Change Report

Cohort 1 Group 8

Roy Asiku, Tianqi Feng, Andrew Jenkins, Nicholas Lambert, Tom Byron

Communication and Project Transfer
As a team, we chose to use Discord to communicate easily about any topics we were unsure about, to ensure every group member was aware of any current discussion about the changes made, or steps being taken by the group to modify the project as we saw fit. We also used it to hold scheduled meetings to discuss what action should be taken next. We then noted down what was mentioned in the meeting, as well as the next steps agreed upon by the group in a separate text channel.

We used UML charts to schedule any tasks individual group members would be doing, allowing us to keep track of the group's progress and focus the resources of the group on any specific tasks that may be behind schedule, for example reviewing the documentation left by the previous group.

We made and kept track of changes to the implementation section of deliverables by pushing any updates to the Github page, where we could all view an up to date version of our current version of the game, and then hosting a meeting on discord to discuss if necessary. We ensured that each git commit underwent prior testing, and any issues with the code could be discussed during a meeting if they were not able to be simply resolved.

We also used Google Drive to make changes to our assessment deliverables thanks to its simplicity and ease of use.

Requirements
Previous Version: https://st1835.github.io/files/Req1.pdf
New Version: https://nicholaslambert03.github.io/ENGpart2Website/files/Req1.pdf

When taking over the requirements for assessment two there were two main points to cover. The first was standardisation, simplification and decomposition of the already existing requirements. The second being the addition of new requirements given to us within the expanded project brief for exam two.

For standardisation, many of the requirements that logically sit within the same modular components of the game had overly convoluted names which made it difficult to tell what the requirement was going to be at first glance. To counter this a standard naming convention of REQUIREMENT TYPE _ MAIN TOPIC OF REQUIREMENT _ SPECIFICS has been implemented, this not only makes requirements easier to understand and recall, but has the added benefit of allowing requirements of the same grouping to sit together in an alphabetically sorted table, making it easier to evaluate those groups.Furthermore certain suffixes were removed from requirements such as 'ing' to introduce an easy to predict and match naming convention.An example of all of this is changing UR_SCORING and UR_HIGHSCORE, to be UR_SCORE and UR_SCORE_HIGHSCORE. This makes it easier to locate any requirements to do with the score which will be convenient for testing. Secondly, I renamed the priorities from May, Should and Must to May, Should and Shall, as the level of implication of must and should was deemed too similar and didn't indicate the level of importance required.

Many of the internals of the requirements or even entire requirements were redundant and implied by other requirements or describing procedures to be followed which are not

requirements of the system. To fix this, I either combined requirements into each other if one implied the other or removed requirements that were redundant. An example of this would be combining UR_AGE_RANGE, UR_GAME_STYLE, and UR_GAME_TONE into UR_GAME_TONE as the age range and the game style were all implied within the tone requirement. And removing requirements such as UR_0_ENERGY which was implicit in the UR_ENERGY and UR_ACTIVITY_TIME.

Then requirements that deserved multiple requirements were decomposed into multiple requirements and in some cases new requirements categories. This helps improve our testing as the individual subsections of the previously overloaded requirements can be ensured. An example of this is the UR_ACTIVITY becoming UR_SLEEP, UR_EAT, UR_RECREATION and UR_STUDY and all the subsections derived from those four.

After this reformatting, came the introduction of the new requirements. To facilitate the addition of the need for achievements instructed by the updated brief, UR_ACHIEVEMENTS was added with the functional requirements FR_ACHIEVEMENTS_TRACK and FR_ACHIEVEMENTS_SHOW. These include the introduction of achievements, their tracking throughout the game and the user being able to view them. Along with this, UR_ACHIEVEMENTS_COUNT was added ensuring it was noted that there must be at least three achievements. To facilitate the new requirement of a leaderboard UR_SCORE_LEADERBOARD was added with the functional requirements FR_SCORE_TRACK and FR_SCORE_SHOW, whose functionalities are similar to those described earlier.

Architecture
Previous Version: https://st1835.github.io/files/Arch1.pdf
New Version: https://nicholaslambert03.github.io/ENGpart2Website/files/Arch1.pdf

Upon first receiving the project and analysing the previous team's architecture, it was very difficult to understand how the game's different classes interacted with each other. Many functionalities were not split into distinct classes - there was instead a lot of overlap between classes. This, in combination with the project having no Javadoc or comments, made it crucial that we determined how the game was structured. Due to these issues, the first thing we did was go through the code and add Javadoc and comments to all of it, which gave us a better understanding of its composition. Since we were already familiar with UML diagrams, and the UML diagram from the previous team looked incomplete and unclear, we decided to rework it.

Due to the project's issues with overlapping, and the code having a very large number of classes, it seemed more appropriate to group the project's structure by package, as it reduced overlapping and made it easier to track how everything was connected. We also needed to add our requirements for part 2 into this diagram, FR_LEADERBOARD_SHOW and FR_ACHIEVEMENTS_SHOW, which neatly fit into the Helper group. The addition of comments describing each group's role also helps to understand how it operates.

The sequence diagram that the previous group has created, on the other hand, we thought was very clear and as detailed as it needed to be. As such, we decided not to alter it, as the

main functionality of the game was already implemented, and we did not add anything that would cause a change to this diagram.

The CRC cards the group created were somewhat helpful in determining how classes are grouped, but it is again seen here how many classes are in many different groups, due to each class having such a wide range of functions. We believe that the comments and the layout of our new UML diagram displays the grouping and roles more clearly than these cards, and can be used instead.

Method Selection and Planning
Previous Version: https://st1835.github.io/files/Plan1.pdf
New Version: https://nicholaslambert03.github.io/ENGpart2Website/files/Plan1.pdf

Firstly, it was important that we update the method selection and planning document to include the planning behind the second half of the assessment. For this we decided to use Gantt charts. This is for two main reasons, the first being simplicity, not only does it provide easy visual clarity of the plan to all group members, but as a UML based diagram it also allows for easy creation and editing making it the perfect tool for changing schedules. Secondly, it helped maintain continuity in the report as the group before us had also used Gantt charts. This is an added bonus, although we agreed we would not let continuity get in the way of a much better solution.

Secondly, in places where we decided to use different methods, such as discord for communication instead of slack, or thought methods needed better clarity, mentioning Github and not only the use of Git, the document was updated in italics to show the areas that had been written. This helped separate our methods from the other groups, where we saw modification as a different approach offered a considerable benefit, which aids the clarity of the report, and keeps method documentation accurate.

Finally, we added in our plan for part 2 and its evolution throughout the project. The plan had less iterations than expected, as we were generally more organised than we were in part 1, and found that we gave ourselves sufficient time for most sections of work.

Risk Assessment and Mitigation
Previous Version: https://st1835.github.io/files/Risk1.pdf
New Version: https://nicholaslambert03.github.io/ENGpart2Website/files/Risk1.pdf

We chose to make no modifications to their description and justification of the risk management process followed, as we felt that they, for the most part, adequately and concisely explained their justification for the processes they chose to use, and we have used similar processes in order to manage any risks threatening our project.

The exception to this would be that the format of the risk register was not explained in a large amount of depth, so we chose to expand on this. Specifically, we added more explanation behind why each risk was assigned an owner.

Next, we added colour-coding to the risks, simply to make the table more intuitive to use and user friendly - this was a minor change. The risk register was already systematically laid out

and well organised, so no other changes to the format of the risk register were made. A less minor change we made was adding a third type of risk to the risk register - we added the ‘business’ risk type to the table, as we believe that the previous classification of the risks into ‘project’ and ‘technology’ was limiting our ability to conduct risk identification and analysis. This is because risks concerning areas such as the demographic of the game or being sure the game is in no way insensitive, do not fall under either of these two categories.

We also added two new risks to the risk register, as there were only 5 risks before, which was limiting our ability to monitor and mitigate risks - as there were some that we were not aware of as a team. One of these risks was under the ‘business’ category, which was newly added - the other became more relevant since we were inheriting a game from others - this was the risk that the documentation of the code was limited or low quality. This could prove to be an issue for the group as we may not have been able to understand properly how the provided code worked, slowing down the implementation process massively.

We also changed the owners of the risks, obviously because the team members are now different, but also because they were previously assigned to ‘everyone’ apart from one risk. We believe that assigning almost all risks to every single group member has more drawbacks than positives.

First of all, while the ‘bus factor’ is very high, as if one group member was unavailable for any reason, there would still be a large number of group members assigned to a task, the fact that everyone is assigned to a task would lead to a lack of clarity over the job of each member. We believe that it doesn’t make sense for every single team member to work on every single risk’s mitigation, as each member would have too much to focus on, while not playing a significant role on their own in the mitigation of a risk. This is something that we as a group particularly cannot afford to do, as we are only made up of 5 group members, and therefore the proper management of our resources as a group is essential. Instead, we assigned multiple group members (not everyone) to each task. This ensured that the ‘bus factor’ was still high, and it was exceedingly unlikely that a task would be left without an active owner, but the efforts of each group member were more effectively spread.