COSC 442

Software Quality Assurance & Testing

Spring 2018

**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 4 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). I have created a Slack channel for general discussion for this project; I encourage you to use it to discuss problems encountered. For this project, your group is expected to create and maintain a ***new*** repository as well as the issue tracker on GitHub 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 may utilize many of the techniques (e.g., refactoring, documenting, static analysis, code complexity analysis, coverage, testing, etc.) and tools (e.g., Eclipse, JUnit, Git, EclEmma, Google CodePro Analytix, Javadoc, Pitclipse, Selenium, etc.) that were covered separately, as well as other techniques and tools that you find on your own, in a self-directed 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:** Wednesday, May 16, 2018 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 Git/GitHub version control system. 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 Git/GitHub repository server. Please name your repository with an appropriate name ending with ­­*–finalproject* so that I can search/find it easy in my list of repositories. Your group should create a new repository, and add each group member as a contributor. You should additionally add me ([cosc442spring2018@gmail.com](mailto:cosc442spring2018@gmail.com)) as a project committer/member so that I can access your project. 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 <https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging>. Additionally, your group is to set up and utilize an issue tracker to identify, document and assign quality assurance / testing tasks. GitHub has a pretty good issue tracker that you should utilize, see <https://guides.github.com/features/issues/>.

Note that working in a group using GitHub has a different workflow than when working individually; to become familiar with using GitHub in a team – check out <https://guides.github.com/introduction/flow/>, <https://www.youtube.com/watch?v=61WbzS9XMwk> and/or <https://code.tutsplus.com/articles/team-collaboration-with-github--net-29876> – 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 2 – Project Selection.** The open-source project we will use this semester will be FreeCol 0.11.6. (website: <http://www.freecol.org/>; SourceForge: <https://sourceforge.net/p/freecol>), a “turn-based strategy game based on the old game [Colonization](http://www.freecol.org/about.html#original), and similar to Civilization”. FreeCol is under active development (latest release was 10/2015) with a pretty good developer community (see <http://www.freecol.org/community.html> and <https://sourceforge.net/p/freecol/discussion/?source=navbar>), some documentation (<http://www.freecol.org/documentation/>), existing bugs/feature requests (<https://sourceforge.net/p/freecol/_list/tickets>) and some existing test cases (<https://sourceforge.net/p/freecol/git/ci/master/tree/test/src/net/sf/freecol/>) and test guidelines (<https://sourceforge.net/p/freecol/git/ci/master/tree/test/TEST-HOWTO>). I have provided an *Eclipse-ified* version of FreeCol that you can [fork](https://help.github.com/articles/fork-a-repo/) from the course’s GitHub repository at <https://github.com/cosc442spring2018/FreeCol> to bring into your group’s repository.

**Task 3 – Prioritizing Components.** FreeCol is a medium-to-large piece of software consisting of 110,00 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/), [CodeCity](https://marketplace.eclipse.org/content/codecity) 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 (Word template and description provided on Blackboard). Although I have provided a Word template for the Software Quality Assurance Plan, your group’s actual Software Quality Assurance Plan must be documented and submitted using GitHub’s [Wiki](https://help.github.com/articles/about-github-wikis/). This wiki 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).

In parallel with the Software Quality Assurance Plan, your group must establish and maintain an issue tracker that documents the changes you propose to make and follows through with those changes actually made. I strongly suggest that you use GitHub’s native [Issues](https://guides.github.com/features/issues/) feature for this, but you may use any other integration/service provided by Github (see <https://github.com/integrations>).

**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, 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, documented in your Issue tracker, and work on improving/testing the packages/classes you selected from Task 3 in order of priority. All tests and tools used as a part of this project should be integrated into the Ant build of the project. To learn about Ant, see <http://www.vogella.com/tutorials/ApacheAnt/article.html>.

**Task 6 – Adding a New Feature.** Utilizing the inherent creativity/mischievousness of a student mind, the final, major task of this project is that each group must carry-out the creation of a new feature for the game (created as an Issue), implement the feature and provide any necessary tests associated with the feature. The new feature can be anything you want (appropriate please!) – be as creative as you’d like.

**Task 7 – Making a Pull Request.** The primary idea of [forking a repository](http://ericgreer.info/assets/images/fork-me.jpg) is so that a separate developer/development team can make changes to a project and then integrate it into the main repository through a pull request (see <https://help.github.com/articles/about-pull-requests/)>. When your group has completed *all* its work, you should make a pull request to the original <https://github.com/cosc442spring2018/FreeCol> repository.

**Task 8 – 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:

* 40% Improvement / Testing – based on test cases, refactorings, documentations, subversion quality, tools used, etc.
* 30% Software Quality Assurance Plan Wiki – based on professionalism, completeness, documentation, clarity
* 10% New Feature – based on creativity, functionality and quality of associated tests
* 5% Weekly Progress Meetings – based on informal, weekly group progress meetings
* 15% Personal Effort – based on Git commits/graphs, peer reviews, Issue tracker assignments/completions, 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, 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.