Part 2. Change Report

Team 5 | Team Pending

Thom Robinson
Iris Scoffin
Ayman Elsayed
Annice Headman
Izaac Threlfall
Yihong Zhao

Part A - Approach to change management

Our team has decided to take over team 10’s project. This involved not only taking over their game but also their existing documents. In part B of this document we explain any changes made to these documents, and in Part 3B of this assessment we detail major code changes.

Generally, we used a shared Google Docs file as a way to keep documenting what each person has done and what is tasked from each member after every meeting, similar to a log book. This alongside our Gantt Charts helped us know what needs to be changed and when.

Approach to code changes -

In order to make changes to the game Team 10’s repository was forked to give us a version of the game that we could work on without impacting the original repository.

From here any major changes/features as described in Part 3B were created as Git Issues that were assigned to one or two team members. Git Issues were beneficial as they helped everyone stay on track and made it easy to understand the tasks that needed to be done and who's working on it. These new features were then developed on their own branches using sensible branch names to indicate the feature in development. Branch owners adopted the practice of pushing changes often so that participants could observe and help.

Upon completion of the feature the branch was merged back into main, however, when branches ran too far behind we utilised Git’s rebase feature to bring the old branch up to date, avoiding more merge conflicts when the branch was eventually merged.

Our team decided to keep using scrum methodology with weekly meetings instead of adapting to the previous team’s extreme programming methods. That was because this was the methodology we used for assessment one hence we thought it will be more efficient to use it again instead of trying to adapt a different method. As such, branches were used in our sprints.

Approach to Documentation changes -

Firstly, to acquire editable copies of the previous teams documents, we converted the original PDFs into editable Google docs file. From here, we used a shared Google docs file to record what changes need to be made to each relevant file and who was responsible for making that change. This kept a record of changes made that was useful for producing our change document below.

Generally the documents regarding topics such as requirements, methodology and risk management were updated at the beginning of the project because the contents of these documents were unlikely to change much and were essential for completing other documents. Whereas documents such as architecture were planned to be updated after the game was completed to provide the most complete description of the architecture.

Part B - Change Report

This part of the document describes the changes made to existing documentation. With each section starting on the top of a new page.

I. Changes to Requirements

Found at : https://eng1-team5.github.io/static/A2u/Req2u.pdf

When starting assessment 2, we as a team felt another meeting with the stakeholder was unnecessary. This was because after completing assessment 1 we had a good understanding of how to elicit requirements from the brief. The introduction to the document has been changed to reflect this, removing details about stakeholder meetings.

When updating the requirements, we began by removing any requirements we felt were outside the scope of the project such as NF_007, allowing for remapping of keys. We also removed any features we felt were unnecessary like F_013, Random college spawns. We then changed the names of all the requirement IDs to make them more meaningful and improve readability of the document. For example, NF_003 became NF_Difficulty. These changes allowed the “Name” column to be removed, making the whole table easier to read. Finally, we added the requirements for assessment 2 to complete the table.

II. Changes to Concrete and Abstract Architecture

Found here : https://eng1-team5.github.io/static/A2u/Arch2u.pdf.

Firstly, we thought it would not be necessary to make any changes to the abstract architecture as it still held up as the starting point for the project and made sense to keep as it. Any new features or classes seemed too small or inconsequential to add to the initial abstract UML diagram, especially as we were working on the already existing concrete architecture for part 2 rather than the abstract diagram.

<u>Change 1 - Rebuilding and Updating Abstract Data-Flow Diagram</u> - It was decided that the abstract data-flow diagram should be updated to reflect the updated state of the game. However, we decided to be selective with what information is in the data-flow, as adding new mechanics like buffs, weather and obstacles would have increased the diagram to be large in size and would no longer provide the useful overview of the game that the diagram previously provided. Thus, the only change made to this diagram was to add in the same collision / shooting loop that “College” had but applied to a new Enemy class on the diagram.

Furthermore, to apply this change the whole diagram had to be reproduced in draw.io as the previous team had not supplied us with the original draw.io file. However, other than the addition of “Enemy” on the diagram, the rest remained unchanged.

<u>Change 2 - Rebuilding the Concrete Architecture Diagram</u> - Upon contacting the previous team we found out they used a program called Visual Paradigm, which is a subscription based modelling tool starting at $6 a month that allowed them to automatically create a concrete UML diagram in full detail using the project’s file. Furthermore, the previous team could not provide us with an editable UML file as Visual Paradigm only produced an image file as its output. Therefore, unless we paid for Visual Paradigm, or used the trial, which covered any image with an intrusive watermark, we had no way to alter the concrete diagram.

To solve this problem, we decided it best to completely rebuild the entire UML document using a PlantUML document in Visual Studio Code. This decision meant we did not have to spend money on software, and created a UML diagram that can be edited any time in the future. Also, the original UML diagram was far too detailed and large to provide useful concise information for anyone working on the game due to its inclusion of every single attribute in detail, therefore, a more concise, well laid out diagram would be more valuable to future developers.

Some of the major changes are:

- Altering diagram to a more compact, readable diagram using a semi-hierarchical layout centering around the main game class.
- Enemy now implements the IHittable interface.
- Adding new classes for Buff, Collectable and its children Pickup and Upgrade as well as Upgrade’s unique interface IInteractable. Any relationships and multiplicities with the Game and Player have been added as well.
- Adding the Weather and Obstacles class
- Replacing the now removed AttainGoldObjective with the new

GetLevel5Objective class.

The rebuilt diagram can be seen on page 4 of the architecture document. The resulting diagram is more concise and easier to read due to its omission of unnecessary or internal class details. But it is also more valuable to potential stakeholders and developers as it includes new details such as a hierarchical layout, relationship multiplicities and distinctions between interfaces, classes and abstract classes.

Change 3 - Requirement name changes - The previous team had used a number system for their requirements (eg. F001), these had to be changed as described in the previous section of this document to a more descriptive version (eg. F_Saving). As a result of this change, all previous and future architecture justifications have to use the new requirement name system. Therefore, I began by altering the existing justifications to use the new requirement names, and removing references to the defunct AttainGoldObjective and KillShipsObjective classes, replacing them with GetLevel5Objective.

Change 4 - Writing new justification - In part two we added a total of 6 new classes and one new interface to our concrete architecture. To explain our reasoning, two new labelled paragraphs were added to the document, “Obstacles and Weather” describing how we implemented the Obstacles and Weather classes and how their collisions work and the effect they have on players.

The 4 classes and one interface used to implement both upgrades and buffs are described in their own section, explaining how they build upon each other in order to maximise code reuse and explaining why we created a new interface.

Finally, for both sections their appropriate requirements have been included, explaining how each are satisfied by the additions of these new classes and their functions.

Change 5 - Updating old justifications - Before we took over the game, the enemies implemented in the game had no collisions with the player, combat or targeted movement. Therefore, in part 2 of the assessment we added targeted movement when in range of the player Enemies added. As such, the existing section has been updated to reflect the current state of the enemy class, explaining its implementation of IHittable and how that supports damage, collision and rewards. Finally, this was linked back to the completion of the F_Ship_Attack and F_Ship_Death requirements.

Additionally, a short explanation was added about the Game class and how its storage of all game object referenced simplify processes such as collision, changing difficulties and the new JSON saving and loading.

Change 6 - Restructuring and Fitting to Size - Originally, the file already was a total of two pages long, with no extra space to add the additional justification. Therefore, we knew it would have to be cut down or refactored in order to fit within the page limit without losing any of the important justifications and links to requirements.

Additionally, we decided it best to add labels to each section of the architecture justification so that it was easier to locate sections and immediately understand what section of the diagram we were explaining.

Specifically, the following sections have been restructured and tweaked:

- The sections regarding particles and visual appeal were combined and shortened into a single paragraph, labelled “Particles and Visuals”.
- Restructured around general collisions and the IHittable interface by explaining collisions for the IHittable classes into a section labelled “Collisions and IHittable”.
- The section regarding the updateAI() function was removed as most of what it explained was redundant and was explained better in relevant classes’ sections.
- The sections regarding the Bindings class and information about singleplayer and lack of network access have been combined into a section called “Input”.

Additional sections were cut down and made more concise, but are too small to list here.

III. Changes to methods and plans

Found here : https://eng1-team5.github.io/static/A2u/Plan2u.pdf

Upon inheriting Team 10’s project we noticed that there were methods and tools that they had used that we had not. These were detailed in their methods and plans document and had to be changed to reflect the methods and tooling used in Assessment 2. All changes and their reasoning are briefly described below.

Methods:

- Had to change to delete the part regarding the team using spiral hybrid lifecycle as our team didn't really put much emphasis on the different life cycles and felt that it might hinder how we work so we then decided to focus more on the development method that we used
- Did not change the part talking about using agile methodology as we did use scrum which falls under the agile methodology
- Had to change the part talking about extreme programming as we ended up using scrum instead as it worked better for our team. We also explained how scrum worked for us in detail
- Did not change the part talking about pair programming as we also used it to develop some of the new features in the game.

Tools:

- Did not change the part regarding the use of github for collaboration as our team has used it as our main way of sharing code and working collaboratively mainly for the reasons the other team has mentioned.
- Did not change the part regarding using github pages for the website as that’s what our team used for our website
- Had to add a part talking about the use of github issues as it was a very important tool that helped us a lot and made team management much easier
- Did not change the part regarding the use of discord as it was also our main way of communication as well and we used it alot to have our online meetings
- Had to change the part regarding the use of the intelliJ IDEA CE as every team member used the IDE they're most comfortable with, and also mentioned the use of source tree and github desktop as tools that helped in collaboration.
- Did not change the part regarding the use of google drive as it was what we used to share documents and everyone was familiar with using it.

These were all the changes that had to be made regarding the tools and methods used. Part B which is regarding team organisation does not need to be changed as the approaches mentioned by the other team are very similar to how we operated for this assessment.

Planning:

There were not many changes to be made regarding the plan for assessment 2 As we used the same agile methodology as the previous team. We chose to continue with the use of advanced gantt charts rather than standard charts as they provide a better understanding of the workflow. They also provide additional information such as who is working on each section. These gantt charts were updated every week in order to plan ahead and account for changes and delays in the plan, these can be found on the website. (Found at https://eng1-team5.github.io/)

However, we did not keep a logbook as we did not feel it was necessary alongside gantt charts and our use of github issues.

IV. Changes to Risk Assessment and Mitigation

Found here : https://eng1-team5.github.io/static/A2u/Risk2u.pdf

When updating the risk assessment and mitigation plan, we started by comparing the risk register to the updated requirements and removing any risks that were associated with now removed requirements, such as rR3 which was related to controller support. We also added some risks (rP6 and rR7) based on both the new requirements and feedback from our original risk register.

Additionally, we adjusted the likelihood and severity of some risks (rT3, rE1, rE3, rE4) based on our personal opinions on how likely the risks are to occur and how much we think they would affect us. Finally we added colour coding to the likelihood and severity columns so that the high, medium, and low ratings are clearer.

Finally, regarding the introduction of this document, not much was changed apart from explaining the colour coding used in the document and how it can provide a view of the severity of a risk at a glance.