Change Report

Group 12

Team 12

Daniel Agar

Charles Thompson

James Leney

Leyi Ye

Matt Smith

Vanessa Ndiangang

For assessment 2, the deliverables followed a waterfall-esc pattern, as many of the updates to the deliverables depended on the requirements but more so on the implementation. The requirements were consulted initially, as was the updated brief, to gauge an understanding of what had and hadn't been completed by the previous team. If those requirements could be updated, then attention could be turned to the implementation. To coordinate the completion of tasks, we adopted GitHub Issues as a means to break up the task at hand into sub-tasks that everyone could complete individually, maximising utilisation of each team member's available time. This also helped when integrating changes into the codebase, as conflicts became a rare occurrence as our tasks were specifically designed to not share domains. For example, scoring and leaderboard-related tasks were split up as the vast majority of work towards both of them was separate. As changes to the implementation progressed, a larger pool of information became available to inform the work on other aspects of the project. Being heavily dependent on the implementation changes to architecture naturally came after a short lag time caused by development, although changes weren't instantaneous upon changes to the implementation as, especially early on, the direction in which the code structure was going was quite unclear with big decisions to be made in order to decouple and refactor certain sections. Testing began at much the same time as the implementation work, as the previous groups' work could be used as a basis to start writing tests. This became crucial as changes to make the code more testable could be put forward early to ensure that the finished product would be as testable as possible.

Requirements:

- Original Deliverable (Req1)
- Updated Deliverable (Req2)

The requirements deliverable required little change, we continued to elicit the individual requirements in the same way, consulting the brief and then taking queries about the finer details such as "should the user be forced to sleep?" To the customer to gain a further understanding of how the feature in question should be implemented. To ensure we continue to follow the previous groups practices we found a book[1] which outlined the groups elicitation techniques in detail for us to refer back to. Naturally with an expanded brief our

requirements tables had to be updated. This process was again the same as the previous groups with user requirements being written up first then linking functional/non-functional requirements being written from this. Examples of this include the new entries in the tables to encompass everything needed to outline what the leaderboard/streaks functionalities should look like to the user and how these should be implemented in relation to the technical details.

Architecture:

- Original deliverable (Arch1)
- Updated deliverable (Arch2)

With the addition of new classes, the diagrams which described the architecture of the implementation required alterations. The new product brief required us to create a leaderboard with the name and scores of the 10 best players who successfully completed the game. To save the player's score, new classes such as GameData, Save and Score were implemented. This allowed for each player's score to be stored on an external file, which could then be read into the game and create a leaderboard for the user to view. As a result of these new classes, the class diagram had to be changed to show the new associations between classes. The creation of the new classes also resulted in amendments being made to the original classes to account for the new functionality, such as the ability to earn streaks and the full constraints of the player's score calculated at the end of the game. The new class diagram therefore shows more class properties and their corresponding access control information.

The new brief also requires the player to achieve "Streaks" - an award given to the user for completing specific tasks each day. Achieving streaks required the game to track the activities completed by the player each day, which was implemented through the existing classes Day and DayManager holding more attributes and methods that despite increasing the classes' complexity, simplified the process to understand which streaks were to be displayed to the user.

We decided to extend the main menu with a new button to view the leaderboard as well as making all views prior to playing the game have the same background and UI theme to create a more enjoyable experience for the user. This therefore resulted in adding more states to the Screen state diagram to reflect the improved UX. With new requirements FR_SCORE and UR_SCORE elicited, the gameplay state diagram also had to be updated to reflect the more detailed tracking of the player's activities.

Despite the previous group suggesting an ECS architecture with the component diagram and instead deciding upon an OOP approach in the previous deliverable, we felt it wasn't necessary to include the section in the updated version due to it being unrelated to the current state of the implementation/architecture.

Method Selection and Planning

- Original deliverable (Plan1)
- Updated deliverable (Plan2)

In terms of the Software Engineering Methods, little change was made. Similar to the previous team, we continued using GitHub as our central repository, Whatsapp as our general-purpose communication channel, GitHub Pages for building our website, IntelliJ and Gradle as our IDE and game development framework, and Tiled to extend the map. The reason for this was mainly because we were already familiarised with these platforms from assessment 1, and no problems surfaced that would have prompted us to make any changes.

The only modification we have made in this section was switching from Slack to Discord for online meetings and implementation-related communication. This is because we already used Discord for the first part of the assessment, and we felt that it was easy to use and had all the functionalities that we needed, so there was no need to move to another platform.

Regarding Team Organization, at least one weekly meeting was made during the Wednesday practical, and additional meetings were arranged depending on the circumstances of each week.

In contrast to the other team, and also our organisation in assessment 1, instead of allocating work based on individual strengths, this time we prioritised the preferences of each member, so we could explore and investigate more on areas of our interest.

Lastly, in terms of planning, we changed the format of the Gantt Chart by adding the work distribution so it is clearer and easier to understand.

Risk Assessment and Mitigation

- Original Deliverable RISK1
- Updated Deliverable Risk2

We continued with the previous group's risk assessment strategy as well as the format of their risk register. Our first course of action was to address the ownership of each risk, as this isn't a very risky project we decided that the owner of each risk didn't need to be an expert on the risk itself; further to this many of the group members tasks spanned multiple aspects of the work so it made sense to attribute the ownership of all the risks to everyone.

It was clear a review of the risk register was necessary, with risks such as 'requirements being vague' and 'underestimating the complexity of the software being developed' not being relevant at this stage of the project; both of these risks being ironed out as any queries surrounding the requirements and/or the problem at hand can easily be answered through further research or simply setting up a meeting with the customer.

Finally we felt some risks needed breaking up further. Risks such as 'Not meeting the stakeholders requirements' needed dividing up as failing to meet the stated requirements within the deadline is one thing and misunderstanding the requirements and implementing the wrong thing is a different thing entirely. Removing ambiguity from our risk descriptions certainly helped us capture a clear idea of what risks we may face and allowed us to move through the project with them in mind to mitigate their potential impact.

References

[1] I. Sommerville, *Software Engineering*. 10th ed. Essex: Pearsons Education Ltd, 2016.