4. Technical process

4.1 Methods, Tools and Techniques

Discussions about the processes below will be made in person or by e-mail.

Assignments for creating or altering any particular piece of code, document, etc. will be by volunteering. If someone volunteers for more than they can handle they should simply contact the other members of the group for help.

We will use version control, code reviews, and software tests (unit, integration, system, and regression) as described below.

4.2 Software Documentation

Software documentation will be handled as follows:

a. All code shall be documented with reasonable comments.

Use the following header for classes (comments in react are notated the same as Java):

/\*\*

\* [Brief description]

\* @author

\*/

Use the following header for methods:

/\*\*

\* [brief description]

\* @param [name] [description]

\* @param [name] [description] ...

\* @throws [exception]

\* @return

\*/

This comment style should allow automatic API documentation based on the comments in our source code. [I will update this once I have found an appropriate tool to use with React. Netbeans can do it for sure.]

b. Citations for any code borrowed or adapted from elsewhere should go in the header for the code file and directly above the code where it’s used.

4.3 Project Support Functions

a. Configuration Management and Continuous Integration

i. We will use Github for version control of all documents, code, and materials, except for documents where multiple developers are expected to work simultaneously frequently.

For documents that multiple developers are expected to work on simultaneously frequently and for which there are set due dates, or which were created before the Github project was created, we will use a Google Doc, since it permits simultaneous editing. At the moment, the only documents fitting this criteria are the Requirements document and the document containing the use cases, class descriptions and diagrams.

Developers are expected to work in their local copy of the project and perform the following actions to implement changes to the master branch via Github:

1. Sync your project in the Github desktop app before doing any work on an existing document. Sync is in the upper right.

2. Save your local document after you've made your changes or added a new file.

3. If you are finished working on that document or have made an important change, commit it.

4. If you have more files to work on, GOTO Step 1.

5. ELSE stop.

ii. Since this project is expected to only have one full release at the end of the semester, we will not be using a version control numbering system (either for the software or for documentation).

iii. To avoid duplicating effort or errors in versions of files: If a class (code file) is under review, the developer should not alter the file until the review is complete. If a developer who is not the author of a class wishes to make changes, they should notify the author before and after they’ve made the changes. The process for making changes is in 4.3bi. Github should handle incorporating changes but developers should still be careful to avoid working on the same parts of documents at the same time so communication is vital.

b. Quality Assurance, Verification, and Validation.

Regular code reviews will be performed to validate the project parts at each step to ensure that they conform to the project requirements.

Verification will be done in accordance with the unit, integration, system, and regression tests.

[See test cases document.]