3. can be run on a Windows OS; and, 4. is primarily written in Java. For your convenience, you can browse projects that fit these conditions at <http://sourceforge.net/directory/games/developmentstatus:alpha/language:java/os:windows/freshness:recently-updated/> and <http://sourceforge.net/directory/games/developmentstatus:beta/language:java/os:windows/freshness:recently-updated/>. When selecting the project that your group will work on, take care in looking at the project, downloading the code and compiling/building it. Your group will be solely responsible for being able to compile/build/execute the project when submitting the final project.

*Once your group has completed this task, email me (*[*jdehlinger@towson.edu*](mailto:jdehlinger@towson.edu)*) with the project that was selected along with the SourceForge URL for approval.*

**Task 3 – Prioritizing Components.** The project selected is likely a medium-to-large piece of software consisting of thousands of lines of code, hundreds of methods and dozens of classes. Often times, because of time/resource constraints, quality assurance groups will have to be selective in what parts of software they target to improve/test. This selection may be done in a number of ways (e.g., most used class/package using software profiling tools, most complex class/package using complexity measurements, largest class/package, etc.). To focus your quality assurance efforts, your group should prioritize the classes to improve/test. There are a number of Eclipse plugins that you might want to consider using to help you make this determination, including [Google CodePro](https://developers.google.com/java-dev-tools/codepro), [JDepend](http://andrei.gmxhome.de/jdepend4eclipse/), [Metrics](http://metrics.sourceforge.net/) and [X-Ray](http://marketplace.eclipse.org/content/x-ray-software-visualization). Whatever prioritization your group selects should be documented and rationalized in the Tactical Test Plan (Task 4). At the completion of this task, as a group, you should have prioritized the packages/classes that you will test, improve, document and/or refactor as a part of improving this software’s overall quality.

**Task 4 – Developing and Documenting a Software Quality Assurance Plan.** As a part of planning for and documenting the activities your group will undertake as a part of improving the software quality, your group will prepare and submit a Software Quality Assurance Plan (template provided on Blackboard). This document should thoroughly document your plan for how you will improve the software (Task 5), the packages/classes selected to improve/test and the rationale (Task 3) and ***all*** activities carried out throughout this project to test/improve the software (Task 5).

Many of the sections in the Software Quality Assurance plan can be prepared prior, or in parallel, to starting to test/improve the software and some will be updated throughout (i.e., this will be a *live* document throughout your project that will likely need to be updated while working on Task 5).

**Task 5 – Improving the Software Quality / Testing.** Throughout the semester, we talked about a number of ways to improve/test software quality, including refactoring, documenting and testing; a number of testing coverage criteria; as well as tools to help us (e.g., Eclipse, Google Code, JUnit, Subversion, EclEmma, Google CodePro, Javadocs, Fitclipse, etc.). In this task, you will, as a group, decide which tools are appropriate for improving the project you selected.

In addition, you must utilize at least *two* additional/new tools per person not used in a project in class. There are a number of excellent, candidate tools that I have found useful for improving/testing the software, including [FindBugs](http://findbugs.sourceforge.net/), [CheckStyle](http://eclipse-cs.sourceforge.net/), [Eclipse Memory Analyzer](http://www.vogella.com/tutorials/EclipseMemoryAnalyzer/article.html) and [PMD](http://pmd.sourceforge.net/). Each of these tools statically analyzes Java code checking for common mistakes/bugs. Additionally, you may want to check out [Google WindowTester Pro](https://developers.google.com/java-dev-tools/wintester/html/index), a very nice automated GUI testing tool. The [Eclipse Marketplace](http://marketplace.eclipse.org/) literally has over 1.5 million Eclipse plugins to help developers, including a number of testing plugins for Java projects.

Once your group has decided on the tools and approach you will take to improve/test the software, you should assign tasks according to each person’s skill and work on improving/testing the packages/classes you selected from Task 3 in order of priority. Note that if you are working in a group, the process to use Subversion is slightly different (see <http://agile.csc.ncsu.edu/SEMaterials/tutorials/subclipse/>) – rather than “committing” your code immediately, you will want to synchronize it with the repository before committing it (if you commit it, you may write over a group member’s work).

**Task 6 – Peer Review.** Each group member, independently, should download the Peer Review from Blackboard (Projects/Final Project), complete the form and submit it to me via Blackboard prior to the project deadline. I will use these peer review forms to inform the score I assign for individual effort.

**Grading:** This project will be graded as follows:

* 50% Improvement / Testing – based on test cases, refactorings, documentations, subversion quality, tools used, etc.
* 30% Software Quality Assurance Plan – based on professionalism, completeness, documentation, clarity
* 10% Weekly Progress Meetings – based on informal, weekly group progress meetings
* 10% Personal Effort – based on Git/Subversion commits, peer reviews, Responsibilities (from the Tactical Test Plan)

**Submission:** All projects’ Software Quality Assurance Plans must follow the Professionalism description from the Course Syllabus. You must submit your plan (in both docx and pdf file formats), the code, test cases and any other artifacts for the final project into your project repository. Failure to submit the project following these guidelines may result in your project not being graded.