Software Process Reflection

How Waterfall is Used

While completing deliverable 2, the team followed a waterfall software process. For each of the bugs selected, there were 5 phases that were to be done in order, Requirements Definition, System and Software Design, Implementation and Unit Testing, Integration and System Testing, and Operation and Maintenance. Each of the later phases required the previous phase to be completed for that phase to be started. This organization of the phases was done using the project board on GitHub, each bug had an issue made and a corresponding note added to the board. The note would represent which phase the bug was currently in, starting in the first phase's column and moving to the next column once the phase was complete. Since the Waterfall approach has a focus on what are the results after each of the phases, team members have a written report detailing the work done in the phases and any important information before starting the next phase. This is also bundled together with any relevant files such as code to demonstrate the work done in that phase. The bugs that weren't chosen were rejected midway through the second phase, once we knew enough to make work estimates.

Efficiency of Waterfall

Using the Waterfall software process has helped the team be more organized and efficient in their development. As expected, when we chose waterfall, the requirements didn't change during the process. This means to check on a member's progress was as simple as to check their prior phase results and ask if they're on track for the next phase's result deadline. This also streamlined the written process; each member knew what to do and by when. Potentially saving us from a lot of confusion if each member was going through a unique process for their bugs.

It also made the process of members helping each other easier. If a member didn't understand something in the code, then others could help them out and resources online (such as documentation and previous issues) would be accurate because they wouldn't have started coding until phase 3, after understanding their requirements, the system, and how to implement the fix.

And Lastly, it made the process of working together easier. Once the 3 bugs to fix were chosen, 2 members needed to catch up on what had been done previously to assist in fixing the bug. This process was simple because all and only the planning had been done by this point (end of phase 2). There was ample documentation about requirements, what the current situation is, whether other parts of the system will be

affected and so for them to easily catch up. And no coding had been done yet so they didn’t need to be filled in on anything.