Top Right Corner

Our formal approach to change management

We planned all major changes together as a group, we evaluated the system that we inherited and checked it against our requirements one by one. This allowed us to determine any missing or incorrectly implemented features and any of our requirements that may need modifying. Once identified we gave some thought to the amount of effort needed to implement a given change and how important we felt that the change was[1]. The estimates for effort were based on the size of the change. Each change was then ranked primarily according to its priority and then by the amount of effort we thought it would require. Finally as many of the most high priority changes that we thought we could make in the next sprint were then allocated to team members. On a few occasions we realised that other changes were needed outside of one of our meetings in these cases the issues were raised and discussed in our group chat. In many ways due to our Scrum-like approach our change management process was very similar to our development process outlined in assessment 1: discuss, plan for the next sprint and then implement (this is largely because Agile processes were designed around the idea that changes are going to happen, it's a natural part of software development[2]). The only real difference was that we recorded the changes made.

We also evaluated all of the tests and documentation we inherited and after deciding whether to keep our old documentation or use what we'd inherited planned out the changes to said documentation and tests in the same way.

Due to the simplicity of the project, the small size of our team and small number of changes that needed to be made we chose not to use any change management tools as the overhead of learning to use such tools would have eaten into the time that we could have spent implementing the changes. We also didn't feel the need to use tools that allow team members to raise change requests to be reviewed by others at a later date (as these tools allow[1]), partly because we only had a short amount of time in which to plan and implement and time spent waiting for a response was not time well spent but also because the vast majority of our change requests were raised and discussed during meetings.

All changes were formally recorded (as recommended by Sommerville[1]) in their own document in a folder on our google drive. We did not choose to use the IEEE 828-2012 standards for configuration management because we wanted to avoid the large amount documentation overhead (this is a common choice for teams following an Agile methodology[1]). Instead we simply recorded the changes that had been made, who had made them and why. This fits in with our Scrum-like approach as we don't feel the need to spend time producing documentation that's never going to be used, we had a short amount of time to make our changes and had other commitments (including a summative ARIN exam) so we felt it best to focus on producing "real work". This is in fact one of the reasons that we chose to adopt an Agile methodology in the first place. Also, all changes made to code were marked with comments to make it clear that changes had been made.

Changes made to the GUI report

We chose to modify team Fractal's GUI report rather than our own report from assessment 2 as our report related to a totally different GUI. One of the main changes we made was to replace all references to their requirements with references to our requirements for traceability purposes (requirements can be found here: https://sepr-topright.github.io/SEPR/documentation/assessment3/RequirementsNoColouring.pdf).

We also cut out the first paragraph that related to the libraries used to implement the GUI as it held no relevance to the report as well as any text that contained superfluous opinions or just stated what was present without any explanation or justification. Originally there was little consideration given to playability and usability so we made comments about how various aspects of the UI supported these.

We also described the popup windows and game over screen that we added as well as the market screen that we heavily modified.

Finally we added links to pictures of the GUI elements described so that the reader can clearly see what is being described and fixed the spelling and grammatical errors that were present.

A version of the GUI report where changes are highlighted in yellow, additions in green and deletions in red can be found here: https://sepr-topright.github.io/SEPR/documentation/assessment3/GuiReportChanges.pdf

The updated versions without those highlights (and deleted sections removed) can be found here: https://sepr-topright.github.io/SEPR/documentation/assessment3/GuiReportNoColouring.pdf

Updates made to the testing report

We chose to modify our testing report from assessment 2 rather than take on Team Fractal's as we didn't like the format and it didn't match our testing methods very well.

We did not automate our UI testing as we did in assessment 2. We ultimately decided that it was not ideal due to the fact that LibGDX does not appear to be designed support automated testing in this way and that caused complications in many cases when we tried to instantiate LibGDX classes without an application window being open. We also didn't like the fact that we had to modify classes, adding methods to return buttons and labels that would only ever be used during testing or the fact that sometimes our tests would involve blindly searching through labels looking for a given string (in these cases there was no real way of knowing what it meant if a test failed or that it passed for the correct reasons). Instead we decided to automate the testing of all backend components and the user interface testing would be done as part of our overall system testing. Following this all references to automated GUI testing were removed from the report and a comment about system testing also being necessary to test the UI was added.

We also couldn't perform the exact process that we had wanted to: writing unit tests, then integration tests and then finally system tests as most of Team Fractal's tests were integration tests forcing us to go backwards and write unit tests after these integration tests had been written. So a comment about this was added. Although it is worth noting that ultimately we did end up with a suite of automated unit and integration tests. We also had to write a lot of tests as Team Fractal's automated testing was lacking in a lot of areas.

We added JavaDoc comments to every automated test which can be found here: https://sepr-topright.github.io/SEPR/documentation/assessment3/testJavaDocs/doc/index.html

The updated version of the testing report with changes highlighted in yellow, additions in green and deletions in red can be found here: https://sepr-topright.github.io/SEPR/documentation/assessment3/updatedTestingreport.pdf

An updated version without any highlighting and deleted sections removed can be found here: https://sepr-topright.github.io/SEPR/documentation/assessment3/updatedTestingreportnocolouring.pdf

Updates made to our methods and plans

No updates were made. The latest version is our report from assessment 2 and can be found here:
https://sepr-topright.github.io/SEPR/documentation/assessment2/updatedMethod.pdf

Updates made to our risk assessment and mitigations document

We all reviewed the risk register and considered each risk in turn to decide whether it was still necessary and if the estimates for severity and likelihood were appropriate. However, no updates were made as it has been only 4 weeks since the last changes were made and in that time nothing has occurred that makes us feel that we should change the likelihood or severity estimates associated with our risks. Similarly we haven't had any reasons to remove any risks from the risk register or add any new ones.

The latest version of the risk register is our one from assessment 2 and can be found here: https://sepr-topright.github.io/SEPR/documentation/assessment2/updatedriskassessment.pdf

References

[1] I. Sommerville, Software engineering, 10th ed. Harlow: Pearson, 2016, pp. 731, 745-750.

[2] A. Dr. Myles Bogner & David Elfanbaum, "An Agile Approach to Change Management", CIO, 2017. [Online]. Available: http://www.cio.com/article/2410953/change-management/an-agile-approach-to-change-management.html. [Accessed: 20- Feb- 2017].