Module	SEPR
Year	2019/20
Assessment	3
Team	Dalai Java
Members	Jack Kershaw, Max Lloyd, James Hau, Yuqing Gong, William Marr, Peter Clark.
Deliverable	Change Management

Formal Approach to Change Management

In order to manage changes effectively, we took an Agile approach to change management through all of our deliverables, basing our approach on the Twelve Principles behind the Agile Manifesto [1]. Our group initially started development following the Scrum methodology, in which our equivalent 'Scrum Master' was the project manager. Their job was a general overseer of the project, making sure all deadlines are being adhered to. Our sprints were more focussed on increasing the quality of the program and adding requirements due to the tight time constraints. As we are following an Agile style of development while taking over another team's project we were required to reconsider our approaches to development. We called a group meeting and under the guidance of the project manager, we replanned our 'sprints' that would lead us up to the deadline. The most essential part of this meeting was to prioritise essential requirements and place them at the forefront of our development cycles, whereas less essential requirements were assigned a lower priority. Due to the nature of Agile development, we were able to adapt to the changing of requirements due to the short nature of our sprints. This iteration meant that we were constantly updating our plans with each sprint as opposed to writing all documentation at the beginning, meaning that less time was spent on initially 'perfecting' documents.

As a team we decided to take on NP Studios' project. The first step in formally taking over their project was to clone their Github repository and create a new repository containing the cloned code. Once we had a version of their code the most important task was to have a group meeting where we looked through the code and their architecture report to gain an understanding of how the project worked. We then constructed a list of tasks that needed to be achieved in order to meet all the deadlines we had.

After looking through the project code we realised due to the nature of their level design, we could not develop future levels in the same art style as the original levels. Thus, our next step was to retrieve all of the assets used in the original game in order to create the remaining levels without breaking consistency throughout the game. We were able to import their maps into TiledMap editor where we used the same structure to provide the same cohesive level design.

In order to provide accurate updates to their previous documentation based upon our changes, we sent a request to NP Studios for editable copies of their documents from the previous assessments. In regards to the testing report, we decided that we would leave the author of the original tests the same as they were not part of our group, however all tests implemented after we officially took on their project have the appropriate author attached. When changing their documentation to be adapted to our group's style of development, any additional or updated requirements were assigned an appropriate owner.

The last step in formally taking on their project was to create a clone of their website. We initially attempted to find the CSS file for their website but found out it was generated through a program called Jekyll. Jekyll generates a static website based upon a template which can be edited through various markdown files. Therefore attempting to clone their website would not be a solution. The solution we had in the end was to create a fork of the existing NP Studios website repository, then run Jekyll again on the fork to produce a copy of their website to which we can update with the new deliverables and changelog.

Changes to Testing Report

Barring a few changes to grammar, we did not have to make many major updates to the 'Testing Methods and Approaches' section of the report. This was because we believe that the test strategy devised by NP Studios has been very well thought-out. For example, testing code as it is written is a good approach as it means that neither the testing nor the development team has to wait for the other to complete before commencing. As we used JUnit and Mockito in our testing for the previous assessment, we were able to simply extend the tests created by NP Studios without changing platforms. They have also assessed the risks of JUnit and Mockito comprehensively and described mitigation for such risks, ideas which we have decided to take on as we continue the testing of the product.

We did, however, make changes to the section describing the team's roles in the testing - whilst this was mentioned briefly in NP Studios' report, no specifics were included. We agreed with the idea of having a dedicated testing team as this allows other members of the team to be purely focussed on the implementation, and hence we only assigned roles to three members of the team, leaving the other three to focus on the implementation. We also added a concrete deadline for the tests to be completed to the schedule in order to ensure that it was clear that tests were of importance.

In terms of the tests themselves, we then went through every requirement and made significant edits to the 'related requirements' section and added manual tests for each requirement that had not yet been tested, as a large amount of the code had been left untested due to incorrect tracing between requirements and tests. Once we had rectified this it was clear that not every test had a relevant requirement, meaning that a number of new requirements needed to be added (these are documented in Impl3). We updated the Traceability Matrix accordingly. There was also a small number of Unit tests which had not been implemented, despite the documentation marking them as having been passed (examples include test_2.1 and test_3.1.1); we completed the implementation of these tests to ensure that they were passed.

In terms of the test documentation, we made the decision to edit the descriptions of the majority of the tests. This is because some of the test descriptions provided were highly ambiguous (for example 'Test to ensure default constructor works as intended via getters'), whilst others still made it difficult to decipher which class was being tested. In light of this, as well as improving the clarity of test descriptions, we also created another document named 'Unit Test Traceability' which clearly shows how each test in the test documentation links to the Unit tests in the Java program. We also added a number of new unit tests, ensuring that the Pipe class for the minigame had been tested comprehensively, and adding extra tests in existing classes to ensure maximum possible line coverage. We also removed two tests from the FortressTest class as we came to the conclusion that it contained three tests testing the same 'getter' but with different values, something we deemed unnecessary.

We had to make radical changes to the report on the tests, mainly because the team had used this section to describe their testing procedure at length as opposed to providing test statistics and commenting on the completeness and correctness of code. We added a table outlining the results of Unit testing on each class, describing how we believed the tests were complete in further depth. We also wanted to make it clear that regression testing was an important part of our testing strategy for Assessment 3 as preserving the existing behaviour when refactoring code is incredibly important. To show this, we added an extra column into our Test Documentation in which we tested each existing test again, and outlined our reasoning behind this in the Testing Report.

Changes to Methods and Plans

As our Methods approach was very similar to NPStudios, not much needed to be changed. The methodology that they chose was Agile. Scrum, the methodology that our team initially chose, is very similar. The key difference being that Scrum has less focus on a specific team leader; there is a 'Scrum Master' who essentially fills that role but their position is more of a coaching one than an actual leader. Our project manager is our group's equivalent of the 'Scrum Master' (the role taken by Jack Kershaw). Furthermore the 'sprints' focus more on specific features that will add to the overall quality of the product rather than sequential phases. All code must be tested as it is built to ensure that it will merge well with the rest of the project rather than having a specific testing phase.

They also opted to use GitHub, Google Drive, and Trello. They did however start using GitKraken Glo Boards for the second Assessment which appears to be similar. We are currently comfortable using Trello and therefore have continued to use this as it serves the same purpose as GitKraken and provides all the necessary features for effective group work.

NPStudios opted to create a backlog of tasks to complete for each sprint, the organisation of which was on Trello. This is similar to our approach of having tasks 'to do', tasks 'in progress' and 'completed' tasks. As this is nearly identical to what we wrote in our report we can leave this in also.

NPStudios also chose IntelliJ as their preferred IDE. Our group have already been using this IDE, (with the exception of the website) however it is something that we omitted from our report . IntelliJ offers easy integration with libGDX and is relatively easy to use, NPStudios highlighted that "we found it was easier for us all to use the same one so that we could all help each other out if we ran into issues". This is a very good point and has led to our team resolving problems quicker as we are all familiar with the IDE. For these reasons we have continued with IntelliJ as our IDE. (we have not however been to find the updated deliverable for the methods document for Assessment 2 and so have added a paragraph of our own).

The game engine which they chose was libGDX; this was the same engine we had developed our initial game with and as such was a major consideration in choosing a project to take over. libGDX has the advantage of being relatively simple to learn whilst still having plenty of features allowing us to make a well refined product. Further to this, it allows for easy porting to Android which abides by the requirement of the game being easy to port to mobile.

The approach that NPStudios took to team management was also very similar so only minor alterations were needed. Many of our techniques were the same, such as allocating sprint tasks on Trello and allocating tasks to groups rather than a single person. The only notable exception being that they didn't give specific roles to team members for tasks other than writing. Choosing roles for individual team members was something we overlooked until the previous assessment. We found that people had fallen into certain roles anyway and thought that making these official was necessary so as to give people clearer responsibilities and goals.

The 'Plans for future assessments' document did not appear to have been changed from Assessment 1 (based on the copy we have received) and as such needed to be changed completely. Plans have been written up for Assessment 4, as this is the only assessment that we can plan for at this stage and all of this section has been written by us, including the Gantt chart which contains clearly defined deadlines for each portion of the final assessment.

Links

Updated Testing Report: https://dalaijava.github.io/files/Updated%20Testing%20Report.pdf
Updated Plans: https://dalaijava.github.io/files/Updated%20Methods.pdf

Bibliography;

[1] "Principles behind the Agile Manifesto", *Agilemanifesto.org*, 2001. [Online]. Available: https://agilemanifesto.org/principles.html. [Accessed: 14- Feb- 2020].