Choosing a Software Development Process

Agile vs Plan-Driven

The first decision to make when deciding what software process to use is whether we wanted an agile software development process or a plan-driven software development process. For deliverable 1 we decided to use a plan-driven software process for the following reasons:

- Agile is mainly for accommodate changing customer requirements. For deliverable 1 the requirements to fix the bug will not be changing.

- The easiest part of bug-fixing is writing the code. The difficult part is understanding what the bug is and where it is in the code. Plan-driven does all the planning at the beginning so seems best suited to understanding and finding bugs.

Incremental

Incremental development has 3 main pros but none of them are beneficial when working on bugs. The first is that it is easier to accommodate changing requirements, but since we are fixing bugs, our requirements will not be changing. Next is that it is easier to get customer feedback during the process, but during bug fixing, there is no need for customer feedback until the bug is fixed. And finally, incremental allows for rapid delivery of useful software. This is also not necessary since working code is already deployed and there are no intermediate stages for the software during bug fixing. Incremental also has a flaw that the process is not visible making assisting team members more difficult.

Reuse-oriented

Reuse-oriented is best used when there are existing components you can integrate into your system. But for us, the system is already made. Reuse-oriented would not be helpful for bug fixing.

Waterfall (The process we will be using in our next deliverable)

The software development process we decided to follow for the next deliverable is waterfall. Waterfall is a rigid 5 phase process, where each process must be completed before the next can be started. The main issue with waterfall is that the rigid nature makes it difficult to accommodate changing requirements, but, as mentioned earlier, this is not a problem during bug fixing. The main benefit of waterfall is that each phase produces a result. This is beneficial to us for 2 reasons. The first is that we can delegate bugs amongst the group members and easily see results so we can know everyone is on the right track. The second is that it makes it easier for group members to ask for help because they will have the results from their last phase to get each other quickly caught up with the problem.

Our steps for fixing bugs using the waterfall process

The waterfall process is broken down into 5 phases. These phases are requirement definitions, system and software design, implementation and unit testing, integration and system testing, and operation and maintenance.

Requirement Definition phase

For the first stage of the process, the team will brainstorm the requirements needed to fix the bugs. At the end of the phase, we will have a report on the details of what the bug is, proper steps to recreate it, and what the proper functionality should be once the bug is fixed.

Software Design phase

In the next stage of the process, the team will work to understand the bug and its impact. By the end of the phase, we will have a report on where the bug is in the code/uml, what causes the bug, and what would be impacted by changing the bug.

Implementation and unit testing phase

For this phase, the team will fix the bugs. After the phase, we will have working unit tests for the function/class where the bug is locating. These tests should show that the bug has been fixed and that it’s surrounding functionality was unimpacted.

Integration and System testing

In this phase, the team will confirm that the rest of the system is undisrupted. After the phase, the forked project on GitHub should still pass all tests and we will have also specifically tested all classes impacted by the code (found in the Software Design phase).

Operation and Maintenance

In this phase, the team will attempt to close the issue on GitHub. To complete the phase, we will submit a pull request as they have outlined in the following links:

https://matplotlib.org/devdocs/devel/contributing.html

https://matplotlib.org/devdocs/devel/gitwash/development_workflow.html#development-workflow