CS 2820 Introduction to Software Development

Spring 2022
Steve Goddard

Project, March 29
(Total of 300 points)

Due: April 11, 22, and May 6 (Three Sprints)

Team Assignment (teams of 4).

300 points: Create code to analyze and visualize the execution and latency of the WARP program.

1) One of you will be the Maintainer and invite the other team members to be a developer on your project, with an expiration date of Friday, May 20. All team members will work with this repo.
2) The project requires you to work in a team of 4, using a modified Agile Scrum process to complete the Latency Analysis component of the WARP system using the tools we have learned this semester. There will be 3 sprints, each approximately 1.5-2 weeks in length.
    a. Sprint 1 due: April 11
    b. Sprint 2 due: April 22
    c. Sprint 3 due: May 6

    Unlike a normal Scrum process, you will have defined deliverables that will be graded. You may deliver more than the minimum required, for which credit will be given in the next Sprint. Delivering less than the minimum results in a loss of points for that Sprint, AND you will need to make up the progress in the next Sprint. For each Sprint, you will evaluate your partners and provide the repo URL in the online Project assignment. Only one URL per team will be required.
3) Each Sprint will require a separate git branch and contain the following artifacts:
    a. UML diagram(s) of new or updated classes (if any) in that Spring.
    b. JUnit tests for all LatencyAnalysis and LatencyVisualization methods delivered in that Sprint, except the constructors (think about how to accomplish this), and JUnit tests for any new methods you might create in other classes, if any are created. To accomplish this requirement, make all new methods public for testing purposes, but comment whether it should be public or private in the JavaDoc comments.
    c. JavaDoc updates for all code, and JavaDoc files for the entire package of code.
    d. Source code (compliant with the style guide)
    e. Updated README.md to document each team member's completed tasks and tasks left to be done (and by whom).
4) Be sure to merge your Sprint {1,2,3} branch with the main branch so that you continue to build on prior 'releases', but maintain the branches so that the TA can evaluate progress.
5) **PROGRAM SPECIFICATION**: complete the `LatencyAnalysis` and `LatencyVisualization` classes such that they create the `*.la` files available from ICON and evaluate the end-to-end latency of WARP flows, as requested in the `Warp` main program when the runtime configuration option `-la` is used. Be sure to complete all methods declared in the class. Your output will look like the following for an Example1X graph with periods of 5 and 10 for flows F0 and F1 respectively:

```
Latency Analysis for graph Example1X created with the following parameters:
Scheduler Name: Priority
numFaults:      1
Flow/Time Slot  0    1    2    3    4    5    6    7    8    9
F0              Rx   x    xL   -    -    DRx  x    xL   -    -
F1              R    -    -    x    x    -    -    -    xL   -
```

After the header information, the top row of the visualization is a row of time slots. Each subsequent row consists of a flow name, and a visualization of its execution, wherein each time slot entry has one or more of the following entries:

`R` if an instance of the flow was released at that time slot;

`D` if that time slot is an absolute deadline for a released instance of the flow.
`x` if a push or pull for the flow was scheduled at that time slot (push/pull instruction in that time slot);
`L` if the last push/pull instruction of the released flow instance is executed in that slot.
`–` if the flow was idle in that slot (i.e., there was no push, pull, release, or deadline at that time slot.

If a flow misses its deadline, there will be no L for that release. The deadline miss is already reported in the Latency Report with the message `UNKNOWN latency for FlowName:Instance; Not enough transmissions attempted`. (This was already in the code that built the Latency Report that you refactored in HW5.) Notice, as in time slot 5 above, that a flow can have an entry with a `D`, `R`, and `x`. This entry indicates that time slot 5 was an absolute deadline for the first instance of F0, the release time for the second instance of F0, and a push/pull instruction was in time slot 5 in the program schedule table.

Your TA will evaluate your assignment by pulling your files from your Sprint branch repository. The assignment will be scored at follows:

Sprint 1

Artifacts: 50 pts.

1. (10 pts) High level plans and status in the README.md file. Clearly document who will do what tasks. Identify and explain any artifacts (diagrams and design documents) that should be considered in this Sprint delivery.
2. (25 pts) UML Sequence Diagram showing program flow starting with Warp processing the '`la`' option. Consider using https://sequencediagram.org to create your diagram, but you can use PowerPoint, or any other tool if you want.
3. (15 pts) Design and project plan documents (e.g., UML class diagram, pdf file, pptx file, etc.) that capture what you are planning to do. For example, what are the tasks to be done? In what order will tasks be done? Who will do them? How will you test? How will you document? You should also have identified some of the methods you will need in the Visualization class at least.

Sprint 2

Artifacts: 50 pts.

Completed or nearly completed Visualization class, including documentation.

1. (5 pts) High level plans and status in the README.md file. Clearly document who will do what tasks. Identify and explain any artifacts (diagrams and design documents) that should be considered in this Sprint delivery.
2. (5 pts) Updates to the UML diagrams (Class and Sequence) with the new methods, even if helper methods are stubbed out.
3. (10 pts) Code in the `LatencyVisualization` class file that follows the style guide, shows good design, and flows correctly (doesn't crash and creates some sort of output that indicates progress). If the class is not fully implemented, high-level helper methods documented with JavaDoc and comments explaining what will be done. Use step-wise refinement, with stubbed out helper methods so that the program flow exists, if the method is not yet finished.
4. (5 pts) JavaDoc comments and updated documentation files.
5. (20 pts) JUnit tests for the `LatencyVisualization` class.
6. (5 pts) Plan for Sprint 3, including tasks assigned to each person. Feel free to have part of this plan already completed. It is OK to finish early!

Sprint 3

Artifacts and Correctness: 200 pts

Completed, working project

1. (10 pts) High level plans and status in the README.md file. Clearly document who will do what tasks. Identify and explain any artifacts (diagrams and design documents) that should be considered in this Sprint delivery.
2. (25 pts) Updates to the UML diagrams (Class and Sequence).
3. (80 pts) Design and code correctness. All helper methods should be public, so they can be tested, but comments indicating which should normally be considered private or protected.
4. (25 pts) JavaDoc comments and updated documentation files.
5. (60 pts) JUnit tests for the `LatencyVisualization` and `VisualizationAnalysis` classes.