ENG1 Assessment 2 Report

Cohort 3, Group 23

Deliverable #2: Change Report

Oliver Dixon
od641@york.ac.uk

George Cranton
gc1263@york.ac.uk

Praj Dethekar
pd910@york.ac.uk

Denys Sova
ds1855@york.ac.uk

Shivan Ramharry
sr1907@york.ac.uk

Albara Shoukri
aams508@york.ac.uk

Rafael Duarte
rd1395@york.ac.uk

Department of Computer Science
University of York

Semester 2, 2023/24

Overview of Change-Management Processes

Throughout the Brownfield Development stage, Team 23 used the VCS-integrated *GitHub Issues* functionality to plan and review changes. The similarly integrated *GitHub Pull Requests* and *GitHub Actions* feature-sets were used to prototype and integrate changes being managed under Issues, respectively. Alternative approaches were considered, such as using a custom system of typeset (LaTeX) change-report cards, or a Google Docs spreadsheet, but the employment of GitHub Issues was the unanimous decision of the Team due to its pre-existing nature as an integral component of the Team's workflow. The automatically available ability to *tag* GitHub Issues under a set of categories, and easily assign existing collaborators to handle each particular issue systematically, has been shown to increase the general productivity of a team while ensuring greater adherence to the project requirements [1].

When considering our approach to change-management; in particular the handling of the changing customer requirements, the Team extensively researched proven methods to align with the established agile approach [2]; the processes stated here represent the outcomes of the investigations and Team meetings.

An example GitHub Issue and corresponding Pull Request (annotated with the result of a GitHub Action executed on the build-reference server) can be seen here:
https://github.com/ENG1-Group-23/A2-implementation/issues/10;
https://github.com/ENG1-Group-23/A2-implementation/pull/13

Register of Changes

For each updated Assessment 1 deliverable, links to the original (Team 25) and updated (Team 23) versions are provided. The updated versions highlight the added prose with a light-orange backing.

Requirements

Original deliverable: https://eng1-group-23.github.io/A2-website/Req1.pdf
Modified deliverable: https://eng1-group-23.github.io/A2-website/Req1-modified.pdf

User Requirements

For user requirements, we decided to make only four major additions:

1. The `UR_NAME` requirement, which gave the player's character a name. We implemented this by taking the username on the player's device and using it for the character. The main reason for this addition was that we needed a way to discriminate each score entry in the leaderboard. We could have used a random number as an identifier but we thought it would make more sense, given the context of the game, to use a human-readable name.

2. The `UR_LEADERBOARD` requirement allows the player to see a leaderboard of the top-ten scores with their respective character names at the end of the game. (High score data persist across game invocations.)

3. The `UR_ACHIEVEMENTS` requirement which gives players the ability to get different achievements by doing specific hidden tasks throughout the game's duration. This was added due to the changes made to the product brief by the customer.

4. The `UR_PAUSE` requirement which gives players the ability to pause while playing the game. We believe that this is a crucial requirement because it creates a significantly smoother playing experience. Pausing allows the player to go away from their device for any reason and for any amount of time, without suffering in-game consequences.

Also, a requirement was deleted:

1. The `UR_TIME_SKIP_ANIMATIONS` requirement was removed, as it was deemed to require too much effort implementing comparative to the payoff we'd get.

Finally, 3 minor changes were made:

1. The `UR_MUSIC` requirement was renamed to `UR_AUDIO`, to encapsulate both sound effects and music. Description changed to reflect this
2. The `UR_STUDY_GAME` requirement was renamed to `UR_MINI_GAME`, to encapsulate the new minigame we had added. Description changed to reflect this.
3. The `UR_OBJECTIVE` requirement's description was slightly changed to remove it being visible on a starting screen, as it was implemented in a tutorial menu instead.

Functional Requirements

In the functional requirements, we made six major additions:

1. The `FR_SCORING` requirement, which increments the player score by a base value of 15 after each day. A multiplier is used on the score to reward or punish the player depending on the amount of times an activity is done. While having a numerical score is part of the product brief, the manner in which we decided to handle the implementation was a group-wide decision. We chose to make the scoring system work in a way that would reward the player for studying multiple times a day but punish them for not studying for multiple days in a row. Doing it this way accurately reflects the real world in terms of the fact that if you don't study you won't do well and also keeps up the story of the game.

2. The `FR_STUDY_MULTIPLIER`, which is the multiplier used on the score once the day is finished. The study multiplier value goes up by 0.5 each time the player studies in one day and is also increased based on the number of hours spent in each study session. We chose to implement it this way because it rewards the player in a way that the more the player studies, the higher their score will be instead of it being a flat increase that the player receives for studying at least once a day. This adds to the complexity of the game as the player now needs to make a decision on how much they would like to study in a day, while also ensuring leaving enough energy to be able to do recreational activities.

3. The `FR_REC_MULTIPLIER`, which is another multiplier that is applied on the score once the day ends. This multiplier works the same way as the study multiplier but for recreational activities. The main difference is that we made this multiplier have a lower effect on the score compared to the study multiplier as we thought it would make more sense to reward the player for studying more rather than spending most of their time doing recreational activities. This made more sense to us story-wise and also gameplay-wise.

4. The `FR_LEADERBOARD` represents a leaderboard table that is visible to the player at the end of the game and shows the top 10 scores with the names of the characters that achieved those scores. We added this to the requirements due to the updated customer specifications

5. The `FR_ACHIEVEMENTS` which are achievements that the player can obtain and hence gain additional points. We decided to make the achievements hidden to the player as we felt like it would be a nice surprise for them to see at the end of the game which is where they see which achievements they got. We also felt like having it as a part of HUD where the player can constantly see what the achievements are might clutter the screen and make the gameplay experience less enjoyable, especially on smaller screens (given the accessibility requirements implemented in Assessment 1).

6. The `FR_PAUSE_ACTION` which allows the player to pause the game while walking around campus. By clicking the pause button (Esc), the player is sent to the main menu. From there, the player can enter the settings menu to change any settings, or resume the game by clicking the play button. By making it send the player back to the main menu, we believe it makes the other menus more accessible, as it takes less effort to reach. It is also more intuitive for the player to understand, as most modern games work the same way, which in turn makes the gameplay experience more enjoyable.

Non-Functional Requirements

For the non-Functional Requirements, we didn't make any major changes as we felt like the requirements already there were able to fully encompass all aspects of our game and accurately measure and test whether our game works on different devices and how well our game runs on different devices, while continuing to complement the updated set of user and functional requirements.

However, one requirement was removed, as it was determined to be redundant:

1. The `NFR_COMPATIBILITY_SUPERCEDED` requirement was superseded by `NFR_COMPATIBILITY`, and thus was no longer needed.

Architecture

Original deliverable: https://eng1-group-23.github.io/A2-website/Arch1.pdf
Modified deliverable: https://eng1-group-23.github.io/A2-website/Arch1-modified.pdf

The following changes were made to the *Architecture* deliverable; each change is justified by a link to the specified requirements (c.f. the *Requirements* deliverable). The updated architectural diagram, evolution of the system architecture, and various design justifications can be found on **Page 3**, **Section 1.3**, and **Section 1.4** of the updated *Architecture* deliverable, respectively.

Other than the added classes listed below, no changes were made to the exposed interfaces of the system architecture. Insignificant changes were made to the implementation of existing classes, such adding behaviour-tracking functionality to private methods and attributes in the `MainGameScreen`, but the interfaces remained unchanged notwithstanding the updated customer requirements.

1. The `Leaderboard` class was added to the `utils` package due to the `UR_LEADERBOARD` and `FR_LEADERBOARD` requirements, both of which were added due to the updated customer specifications provided for the purposes of Assessment 2. This class is self-contained and modular, exposing the minimal set of methods to the package (usable by the `EndScreen` to add a new entry to the leaderboard and commit the changes to the filesystem, or to render the leaderboard as a single LibGDX `Table`).

2. The `Score` class was added to the `utils` package due to the `FR_SCORING`, which was added due to the updated customer specifications. This class also exposes a very compact interface, enabling the game manager (principally `MainGameScreen`) to register certain score-influencing player actions, and for the `EndScreen` to compute and retrieve the final score.

3. The `Achievements` class was added to the `utils` package due to the new `UR_ACHIEVEMENTS` and `FR_ACHIEVEMENTS` user and functional requirements, both of which were introduced due to the updated customer specifications. This class is used to track the general behaviour of the character throughout the playing session and determine whether the player is eligible for any score-boosting achievements and/or “streaks”, where the particulars of these achievements and streaks are defined to be consistent with those stipulated in the updated customer specification, and reiterated in the updated *Requirements* deliverable.

4. A `TappingGame` mini-game was added alongside the existing `TypingGame` to enhance the `UR_STUDY_GAME` user requirement, and also to strengthen the implementation of the SSON (single statement of need), which emphasises the realistic nature of the activities, and the variance thereof, taken by the character in the week preceding exams.

5. To further enhance the realisation of the `UR_STUDY_GAME` and SSON, a `FeedDucks` class was added with a supporting `Duck` entity.

Method Selection and Planning

Original deliverable: https://eng1-group-23.github.io/A2-website/Plan1.pdf
Modified deliverable: https://eng1-group-23.github.io/A2-website/Plan1-modified.pdf

The following changes were made to each section/subsection of the *Method Selection and Planning* deliverable:

- No changes were made to **section 1.1** (Software Engineering Methods) as the methodology chosen by the prior team aligns with how we have conducted our team as well, and **section 3.1** (Systematic Plan for the Project: Key Tasks and Dates) as it dictates the previous teams plan up till the first submission, therefore we decided to not add to that section but to create section 3.2, to distinguish between the different team plans.
- **Section 1.2** (Development and Collaboration Tools) was changed slightly. The local management of version control used by the prior team was the GitHub Desktop program, which our team lacked experience with, therefore we decided to use integrated IntelliJ Git features instead, due to us having a lot more experience with it since it was used during our first assessment.
- **Section 2.1** (Approach to Team Organisation) was altered to include the new team, and its team members in their designated roles, and gave slight detail on how the development team and other deliverables were conducted.

- **Section 3.2** (Gantt chart for Assessment 2 tasks) is a newly added section which details the entire workflow and team member assignments for our team. Starting from when it was handed over to us to the submission due date. We made this a new section instead of continuing the prior section 3.1 (Systematic Plan for the Project: Key Tasks and Dates) so that we can differentiate between teams and our different workflow and tasks.

Risk Assessment and Mitigation

Original deliverable: https://eng1-group-23.github.io/A2-website/Risk1.pdf
Modified deliverable: https://eng1-group-23.github.io/A2-website/Risk1-modified.pdf

The following changes were made to each section/subsection of the *Risk Assessment and Mitigation* deliverable:

- No changes were made to **sections 1.1** (Risk Assessment Process) or **1.2** (the format of the Risk Register), as the risk elicitation process was well-justified and applicable, and the Risk Register was formatted in a way that did not discount any key areas of risk categorisation.

- **Section 2.1** (the Risk Register) was changed extensively, with the replacement of the owners of every risk, change in degrees of severity and likelihood for certain risks, and the rewording of the mitigations for certain risks.

    - After the project was inherited, looking through the Risk Register, it was clear to us that certain risks were no longer fully applicable, and that all of the owners of the risks would have to be changed to members of our group. For this purpose, we initially conducted a set of group meetings to go through the risks together, ensuring that the entire team knew about the risks discovered by the previous team. We also brainstormed for any new risks and recorded our results systematically in a tabular format, as seen in the revised *Risk Assessment and Mitigation* deliverable.

    - We assigned a group member to go through all the risks, adjust the likelihood, severity and mitigations of certain risks, and change the owners of all risks to be members within our group, ensuring that risks were still being monitored and had adequate measures to substantially reduce the chances of any severe risks occurring.

    - Most risks were not significantly changed: **R8**, **R11**, **R14**, **R15**, **R16**, and **R19** had their severity or likelihood changed by only one degree, due to the different priorities of our team. The language of the mitigations of certain risks was also changed for clarity, namely in **R1**, **R8**, **R9**, **R10**, and **R11**. Finally, the owners of every risk were changed to suitable members for risk monitoring, with ‘All’ as the owner where one member alone seemed insufficient.

- In addition, two new risks were identified and added: **R21** and **R22**, due to the new responsibilities incurred by the modified customer requirements (updated for the purposes of Assessment 2).

References

[1] J. Manuel Pérez-Verdejo, Á. J. Sánchez-García, J. O. Ocharán-Hernández, E. Mezura-Montes, and K. Cortés-Verdín, “Requirements and GitHub Issues: An Automated Approach for Quality Requirements Classification,” *Programming and computer software*, vol. 47, no. 8, pp. 704–721, Dec. 2021, doi: https://doi.org/10.1134/s0361768821080193.

[2] S. Jayatilleke and R. Lai, “A systematic review of requirements change management,” *Information and Software Technology*, vol. 93, no. 2, pp. 163–185, Jan. 2018, doi: https://doi.org/10.1016/j.infsof.2017.09.004.