Task Documentation

Reliability Analysis and Visualization for WARP

Objective: Analyze and visualize the end-to-end reliability of message transmissions within the WARP system

Maintainer: Maria Gauna

Developers: Nancy Nahra, Tommy Looi, William Lucas

Project TimeLine and Sprint Breakdown

- **Sprint 1: Due November 12**
  - **Deliverables**: High level plans, README.md Updates, UML Sequence diagram, preliminary design and project plans.
- **Sprint 2: Due November 22**
  - **Deliverables:** Updated UML diagrams, initial ReliabilityVizualization class, ReliabilityAnalysis JUnit tests and JavaDoc comments, README.md.
- **Sprint 3: Due December 13**
  - **Deliverables:** Completed ReliabilityAnalysis and ReliabilityAnalysis classes, Junit test, final UML diagrams updated, README.md

Key Components of Each Sprint

- **Sprint 1: Due November 12**
  - **Deliverables**:
    - README.md: Documenting task assignments to each partner and project status

- UML Class and Sequence Diagrams: Show program flow of 'ra' option in WARP
- Design Documents: Design first preliminary project plans, artifacts that may be needed, UML diagrams, tasks, and project timeline.

♦ Sprint 2: Due November 22

- **Deliverables:**
  - README.md: Updated with progress and team task assignments
  - Update UML Diagrams: Reflecting class and sequence with new methods.
  - ReliabilityVisualizaiton Class: Correct output/flow, higher-level helper methods, stepwise refinement to keep correct program flow
  - ReliabilityAnalysis Class: Initialize Class, create JUnit tests
  - Formatting: JavaDoc comments for new methods, follow Google style guide

▪ Sprint 3: Due December 13

- **Deliverables**:
  - README.md: Final Documentation of task completion and project status.
  - Final UML Diagrams: Completed class and sequence diagrams.
  - ReliabilityVisualizaiton Class: Fully functional with JUnit Test and JavaDoc comments
  - ReliabilityAnalysis Class: Fully functional with JUnit Test and JavaDoc comments

Project Plan and Outline

Things to Remember:

- In Visualization you will use the toDisplay() and toFile() methods, tests should be done simultaneously to finishing each method, documentation shall include the ReadMe and JavaDoc comments.

Tasks to do:

1. Edit reliability Analysis: constructor, find sink and find source methods, equation helper method

2. Edit Reliability Visualization: to make the output look the same: methods to potentially add; displayVisualization(), createHeader(), createFooter(), createColumnHeader(), createVisualizationData(), createTitle()
3. Create Junit tests:
    3a. ReliabiltyAnaylsis methods. Examples include, testing if you're getting the source and sink right, if the equation helper method is outputting the correct answer.
    3b. ReliabilityVisualization methods. Examples include testing if the visualization is being displayed right for a GUI, if the visualization data is correct, if the file visualization is being outputted right
4. Add Java Doc comments to all new methods and tests
5. Update UML diagrams
6. ReadMe for each Sprint

Order of Tasks:

Do 1 first and then 3a. Then do 2 and after 3b. As you do each of these you're gradually doing 4. Then at the end you would do 5. Step 6 should be done add the end of each sprint.

Assigned Tasks:

- **Nancy**
    - JUnit testing for all newly generated code in ReliabilityVisualization and ReliabilityAnalysis
- **Maria**
    - Will update and create code for ReliabilityVisualization, implementing similar process to Program Visualization
- **William**
    - Will update and create code for the ReliabilityAnalysis and check correct output
- **Tommy**
    - Maintain UML diagrams, README.md, task documentation throughout the project, error checking newly created .ra files with provided correct output, aiding in creating JUnit testing