| | |
|---|---|
| **Module** | SEPR |
| **Year** | 2019/20 |
| **Assessment** | 3 - Change Report |
| **Team** | Mozzarella Bytes |
| **Members** | Kathryn Dale, Emilien Bevierre, Ravinder Dosanjh, Elizabeth Hodges, Daniel Benison, Callum Marsden |

## Approaches to change management

***Deliverables:*** *Website:* The previous group used React to build, and Firebase to Host the website. Firebase is linked to your Google account, therefore we had to: set up our own Firebase project, clone their website repository, ask for their previous deliverables to upload, update the content and then deploy our version of the website. We also had to learn the basics of ReactJS to edit the site. This was a lengthy process, therefore we don't think this setup was well suited for a project that can change ownership, instead we would suggest Github Pages with a simple CSS framework such as Bootstrap.

***Code:*** Our change management strategy is based on strategies from the Information Technology Infrastructure Library (ITIL) guidelines to change management. Firstly if a team member wants to propose a change they submit a request for change (1) by amending the change log (2) which all team members can access via a shared Google Drive.

In keeping in line with ITIL (3) during the next team meeting we review any new RFC and decide whether to implement them. The decision is based on whether the change is feasible, how long it would take to implement and the impact on the rest of the software (3). If we decide to implement a change the person who submitted the RFC ensures the change is visible (4) to the rest of the group by adding it to GitHub projects (this makes sure all team members are working towards the same goal).

The development team then discuss how to implement the changes, delegate tasks and implement unit testing. If the change is implemented, then the status of the change is changed to complete on GitHub projects so all team members know the change has been implemented. If the change could not be implemented a review would take place during the next team meeting where a conclusion would be made on whether to continue trying to implement that change.

*Testing:* The first step is to identify what to change. After analysing the previous groups tests, test results and line coverage we concluded that their tests were incomplete and did not accurately test the code's logic. We decided it would take more time to get their tests to work then to implement our own tests.

Our strategy is that when a change is added to GitHub projects the testing team (Emilien & Daniel) would talk to the development team about what functions the latter were thinking of adding to be able to implement the change. Once the testing team has generated their unit tests they add a message to GitHub projects so the development team knows they can test their code. If the code does not pass the test as expected both teams meet to discern whether the issue lies with the test or code and work together to resolve the issue.

***Documentation:*** To highlight if a new function or class had been added for this assessment a banner stating "new function for assessment 3" has been added above the added function. If a new class has been added for this assessment then a banner stating "new class for assessment 3" has been added to the top of the class, this implies that all functions in that class have also been added for assessment 3. If a function has been slightly modified for this assessment then there is a banner stating "modified for assessment 3" above the modified function

*User manual:* Due to adding new features and editing a lot of the graphics and controls we decided it was more time efficient to create a new user manual then to update the previous team's manual. After identifying a new manual was needed our change management strategy was to highlight features and controls from our implementation report that we thought the user should be aware of and add them to the user manual. To improve the clarity and helpfulness of the manual we edited its content based on feedback from a couple of people who played the game for the first time.

## Changes to Testing

We regrettably chose this project without looking at the testing, which was close to non-existent. To facilitate the testing process in the future, we will follow a test-driven development process in order to write code that is easily testable, and as to not only save us time writing the tests but also will save us precious debugging hours.

As of now, we ensured to complement our Unit tests with End-to-end testing to further verify our code worked as expected.

The previous group mainly focused on "Smoke/end-to-end" testing rather than unit testing. Through discussions with our client, we believe that unit testing is a really important measure to properly test a system. The testing report highlighted their difficulty in getting JUnit tests to work with LibGDX, however we managed to get them working, therefore we set out to create JUnit tests to cover the core game features (Firetrucks, Patrols, Pathfinding, Fortresses and Firestation).

We made some previous end-to-end tests that were marked as "failed" pass by re-interpreting the meaning of the requirement. For example, OPERABILITY requirement states that "considerations should be made for mobile versions [of the game] in the future". The previous group marked this test as "failed" because their expected result was to actually port the game to mobile devices. We based our test on the word "considerations" as we made sure that all our core mechanics are easily portable to mobile devices (e.g. every interaction in our game can be done on a tactile screen).

The way the previous group documented their tests between the Traceability Matrix and Test Cases was a clear way of linking requirements to tests, and making sure they tested each requirement, therefore we extended their approach on this when implementing our own tests. We added our additional requirements (discussed in Implementation Report) to the Traceability Matrix and then documented tests on these additional requirements which can be seen in the Test Cases. Any tests that were affected by these requirement changes were edited or replaced with tests that were more suitable to the updated requirements. Examples of these are removing tests for *LEADER_BOARD* and editing tests for *MENU* which can be seen updated in tests *T.U.013 and T.U.018.*

We updated the previous group's Test Cases, Traceability Matrix and produced two new documents (Code Coverage, Testing Results) that further support our tests strength and coverage, which can all be found at https://sepr-docs.firebaseapp.com/testing.
We introduced Code Coverage as a result of our new Unit tests which allows us to justify the strength of our test suite. While the percentages can be used to identify the strengths or weaknesses of our Unit tests, a low percentage on certain classes does not imply weak testing, as some methods do not need to be tested (e.g. getters & setters). The document highlights methods that are tested and those that aren't, which can justify why a certain class' coverage percentage is low. (e.g. render methods, draw-calls and methods inherited from parent classes provided by LibGDX in general).

Writing new unit tests allowed us to design our test suite using boundary testing for numerical variables that were used in a specific range (e.g. Patrols' attack range). Testing extreme bounds allows us to virtually test all values a variable might take at runtime which cannot feasibly be done using end-to-end testing.

## Method and Planning

*Team Management:* We did little to change the working method of the previous team, as we found that they used a very similar Agile approach to ours in the last assessment and we had found it to be very effective in co-ordinating different aspects of the project and keeping team members on track. We continued to use their Belbin Model [1] for team roles and found this to be a really effective way of being able to manage our work and keep on top of it. It gave us greater flexibility in our roles and what we felt was and wasn't within the scope of our role than our original SCRUM roles did.

*Tools:* Due to the flexibility of the tools that we used for the development of our Assessment 2 project, we were able to continue using them to extend the new game. We only inherited one new tool from the previous team, making it much easier for us to work on the project, as we were already familiar with all the tools we were using.

*New Tools:*
Mockito: We used Mockito because it allows mock dependencies to be created which can be used in the testing of individual units. This helped to ensure that we weren't indirectly testing multiple methods.

GitKraken: We used GitKraken over a Git Bash terminal due to its straightforward UI and simple methods to merge conflicts, push and pull commits.

Intellij: This is an intuitive coding environment with built in features to perform tasks like checking the line coverage of tests. Its simple to understand colour coding system of key words helped to make keywords visible and distinguish features.

*Continued Tools:*
Firebase: This is a website development platform that the previous team had used to create and maintain their website. We continued to use this platform for continuity, however found it quite inefficient.

Facebook Messenger: This continued to be an excellent platform for the team to communicate and discuss concerns and ideas, as well as plan physical meetings. We also continued to give regular virtual updates via a poll to allow all team members to know each other's progress.
JUnit: We continued to use JUnit to develop tests for our system so that we could ensure client requirements were being met. JUnit is an open source unit testing framework for Java. As it is a popular framework, there exists a vast range of supporting resources, allowing us to make our tests as comprehensible as possible, thus improving the quality of our code by allowing us to find and fix more issues.

**References**


| | |
|---|---|
| [1] | F.C.Martins, *Implementing ITIL Change Management,* unknown, [Online], Available at: https://pdfs.semanticscholar.org/c11b/896ebda9b7dc1577c09c622c83ee1f3f7724.pdf Accessed on: 27.01.20 |
| [2] | Mozzarella Bytes, *Change management log, 14.02.20, [Online], Available at: https://sepr-docs.firebaseapp.com/changes* |
| [3] | M.Blumberg, A.Cater-Steel, J.Soar , *An organizational Change Approach to Implementing IT Service Management,* Australasian Conference On Information Systems, [Online], Available at: https://pdfs.semanticscholar.org/5cf6/39b7931897e281ee13fe933c247bee8d3dbc.pdf Accessed on: 27.01.20 |
| [4] | EDUCAUSE, IT Change Management – A practical approach for Higher Education, [Online],Available at: https://uit.stanford.edu/sites/default/files/2018/09/19/Educause%20-%20IT%20Change%20Management%20-%20A%20Practical%20Approach%20for%20Higher%20Education.pdf Accessed on: 27.01.20 |
| [5] | W.Cole, *Roles and responsibilities in a software development team ,* 09.08.18, [Online], Available at: https://this.isfluent.com/blog/2018/roles-and-responsibilities-in-a-software-development-team Accessed on: 25.01.20 |