Change Report

Mario

GROUP 28

Joseph Frankland

Anna Singleton

Saj Hoque

Leif Kemp

Shi (Lucy) Li

Hugo Kwok

Introduction

During the takeover, we split our team into two groups, where one is responsible for the coding changes and the other one is responsible for documentation and updating all the deliverables.

For updating the deliverables, the Requirements and Risk Assessment is first updated as they outline the work needed to do for Assessment 2. Afterward updating those documents, we decided to split the subgroup again into two pairs as we do not think that the remaining documents will need four of us to work on at the same time, so splitting into two subgroups, one working on Methods and Plans, and one working on Architecture, would make updating more efficient. The general steps we take to change a document is to first read through the entire document and note down what needs or might need to be changed, and then make a temporary change on the existing, and note down the changes in a change list on a separate document, so we can refer to it later. Some documents like Requirements require a huge amount of changes that we decided to redo the whole document from scratch as the content will be more cohesive than if we just changed a lot of details in the original document. Afterwards, we would review the document we've just changed and have a final look through for any additional changes before we write it to the change report.

In terms of the programming, our approach was to collect everything that needed doing onto a trello board, and assign tasks to ourselves to keep track of who is doing what. The general steps a ticket on the board would go through are being assigned, then being written into the code, and tested either manually or via unit testing once that was implemented. It would then be pushed up to its own branch on github, then a pull request would be opened and the other member of the programming team would review it, and agree to the changes. The changes then would be merged, and the CI pipeline would take over. Tickets on trello were assigned priorities ahead of start time in order to evaluate when they should be done. Some tickets were also marked with a waiting status to represent they were dependent on other tickets. Certain tickets such as CI or UI tickets were assigned to Anna or Leif respectively ahead of time since it takes into account their specialties.

Requirements

Once receiving our new requirements from the stakeholders, our first step was to incorporate these into our newly acquired requirements document. Before tackling this, we made any adjustments to how the requirements were originally elicited and negotiated, and why they were presented the way they were. However through a thorough examination of their elicitation we found that our methods to approaching this were almost the same however it was not clear why the approach was taken and how well this approach worked. Hence using our ideologies we incorporated an explanation as to why we chose to use “open and closed questions”, also going into detail as to why it was much preferred to our initial approach, which was merely “closed questions” - this was backed up by our experience and information collected with asking both types of questions in the interviews. Including this justification, it aided us in understanding and reflecting on what system works and why, when revisiting these stages of development. Other than this change, we had minor changes involving a correction of the definition of NFR as we felt the original one was not satisfactory and didn’t include its several factors such as “security, reliability, performance” etc. In addition to this, we felt as though the introduction to the requirements tables was nowhere near explicit enough to accurately establish how they’d be presented. Thus we integrated a paragraph near the end of this introduction, to describe the contents of our tables that were to be presented. This established its format and what each column meant as well as its links to other columns so that the presentation of our requirements could be easily understood and visualised by any reader. We had enough space to include all of this information due to being able to remove invalid statements or sections, in particular “noting risks and environmental assumptions associated with each requirement” - we felt this was inexact to keep in due to our change of table structure as described later in this change report. This would eventually see the environmental assumptions and risks to be separated from the tables and given their own segment of the document as this allowed isolation and emphasis on more apparent requirements. Removing these segments provided us space for us to include more apparent ones, which comes to our last change to this introductory segment. We found this to be the lack of evidence of research into requirements specification and presentation. We agreed that it was obligatory to mention any form of evidence for research into literature and relevant standards which we utilised before we started the documentation and therefore mentioned our usage of “IEEE 29148-2018 - ISO/IEC/IEEE International Standard - Systems and software engineering - Life cycle processes - Requirements engineering” - which we were able to access for free due to our university status. This literature allowed us to professionally establish the methodology behind how we were able to appropriately elicit and display our requirements and the process behind it. This concludes our changes to the introductory aspect of our requirements document.

When we studied the tables of the requirements document we decided it would be best to update the layout used. We stripped off a couple of columns as they felt unnecessary in the table, which were the alternatives/risks column and the environmental assumptions column. These topics could be addressed in a shorter way as a paragraph of explanation by themselves. Newly simplified, the tables were a lot more readable with a lot less text, meaning that you could find the information you wanted faster. Previously there were only two tables with user requirements and system requirements respectively, functional or non-functional requirements were distinguished by a column in these tables, which led to difficulty finding specific requirements. So, in our redesign we split up the tables into three, User Requirements (UR), Functional (FR) and Non-Functional Requirements (NFR). The UR table has three columns, the ID, description and priority level for implementation. We chose these as they seemed most fitting to present the necessary information relating to these requirements. The IDs are structured like “UR_NAME” in which NAME is something that describes the type of feature we are required to add to the game. This was changed from just numbers as it makes searching for the requirements a lot easier after they’ve been written down whilst also giving the ID’s a better use of identifying or rather giving a brief view of what the requirement portrays. The FR table also has three columns, ID, description and which UR it relates to. The FR names are structured the same as UR ones, we chose to link to the URs as we felt it would help with tracing what we’re doing and allow us to see the progress we’re making over time. The NFR table has four columns, ID, description, URs it links to and the fit criterion. NFR ID naming policy is the same as URs and FRs. The fit criterion was incorporated as it is hard to properly measure if a non-functional requirement has been met, so, these criteria are there to help us assert that they’ve been successfully implemented. The changes to naming format were done so that requirements can be easily inserted and edited anywhere, without having to change ones below it. Requirement descriptions could also be changed as the names are meant to be a short version, therefore, if a new feature is required that's an extension of another it can be updated without having to create a new requirement. This allowed us to accommodate for requirement change management - where requirements can change as a result of changes in the environment, or changes in user expectations/preferences etc. Hence this new simplistic, readable format of our tables allow for us to easily edit these requirements and incorporate new ones if needed. Another thing we had to do whilst changing the table layout was decide on if we wanted to change some of the original requirements and their descriptions. Some were kept the same whilst others that were appropriate were extended instead to keep up with the new requirements. This would help keep all related things together and help with readability.

After recreating the tables, we added paragraphs talking about topics that we think the document lacks and some items that weren't suitable to put into the tables. We used the definition of constraints in “IEEE 29148-2018” to identify the key limitations that we may encounter during the process, which are mainly the possible problems due to different availability of team members, the time we have to complete the project, and the requirement for the game to work on various desktop platforms. We then suggested possible solutions to work around the corresponding constraints. Then, as above mentioned, we summarised the alternatives/risks column and the environmental assumptions column in the original tables and put them into a paragraph. We decided to do this because we think that instead of adding them to the tables, which was too compact for presentation, concentrating them into a paragraph would be best - allowing us to focus on more apparent requirements. Furthermore, it also eliminates some duplicated items on the tables such as the risk of “feature creep” being mentioned several times. As we are summarising we also removed some alternatives and environmental assumptions as some of them are unnecessary for their corresponding requirement, for example, the environmental assumption of UR_MOVEMENT (UR1) was written as “Assume the user has access to a keyboard and mouse.”, which is obvious and unnecessary to be put on the table and so it is removed along with other items.

Risks

Our new Risk Assessment and Mitigation document kept a similar format however we felt there were certain changes that were necessary. Before any changes were made, we examined the risk table and added any relevant risks that were not displayed. An important one that was added was inspired by the process of applying changes to all these documents which was “Data being lost / mishandled” - this risk was configured as we used backups frequently when making changes to the documentation for the change report. Hence it was deemed necessary to add because of the frequency of its usage and was given likelihoods and severities with respect to this.

When adding new risks it came apparent to us that the types of classifications given by the previous group seemed lacklustre and vague. To rectify this we considered all possibilities and came up with different classifications whilst still including some of the previous types. Classifiers added were Product, Project and Business risks. These classifiers allowed for a more broad representation of our risks whilst also giving them a related category. Each of these categories describe independent factors whilst also being easily linkable for our risk table allowing us to combine categories in our risk table. This increases our viewability and furthermore increases our efficiency when it comes to applying likelihoods and severities to our risks. As well as this it allowed us to effortlessly divide risks to appropriate owners in relation to a collective of classifications. This was previously quite vague and broad and specified roles rather than actual owners. We felt as though this was not very appropriate, and in addition to this it was not very clear who belonged to each role. Instead these risk owners were identified by name, and this process was made quite straightforward by our classifiers as we allocated specific types to specific team members based on their quality and contributions to the project.

These edits allowed for efficiency and readability but we realised this document did not answer part of 5 (b) on Assessment 1. They included a type, description, likelihood, severity and mitigation for each risk, but did not include their impact on the project and its factors. The impact allows us to display how each risk will affect the project and its other factors giving us an overview of how severe this risk will be .This was a significant addition that we felt was essential to the risk assessment, because it allowed us to visualise how necessary it was to mitigate the specific risk and it also gave us the ability to gauge the severity of the risk with much less effort. This also improved the table's visibility and gives the viewers a much improved understanding of the risks with the addition of these columns.

There was no section detailing how the risks were identified and analysed so we added a paragraph describing how we did it. This meant describing the process we used to find possible risks and what we did once we had discovered them, such as ranking likelihood and severity as well as writing down what we would do to mitigate the impact of the risks. Doing this felt necessary as it would allow us to set out what was needed to stop catastrophe from happening from the very beginning of the project. Other than risk identification, there was also no mention of what risks actually occurred during development, so we added a section listing out which risks actually occurred during our development. We explained how each risk happened and how they impacted our pace. Afterwards, we also showed how the mitigation plan helped us to minimise the impacts, further reflecting how the risk assessment helped us throughout the project.

Method Planning & Selection

Before editing our method planning & selection we ran through the previous teams methodologies and noted any similarities and what changes we would need to apply. The first change we felt necessary was that there was no mention of a collaborative tool for writing the documentation of deliverables. We all agreed on using the Google Docs as our piece of software for editing our deliverables and added this section to be completely transparent on how we create our game and our deliverables, making it easier for people to see our workflow. Whilst not only providing a platform to synchronously collaborate on the writing, we also noticed it had a useful feature that we felt was instrumental to our documenting on this assessment. The ability to reform backups from edits we made to the documentation made writing our change report significantly easier, whilst also allowing us to accurately keep track of our work, with logs of what people had edited. We made sure to talk about any other options we considered when writing our deliverables in order to give the reader a scope of understanding towards our decision making. We felt that this section as a whole was a necessary addition to the methods and planning because of its importance in our project and how often it was used. It was a rudimentary tool that was used throughout both assessments starting from designing our plan to executing the final documentation.
We also decided to change the IDE we use for development, from Eclipse to VSCode. VSCode was chosen over Eclipse as our development team has more experience with this IDE, therefore, we felt it would allow us to be more efficient as we wouldn't have to spend time getting to know a new IDE as well as having to learn about the codebase, which in itself is a huge undertaking. This was a necessary change as this IDE provided different benefits to us than it was previously described in the documentation and we needed to rectify this. We kept a similar format to how it was presented but ensured the correct information was instead displayed for the IDE we were planning to use.

One thing we didn't change was the game engine we'll be using. If we did change the game engine, we would have to restart the whole project, which isn't feasible in the short amount of time that we have. So, keeping the game engine was essential to keeping the game finishable within the time limit. We also have experience with this specific game engine so we can use our experience to get to grips with the codebase faster and add the new features as fast as possible. This decision showed our capability of being able to use the resources presented to us and work around any difficulties it may have provided. Communication amongst members of the team was also kept the same. Whilst considering the alternatives, we agreed with the usage of Discord to communicate and set up meetings in person as well as online. This wasn't changed as every team member already had experience using this program so there would be no problem communicating on how we would be going forward with the project. A big change we made was to the methodology of the project. We went from using a combination of Agile and Waterfall to just a focus on the Agile methodology. Agile has become the sole focus of our project as it allows us to quickly finish different features and then move onto the next part of the project knowing one requirement has been fully completed. It also means that we can go over a feature and make sure there aren't any glaring bugs in the code and if there are they can be easily spotted and fixed. This change was necessary; it was the form factor of our whole project, and projected how we worked and devised our plan (with how it worked with its key tasks and time management) very closely according to this methodology.

We also provided our tools that we used when designing diagrams and our plan (key tasks, due dates, delegating tasks, etc.). We found there was no previous mention of any tools used in aiding the design of the architectural diagrams - this seemed like a significant so we made sure to involve our usage of PlantUML as it provided us with an easy interface to express every method and class attribute on the concrete model meaning that anyone who views it will understand in detail how our project works and be able to recreate it. The way of delegating tasks has also changed, instead of using a simple spreadsheet, we use Trello. The swap happened because Trello offers a lot more features than a simple spreadsheet, we can set due dates for tasks and then receive reminders via email just before its due. As well as this it's easier to give tasks to specific members as they can join a workboard and assign tasks to themselves or be assigned tasks by the owner of the workboard or other members of the board. We found this change necessary because we followed a similar methodology of tracking tasks with spreadsheets and it was not feasible, and it did not feel transparent with the entire team. This gave us a synchronous method of keeping on top of tasks whilst also giving us a platform to propel any issues people may have on specific tasks that other team members may be able to provide insight on. With this in mind we realised Trello provided us with a much more whole rounded purpose in the management of tasks. Considering this we decided that Trello would act as a better replacement for the gantt chart. Its easy collaborative abilities played as a deciding factor during this decision. In addition to this, trellos interactive gui made it simple to edit dates and provided existing features such as adding checklists and comments - which all in all proved for it to serve as a better tool and was agreed among all team members.

The organisation of the team changed significantly between stages of the project. The biggest change was the fact that we no longer have dedicated team leaders. This was changed as we thought it would be more efficient to have everyone on equal footing when it came to what tasks they want to do. It also means that there would be less friction between a member and a leader if they disagree with something, as decisions would be made by committee. One thing we kept the same was the splitting of members into two teams, deliverables and coding. This was kept the same as it meant people could focus on one type of task and make it one that their strengths align with. Therefore, making the work be done quicker at a higher level as everyone is very comfortable with what they're doing. The section about stakeholders in this section was fully removed, this was because it felt unnecessary and was irrelevant to the organisation of the team and the inner workings of how it was run, whilst also allowing us to free up space for more related additions to the document.

Little things were added onto the document, this involved things like adding task assignees that were more specific as it was not clear who the “developers” or “documentation” team were. Identifying the members for the tasks that were set out, provided a sense of responsibility to the respected member to ensure that task was completed. We found the task table to be informative, however felt it was not very clear how the priorities were achieved, hence adding a small section near the end of the document explaining how each task would achieve its respected priority rating. This allows anyone who views it to be familiar with the table and understand the justified emergence of each task rather than assume they are random priorities based on preference. Lastly we felt the need to display how our agile methodologies reflected on our plan. This seemed necessary in order to see how well our approach worked and whether it was effective or not.

Architecture

A change that must be done in the document is to update the diagrams for abstract and concrete architecture so that it fits to the final product as new classes are created for the new requirements. Some indirect changes were made in the document. We replaced all the requirement names mentioned in the document as the requirements were revamped as mentioned above, eg. “UR1” was replaced by “UR_MOVEMENT”, “UR2” was replaced by “UR_COMBAT”, etc. We also removed sections that talked about requirements that have been removed as they are no longer relevant - for example, in the table at the end of the document, the sections regarding UR10 and UR12 were removed as they were requirements that we discarded for being obvious and unnecessary. Afterwards, we extended the table by adding the new requirements and explained how they were implemented - just like the previous requirements in the table - showing some of the design decisions made along the way. Minor changes were made as we found some elements to be inaccurate. For instance, the document mentioned the requirement UR_PLATFORM was “met by choice of library - Libgdx - which is OS independent” when it should be the language Java that is OS independent. Some explanations were also removed as they were unnecessary - such as where the document mentioned they designed the diagrams so that if someone came across it they would “easily understand in detail how our project works and be able to recreate it”. The documentation should be clear and easy to understand by default, therefore it was removed.

Overall, minimal changes were made to the architecture document for a number of reasons. Firstly, both abstract and concrete class diagrams were well presented and did not have many errors or inconsistencies within them. Therefore the only changes needed include adding the new classes and methods regarding the new requirements from the customer to the already existing diagrams. The description of the association of each class is well donel with examples. Justifying the concrete model was useful as it explained the naming system with keys making it easier for the readers to understand. Furthermore, it also clearly discussed the transition from abstract to concrete model by detailing how the concrete model was built up on the abstract model. Lastly, the table which showed links between the requirements and the concrete designs was informative, as it described the design decisions that were made which informed how each requirement was implemented.