Change Report

James Smith
Amanda Ling
Fran Medland
Hannah Vas
James Kellett
Malik Tremain
Mischa Zaynchkovsky

Summary

The process we took involved using the same documents the other team used and updated them where necessary. The changes we made into the documents shows either the old format/method the other team used which we implemented and followed the same format, or a justification of why we have approached this using a different method. For example, we approached the method selection and planning differently to how the other team approached this in the first assessment, so we decided to follow our original method. This was to ensure the project goes smoothly which reduces the amount of risks that could potentially come up. Whereas, the other sections mostly follow the same process so we tried to update the documents in the same way to emphasise continuity. Overall, there were limited changes we found to the existing documents. From here we added new information based on the new specifications in assessment 2.

When we updated these old documents, we updated any changes we made in blue to show the difference between the two reports and the change we have implemented into the project. We approached it this way to show us easily what differences we have, ensuring that we meet all the new requirements.

We made a new Gantt chart every week (See Method selection and planning below) which allowed us to review changes easily on a weekly basis. If any new changes came up, then we updated the correct document. We used all the documents/code from the other team. For the code, we cloned the repository to get the previous teams game/code. Our changes from the code can be seen through our commits on GitHub, which are all named to state what changes were made. For clarity within the code, we continued to use the previous teams naming conventions rather than switching to different naming conventions that would more clearly designate the code as ours. To emphasise the differences within the code we put
‘// Added code’ for single lines of new code
and
‘// Start of added code
…
// End of added code’ for updated code blocks.

By looking at the commits and the blue text on the documents, this allowed us to keep track of any changes efficiently. This meant when we reviewed any changes, it was easy to see how it is different to the previous teams documents/game. This meant that when we checked the assessment 2 brief, it ensured our changes meet the additional requirements.

Requirements

We followed the process of taking an iterative approach to the requirement elicitation to meet the needs of the client and the new features. We previously used this same approach which worked well for us, so we decided to continue this same process. Due to this, we didn't think we had to change the report significantly. The significant changes we implemented were on the necessary tables. (Shown on Section 2.2, 2.3 and 2.4 in the Requirements document)

After receiving the additional features to add into the game, we updated the necessary requirements. This included adding two new user requirements- UR_LEADERBOARD and UR_STREAKS. As these requirements were key features to add into the game, they were both under the high priority 'Shall'.

From here, we updated the necessary functional requirements, which involved the new user requirements. UR_STREAKS was added into both FR_SCORING and FR_END. UR_LEADERBOARD was added into FR_END.

We added a new column to the Non-functional requirements to show what user requirements were used and necessary for this particular non-functional requirement. We thought this would be important for traceability. Due to this, we felt it was necessary to add another user requirement as none of the current requirements fit well with the non-functional requirements. This new requirement was UR_SYSTEM which checked the game should run on different systems, allowing an ultimate experience and enjoyability for the user.

We also updated the description of some user requirements. These included UR_STUDYING, UR_RECREATION and UR_EATING. From the initial product brief, each activity only needed one building/location. From the new product brief we added more buildings so the description was updated to show the additional building to add.

We implemented the scores different from the other team, so we updated the description of UR_PLAYER_SCORE and FR_SCORING. This was more efficient this way due to the new requirement of streaks. The streaks score could make the score go over the intended maximum of 100, meaning the score is not based on a percentage anymore. We also got rid of the pass mark being 45. We decided to pass the user based on the user's ability to complete all the specific requirements within the game. So, if the user doesn't study for 2 days, then this is an automatic failure rather than having a figure of 45 for failure.

From here, we added a few more functional requirements. These included FR_TEST_ENERGY, FR_TEST_LEADERBOARD, FR_TEST_SCORE and FR_TEST_TIME. These requirements were used to show the tests we were going to implement. These helped us to test the game works, it is functional and it meets the brief of the assessment.

Architecture

Updated Behavioural Diagrams
In the first sequence diagram, Assets exists as a participant however, in the actual code the class Assets does not exist. In the updated sequence diagram, the participant Assets was removed along with the message "**8** Render UI" as "Render UI" is now represented as an operation in GameScreen. The sequence diagram needed to be updated to reflect the current game design more accurately.

Original final sequence diagram, part 1
Updated final sequence diagram, part 1

Updated Structural Diagrams
With the addition of new user requirements, UR_LEADERBOARD and UR_STREAKS, new classes were implemented to meet them, classes Leaderboards and Score. The class Leaderboards allows for efficient score management where scores are stored and retrieved from an excel file. The class Score allows for score calculation with streaks being tracked for additional points. The class diagram had to be updated to reflect the current structure of the game design.

Moreover, the previous architecture design was UI heavy, but with the introduction of classes Time, Energy and EnergyTexture in the entity package this balanced the architecture better. The new classes were introduced for a more efficient game system as previously the roles of these entities were handled in a single class, MainGameScreen. Having separate classes for different functions made for better structure.

Additionally, with oversights in the previous team's code regarding the method 'touchDown()' in the MainGameScreen class, the class Button was introduced to increase extensibility, reduce code duplication and allow for better code formatting.

With the introduction of new classes, the package design of the class diagram also needed to be re-organized with the GameMap package removed and relocated and the button package added to better the readability and understanding of the class diagram.

Original final class diagram
Updated final class diagram
- UML sound package
- UML utils package
- UML entity package
- UML button package
- UML settings screen package
- UML menu screen package
- UML control screen package
- UML typing game package
- UML game screen package
- UML end screen package

Updated Requirements Relating to Architecture

Previously, the traceability to requirements was not quite clear because only a few of the FRs and NFRs had been covered. To show more clarity, more FRs and NFRs were discussed in relation to the architecture.

Any missing FRs and NFRs is because:

- UR's taken only from the 'shall' priority
    - FR_SAVE_PROGRESS (UR_OBJECTIVE - 'should' priority)
- Not architecture related (coding/testing)
    - UR_SYSTEM - 'shall' priority exception
    - NFR_JAVA_VERSION (UR_SYSTEM)
    - NFR_SYSTEM_SIZE (UR_SYSTEM)
    - NFR_MAINTAINABILITY (UR_ACCESSIBLITY)
    - NFR_REALIABITY (UR_ACCESSIBLITY)
    - NFR_COMPLIANCE (UR_SYSTEM)
    - NFR_COMPATIBILITY_SUPERCEEDED (UR_SYSTEM)
    - NFR_COMPATIBILITY (UR_SYSTEM)

Method Selection and Planning

Software Engineering Methods

We decided to Specify that as a team we would follow a scrum method, specifying the agile software engineering method that the previous team had chosen. We had previously used this and it worked well to ensure that work was kept on track and keeps a high level of communication between team members, which for a fast paced project we felt was vital. However most of the rest of the adjustments made were minor, for example having more of a focus on online meetings rather than in- person meetings as this worked better for us, and having our meetings on slightly different days.

Development and Collaboration Tools

As a team we then further discussed the development and collaboration tools that they used, and compared them to the ones that we had previously used to see what would be changed. We especially looked into an alternative to Git, as this was our previously least researched tool, and the other team also had not looked at other alternatives either, however after some research we decided that for our project that was the best choice.

As all the previous code and designs were made using specific applications, we felt it was too much of a risk to switch to other development tools, even if there were better alternatives, that we decided not to research for example using Tiled to edit the game map.

Approach to Team Organisation

We took a slightly different approach to our team organisation, and just slightly adapted what we had previously done, rather than switching to the other teams method fully. As they seemed to run into a couple communication issues. We did decide to incorporate some ideas that they used, but more informally than the very structured and defined approach to organisation.

Systematic Plan for the Project

We decided to incorporate a more formal Kick off meeting to begin, rather than the more informal discussion and task allocation that we started off with last time, as we just had to assign tasks later on anyway. This then created our initial work breakdown chart and first Gantt chart that we decided to include on the document, this clearly stated and defined responsibilities and an approximate timeline that we wanted the project to be completed in along with some smaller tasks too.

Every week when we held our main meeting, we outlined what we had completed, revised the tasks left to be completed and if the timelines were manageable. This helped us stay on track and quickly spot if any issues were causing us to not meet our specified deadlines. In our secondary meetings, we more informally discussed progress, any issues that arose would be handled and if this caused our weekly Gannt chart to be incorrect, it would be corrected in the next meeting.

Risk Assessment and Mitigation

Risk Assessment Process
Overall, the risks that were already in place were good and we thought it wasn't necessary to change the report/risks significantly. So the changes we implemented were very minor. We aimed to follow the same format the previous team implemented to show consistency within the report.

Risk Register
For the old risks that were already in place, we added new owners for people in our group. This allowed us to be aware of any risks that might have come up with the previous group, and if they come up for us, we will know how and who will act on this to try and resolve the risk as quickly and efficiently as possible. There wasn't anything else to add based on the old risks so we moved onto new risks.

As there was a new brief and new aspects we have to cover from the assessment, we added new risks that might come up and the different work we had to do. For example, testing was a significant part to do now, so we made sure there were risks to cover everything we did to ensure everything went as planned.

We followed the same format, showing the likelihood and severity of these new risks and once again assigned a person to them. Also, making sure that we have mitigations written to avoid risks coming up. These features were all easy to follow on from the same format.

References
Another change we implemented was references. This was not done previously, so we added a page at the end with references. This was to allow us to check a particular plan (A gantt chart for example), if we encountered a particular risk.