Change Report

Tom Rosewell
Sam Redgewell
Tom Keeler
Jake Stogden
John Limb
Adham Noureldin

Formal Approach to Change Management

Our first step to managing change is to read and analyse the deliverables from assessment 1, making sure we understand how each part sequentially related to the next, i.e. how the requirements evolved into the architecture diagrams, and how that then related to the code. Doing this we can begin to weave the new requirements into the workflow the previous team used. First we will analyse and collate the new additions for assessment 2 to give us an initial list of changes that need to be made to the original documents. These changes will likely be of high importance and can be implemented first in our 11 week plan to create a suitable approach to change management. While we will plan for the changes we initially know, it is inevitable that we will discover many more changes we need to make due to conflicts between the way we intend to implement the new additions, and the way the previous team implemented theirs. This will also include many bugs and incomplete features we will encounter during our testing and implementation phases. To counteract this, extra time will be allocated along the way to make changes we are not yet currently aware of.

To ensure consistency and to make sure the full workflow for assessment 1 is still intact, we will keep archived versions of each deliverable before any changes are made. We will then compare these to the updated versions at the end of the assessment to ensure no relevant details or information has been lost. In addition, the java docs will be updated for new/changed code to maintain clarity.

Requirements (Link)
The original requirements document was laid out in 2 main parts:
- How the requirements were elicited
- What the requirements are

In order to keep with this basic format, we first appended the source of the new requirements needed to complete the new design specifications to the elicitation of requirements section. The major new requirements were the
- Introduction of combat with both colleges and AI ships.
- A way to save the game state and reload it at another time
- Different difficulty levels (e.g. easy, normal, hard)
- The implementation of 5 unique power-ups that can be obtained via traversing the map
- Bad weather and obstacles that can damage the player's ship
- A way to spend loot gained via completing tasks (e.g. using gold from defeating enemies to fix your ship).

We then converted these new requirements into the existing table, maintaining the same format used by the previous team to ensure consistency.
We also updated the table to fix inconsistencies and redundant requirements. This included merging a few of the existing requirements into the new ones, and removing unused ones.
The requirement elicitation for the initial requirements from our end user did not change so all of that information was kept, however requirement elicitation for this project was straight from the document and updates Via the VLE therefore the new documents did not have to have any information about where they came from and were merely added to the table of requirements

Abstract and concrete architecture (Link)

The first thing we needed to do was recreate the old team's architecture diagrams. This is because these were generated in the Intellij editor, which differs from the current IDE we use (Eclipse). As we do not have access to the original code system, we would have been unable to seamlessly create additions to the diagrams. Hence, the first thing we did was to remake these diagrams in PlantUML. This took a significant amount of time due to limitations in the Plant UML framework, as the size of the diagram ended up breaking the tool. Hence we needed to split the diagram into its constituent part in order for it to be legible. From there we added the changes needed to make the architecture compliant with the new requirements.
The abstract and concrete architecture have been changed dramatically to reflect the changes made to the project as well as our additions needed to fulfil the requirements. The explanations for assessment one also needed to be cut down in order to fully incorporate the additions without running over the page limits.

Methods and planning (Link)
The first section, on methods, was updated to reflect the new methods employed by our group for the development of the second phase of the project. However, the tools used and the reasons behind their use remained the same, as we tried to remain true to the original build so as to avoid bugs and complications down the line, thus this part of the section required no updates.
The next section, on internal organisation, needed to be updated with respect to the allocation of job roles within our group.
The section on the external organisation section was updated with how the team devised planning, and how we adjusted our plan.
Finally, the sections on the systematic plan and project evolution were updated to include outlines for both new plans designed by our group, with a brief description of how/why they were created.
The project evolution section was updated to give an overview of how changes were made to the project in this second stage.
A week-by-week breakdown was added under the ‘weekly snapshot’ section for assessment 2 on the project website. This goes into further depth about the scheduling of events by giving an insight into the groups’ efforts for any given week.
It also carried on from the previous groups week-by-week gantt chart, using the same program, allowing for streamlining in documentation and a more succinct look on the website (Link)

Risk Assessment and Mitigation (Link)
The first section on risk format and detail was not changed, as the previous team’s analysis on types of risks, and how they should be documented and dealt with was in agreement with, and was broadly in line with our team's desired approach.
The previous team did a very extensive analysis of possible risks, however we felt that a few risks we identified during our risk assessment that would be relevant to the project had been omitted, as well as a few risks not being extensive enough. Hence, we added 2 risks to the new table to ensure the risk assessment is to the level we deem appropriate for the project.

R4 - Code is illegible to other developers
We felt this was crucial to add as it had been a problem in our previous project, and as such needed mentioning so the team could rectify and work around it when building their code

R12 - The Game is too Difficult
We felt this classified as both a risk and a requirement as it represented a user requirement to meet the use case and if the game was too difficult it would limit our manual testing options for the game unless we built an entire test game type that artificially decreased the difficulty of the game, this would put more stress on the coding team and as such we made sure to be aware of the games difficulty as a risk and a requirement throughout the production of the code.
As the previous team used a medium to high rating system, whereas we used a 1-10 system, we converted the ratings to our system, assigning what we felt were appropriate values to the existing risks.
This also allowed us to be more specific about the severity of risks involved, the likelihood and to think more critically about the issues with their risk assessment.
Regarding this we altered the severity of:
R6 - Screen Size - Severity
We felt that this would not drastically affect gameplay as the size of the window was smaller than almost all modern hardware and due to the use case being inside the CS labs were almost all computers support 1920x1080 resolution, this was not a M severity problem and as such we set it as 2
R2 - Internal Deadlines - Severity
We felt that whilst the classification of M for this was not inaccurate we wanted to put it higher on the severity rating so we gave it a 7 this was due to occurrences in the previous project where aspects of the program were not done significantly halting the process of the creation of the game.
R1 - Teammember falls ill - Likelihood
We felt that due to the nature of the project a team member becoming too ill to work online/remotely on this project was very low as all members owned a way of interfacing with the project whilst at home, this allowed us to downgrade the likelihood of a member not being able to work to a 2
R6 - Screen size is larger than tested for - Likelihood
We felt that due to the hardware of some members of the team who played this game initially the chance of this occurring in the game after we had all run it individually was very low, especially given the use case of being in a CS lab where most screens are 1920x1080, this allowed us to downgrade the likelihood.
R7 - Overengineered with excess features - Likelihood
Due to the time constraints of many of our members we suspect that we will more likely not hit all of our targets and as such have a significantly lower chance of this becoming a problem for our team in this project.
R13 - Requirements are added mid way through production - Likelihood
We suspect this has a low chance of occurring as we have no indication from the client that their requirements may change, as well as very limited communication with our client resulting in new requirements being difficult to find, if we believed this would have been a more likely risk we would have invited the user for a second interview with a beta version of the game to circumvent surprise updates outside of the bounds of the planning we have made, however we agree the severity is medium as it can ruin our teams schedules for implementation.

We also Implemented specific risk IDs to help us reference them in any issues that could occur deeper into production.
Also, as we are a new team, we added ourselves as risk owners under the appropriate risks. This correlates to the team organisation we devised at the beginning of the project.