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.

<u>Requirements</u>

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 included 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. Much of how the requirements were elicited remained the same, thus this section requires little to no change from the origin documents.

<u>Abstract and concrete architecture</u>

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. From there we added the changes needed to make the architecture compliant with the new requirements.

The Abstract and concrete architecture will need to be changed dramatically. This will be done after our final implementation is made. This is because we will need updated UML diagrams and flow charts that will be based on the final product. If we were to design an architecture beforehand, we may need to redesign it again after our software testing if a major issue is discovered.

Methods and planning

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, 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.

Risk Assessment and Mitigation

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.
As the previous team used a medium to high rating system, whereas we used a 1-10 system, we converted our ratings to their system, to a rating we felt closely matched our original intention.
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.