Annalisa Karacay, Elizabeth Cwick, Jake Sojka, and Justin Lin

Sprint 1

April 10, 2023

Project Plan

**Project Goal**

The overarching goal of this group project is to create code to analyze and visualize the reliability of end-to-end message transmissions in our WARP program. Further details on the necessary steps needed in order to achieve this are listed underneath each sprint.

**Sprint 1:**

Requirements:

* UML Sequence Diagram
* Design Documentation
* Updated README

Who will complete what:

Annalisa: Sequence Diagram, JavaDoc

Justin: Sequence Diagram

Jake: Design Documentation

Elizabeth: README, Design documentation

For the first sprint the main goals are to create a UML sequence diagram and a design and project plan. For this sprint we split up into two groups of two to complete the tasks. The UML sequence diagram was created by Annalisa and Justin while the design and project plan documentation were created by Jake and Elizabeth. The UML sequence diagram was created by walking through the existing code to see all the end-to-end reliability for each flow.

**Sprint 2:**

Requirements:

* Updated UML Sequence/Class Diagram
* Visualization Class + Methods
* JUnit Tests for Visualization
* New/Updated Javadoc comments / updated documentation
* Updated Design Documentation
* Updated README

Who will complete what (Decided arbitrarily, subject to change):

Annalisa: JUnit tests for ReliabilityVisualization class and JavaDoc, Update Sequence/Class Diagram

Justin: ReliabilityVisualization and ReliabilityAnalysis methods

Jake: Visualization Class tests and JavaDoc

Elizabeth: JavaDoc for ReliabilityAnalysis and ReliabilityViualization, Update Design Documents and README

For sprint 2, we will need to create high level plans and status in the README file (including who will do what tasks and which artifacts to consider for sprint delivery). To add on, we will make necessary updates to diagrams and design documents as needed. We will begin coding in the ReliabilityVisualization class file to work towards implementing it. In the class, we will develop methods which will work towards creating a visual representation of the end-to-end reliability. Moreover, the implementation of these methods may extend to sprint 3. If further implementation is necessary by sprint 3, we will create comments as well as documentation via JavaDoc for our high-level helper methods explaining what will be done. Additionally, we will add JUnit tests for ReliabilityVisualization. Finally, we’ll adjust plans for sprint 3 in this document.

**Sprint 3:**

Requirements:

* Update UML Diagrams
* Finishing Touches on ReliabilityVisualization and ReliabilityAnalysis
* JavaDoc comments
* JUnit test for ReliabilityVisualization and ReliabilityAnalysis classes
* README Updates

Who will complete what (Subject to change):

Annalisa: Update UML Diagrams

Justin: JavaDoc Comments

Jake: README Updates, JUnit Tests

Elizabeth: Finishing Touches for Code

In sprint 3, we plan to update whatever is necessary, finish implementing any code, and generally patch any rough spots. Updates to the README, UML diagrams, and JavaDoc will be made, and any remaining JUnit tests for ReliabilityVisualization and ReliabilityAnalysis will be developed. Furthermore, all helper methods will be public so they can be tested, with comments explaining whether these methods are normally set to be public, private, or protected.

**Documentation Plan:**

Each of us will have a turn to document our progress and plans regarding the project depending on the sprint. We will document every time we meet and who is responsible for each task. The documentation will be updated throughout the whole process. The README will be updated throughout the process to reflect any and all changes to the code.

**Testing Plan:**

Same as documentation, each of us will test various methods depending on the current sprint. Our plan is to start with creating the tests for the code before actually writing and or modifying the code. We will thoroughly look through Visualization.java, ReliabilityVisualization.java, and ReliabilityAnalysis.java to create Junit tests for the methods within each class. Once the Junit tests are complete, we will update and change the code as needed running the Junit test as we go. We will make sure after each change that the tests still pass before moving on to the next thing to change or update.