Module	Engineering1 (Eng1) - COM00019
Assessment Title	Assessment 2, Cohort 2
Team	Dragon Boat Z (Team 18)
Members	Robert Dalgleish, Benjamin Jenner, Joseph Lonsdale, Richard Upton, James Wilkinson, Xinyi Zhang
Deliverable	Change Report

Formal approach to Change Management

Upon changing to a separate group's project, each team member was instructed to review the parts of the Assessment 1 project: the codebase, website and deliverables. This was so that each individual was educated on the new project, and could offer an opinion when discussing proposed and suggest changes. Changes will be discussed in the two regular meetings on Tuesday and Thursday.

The first step of Change Management are the deliverables, as many other tasks are expected to be dependent on the adjustments made in them. Regarding these, it was agreed that separate copies would be created for each deliverable to allow for changes to be made to a modified version of the original. According to research done individually, when making changes to the deliverables, each change will require a small explanation for its reasoning, and when making the actual changes a distinct colour would be used to highlight the relevant change, allowing for easier referencing back to the document in the future. To ensure that the changes made to the documentation are relevant and necessary (e.g. changes to requirements must capture the customer's original requirements and the scope of the project), before finalizing changes, an additional team member will perform a secondary check before the change is signed off, as this is good practice according to our research.

After reviewing the codebase, it was agreed that a standardized format to documentation and code would be required. It was apparent that irregularities lay within the code, such as the naming of variables and presence of JavaDocs. Our approach to documentation will involve working through the existing documentation, identifying areas where there is insufficient detail, and checking if all the code is covered in the JavaDocs, including newly implemented features. This will begin on a weekly basis after completing the debugging of the original codebase. With regards to commenting, a standardized format will be created and adhered to when commenting, which will involve method, class and in-line comments.

To make sure that changes made were consistent and relevant, GitHub's VCS features were utilized. Separate branches were constructed that required a review from an additional team member before pulling to the main branch, as is good practice. This check will involve consistent coding style and working functionality. The previous group also used a branching style, in which branches were assigned to team members and were dedicated to groups of features. This was altered to better reflect our incremental changes to stages of development.

Relevant URLs used for research:

- I. Sommerville, *Software Engineering, Global Edition*. Harlow, UNITED KINGDOM: Pearson Education, Limited, 2015. [Accessed: 16 Jan. 2021]
- Atlassian, *[No title]*. [Online]. Available: https://www.atlassian.com/itsm/change-management/best-practices. [Accessed: 16 Jan. 2021].
- The SunView Team, *Four Ways to Improve IT Change Management Best Practices*. [Online]. Available: https://www.sunviewsoftware.com/blog/four-ways-to-improve-it-change-management-best-practices. [Accessed: 16 Jan. 2021].
- *Change Management in Software Development Projects*. [Online]. Available: https://adevait.com/software/change-management-software-development-projects. [Accessed: 16 Jan. 2021].

Requirements

Link to Original Requirements:https://github.com/Dragon-Boat-Z/Assessment2/blob/website/docs/assets/assessment2/deliverables/Req1.pdf
Link to Updated Requirements:https://github.com/Dragon-Boat-Z/Assessment2/blob/website/docs/assets/assessment2/deliverables/Req2.pdf

For changes made, highlighting was used to indicate new requirements (in new doc.), modified requirements (in new doc.) and removed requirements (in old doc.).

Upon analysis of the Requirements document, we came to the conclusion that some requirements could be removed, as they were either no longer necessary or never necessary to the customer in the first place. There were also some that needed to be added, based on our own meetings with the customer and the new requirements specification we had been given.

Our first step after discussing the new features with the customer, was to write up the requirements for these. Some unnecessary requirements were removed and others were modified to make the document clearer and more concise. The full list of changes made to the document are detailed below with their justifications.

Added requirements:
These requirements were added for the new features requested by the customer:

- Added new User requirements 1.2.4 - 1.2.7, 1.3.1 - 1.3.3
- Added new Functional requirements 20.0.11 - 20.0.15

These requirements were added to further clarify what was needed from the game, as we felt that the existing requirements were lacking in detail. This was done to help meet the customer requirements and to make designing tests easier:

- Added User requirements: 1.0.3, 1.3.4 - 1.3.7
- Added Functional requirements: 2.0.16 - 20.0.23
- Added Non-functional requirements: 3.0.2 - 3.0.5

Removed requirements:
These requirements were removed from the document. They detailed features for changing graphics, audio settings and controls, but these features were neither requested by the customer nor implemented in the game. Therefore, the decision to remove them was made in order to prioritise features requested by the customer.

- Removed User requirement UR_CHANGE_SETTINGS 1.2.4, UR_CHANGE_RESOLUTION 1.2.2

Other changes:

The restraints for the original document were detailed in a short paragraph. It was decided this should be updated to a table to better fit the document format, and was extended to reduce ambiguity:

- Replaced constraints

These changes were made in order to fit the page limits. Any removed information was either detailed in a different way or considered to be redundant.

- Removed the test column from all tables - the data contained in these was used for creating the BlackBox Testing document. **LINK TO BLACK BOX TESTING.**
- Removed the subsection titles from User Requirements.

- Removed the priority column from User Requirements and instead colour coded rows. This is described in the introduction.

Abstract and Concrete Architecture

Link to Original Architecture:https://github.com/Dragon-Boat-Z/Assessment2/blob/website/docs/assets/assessment2/deliverables/Arch1.pdf
Link to Updated Architecture:https://github.com/Dragon-Boat-Z/Assessment2/blob/website/docs/assets/assessment2/deliverables/Arch2.pdf

Highlighting was used to indicate changes made to the Architecture.

Changes made to the Abstract Architecture

The original team's Abstract Architecture consisted of just a UML Activity Diagram; the only UML Class Diagram was in the Concrete Architecture. So, to complete the deliverable, a Class Relationship Diagram was developed for the Abstract Architecture: this diagram was needed to show the relationship between the classes in the game without going into the details of how the classes function internally. This allowed changing requirements, such as adding new classes and placing them in the game, to be planned, considered, designed and implemented more easily than if we had done this using Concrete Architecture.

The other changes to the Abstract Architecture stemmed from the new requirements given. There were two types of changes made: new classes added, and new features added.

The **new classes** added to the game affected our UML Class Relationship Diagram.

- A class "PowerUp" (requirement UR_POWER-UPS 1.3.3) was implemented to be a child class of Obstacle - the reason being that it would be easier to add PowerUps to the pre-existing Lane class if PowerUps could be treated as Obstacles. PowerUps have a method isPowerUp() which is true in PowerUp but false in Obstacle.
- "PowerUp" would itself be a parent class to the 5 different types of power-ups included in the game: "PowerUpBomb", "PowerUpHealth", "PowerInvulnerability", "PowerUpSpeed", and "PowerUpStamina". These classes inherit everything from PowerUp, but override the key "applyPowerUp(Boat)" method so that their effects would be unique.

The **new features** added to the game affected our UML Activity Diagram.

- The ability to pause the game (requirements UR_PAUSE_GAME 1.2.4, FR_PAUSE 2.0.12) required a new PauseGame state and PauseUI class
- Saving the game (requirements UR_SAVE_GAME 1.2.5, FR_SAVE_SLOTS 2.0.11) required a SaveGame state. Loading a saved game required a LoadGame state. Both saving and loading use the same SaveUI class to display the save slot options to the user.
- Different difficulty levels (requirement UR_DIFFICULTY_LEVELS 1.3.2) required a Difficulty variable that would be used throughout the codebase, for example in "Lane" where it would contribute to the number of Obstacles.

Changes made to the Concrete Architecture

All design changes made in the Abstract Architecture are reflected in the updated Concrete Architecture.

The PowerUp class and its five child classes were all added to the "Core" package, whereas the UI classes required to implement the pausing, saving and loading features were added to the "UI"

package.

The game states relevant to these new UI features were added to the “GameData” class, where the other game state data was already located, allowing them to be implemented in a similar manner to the original game states.

The Difficulty variable was also implemented in “GameData”, but the Difficulty selection was implemented in “ChoosingUI” as the choice the user makes after choosing a Boat to play as.

Other changes made to the Concrete Architecture stemmed from inconsistencies in the original implementation. For example:

- Stamina needed implementing in the “AI” class, not just in the “Player” class, as without this change, the AI boats had an unfair advantage.
- More Boats needed adding to the game so that there would still be 4 in the final, but less Boats in the final than in other legs (requirements UR_ MIN _ BOATS 1.0.3, UR_FINAL_PLACE 1.3.4, UR_FINAL_RACE 1.3.5).

Methods Selection and Planning

Link to Original Requirements:https://github.com/Dragon-Boat-Z/Assessment2/blob/website/docs/assets/assessment2/deliverables/Plan1.pdf
Link to Updated Requirements:https://github.com/Dragon-Boat-Z/Assessment2/blob/website/docs/assets/assessment2/deliverables/Plan2.pdf

The document was re-organised to clearly distinguish separations in the document layout.

Software Engineering Methods

As we are using the same methodology, Scrum, that was used in Assessment 1, no changes or further justifications were made to this choice as it was correctly justified and raised the key reasons that made Scrum suitable for this project’s environment.

Development and Collaboration Tools

For additions made to this section please refer to text highlighted in *colour*, in the Updated document. For text removed refer to text highlighted in colour in the Original document.

Zoom was removed as a collaboration tool because it did not provide the feature-set of Discord, which we started to use in Zoom’s place. Discord allows us to already voice call, which Zoom also offers, except it can be easily extended into voice channels specific to the different areas of work in the project. To evidence this, the justification of Discord was extended.

Additional evidence of Google Drive and Git were provided to further justify their use over similar tools, as we felt the existing evidence did not capture the key features that were necessary and relevant to this project.

As Assessment 2 requires White Box testing during the development and of the final game, GitHub’s issue system was added as a development tool, as this was not used previously in this context. GitHub Issues, in Assessment 1, was used for the management of Sprint tasks; however,

this was removed and instead solely used for the management of issues in the codebase and when WhiteBox testing. GitHub Issues did not provide the necessary feature set to manage the additional key tasks related to Deliverable and Implementation work. Instead, Jira was used as we knew this would provide the feature set we needed to clearly set out key tasks and stories. Furthermore, having all had experience with Jira in the first Assessment, the Road-Map feature was used to provide a map that would reflect the progress of the project, closely related to the initial Gantt Chart.

Further tools were added to evidence the entire tool selection:

- The tools used to adjust the existing Concrete and Abstract architecture was added. The tooling decided upon was Lucidchart as this could support the synchronous work that will occur in the team
- Java was stated as a development tool used to reflect the Customer Requirements for the project
- The IDE's used were specified as it reflected the consistent approach used by the team to the project. These were Visual Studio Code and IntelliJ.
- JUnit was an additional open source framework used, which allowed the existing codebase to be White Box tested, and therefore provided an environment where new features can be written.

.

Team Organization

For changes made to this section please refer to text highlighted in *colour*.

Few changes were made as the general outlined process is a sensible approach to the planning of this project. It clearly showed the need for detailing regular consistent meetings each sprint and clearly reasoned the importance of meetings.

- An addition was made to further elaborate on key roles which are required when using the Scrum methodology. This described the role allocations within our team for this Assessment. It was decided that the roles would be assigned to the same team members as in Assessment 1. Specific Sub-Teams were also specified as this would help to manage and further guarantee completion of the vaster workload.
- An additional clarification was made to evidence how tasks and their priority would be assigned during Sprint meetings through the use of the initial Gantt Chart. This was to ensure that a consistent approach was used throughout Sprint planning sessions.

Project Plan

For changes made to this section please refer to text highlighted in *colour*.

The Project Plan structure was changed because the original document did not explicitly show how Work Packages were identified or the key tasks required for those Work Packages, nor did it clearly identify tasks or priorities that would allow a clear critical path to be established.

In the Assessment 1 Gantt Chart, only Requirements had been used to identify Implementation tasks that should be completed during Sprints. This approach appeared sound, but as the initial project plan was constructed prior to eliciting further requirements, we decided to elicit tasks based not only on Requirements but also on other aspects of the implementation, so we could quickly

adapt to changes in the Project development and unexpected risks. By basing Implementation exclusively on specific requirements, this would not allow for the flexibility required in this Scrum Methodology, and would not capture other aspects of Implementation dependencies, such as testing and Sprite development. Instead, the tasks were defined as higher level milestones, such as ‘Implement missing features’ and ‘Prototype development’.

The Gantt Chart was extended to include the planning for Assessment 2, by identifying the dependencies that would link Assessment 1 to Assessment 2. We decided to be more concise with task priorities on the Gantt Chart by using one of three colours to assign priorities to tasks during Sprints. This helped to identify the Critical Path that was missing in the initial Gantt Chart structure for Assessment 1.

Instead of using a new Gantt Chart every two sprints to reflect progression, this was replaced with the Jira Roadmap functionality (an iterable Gantt Chart that reflected tasks completed in previous sprints and tasks that needed to be completed). This was done so the initial project plan was not altered and it enabled us to keep track of Sprints and overall progress within one place.

Risk Assessment and Mitigation

Link to Original Risk Assessment and Mitigation:https://github.com/Dragon-Boat-Z/Assessment2/blob/website/docs/assets/assessment2/deliverables/Risk1.pdf
Link to Updated Risk Assessment and Mitigation:https://github.com/Dragon-Boat-Z/Assessment2/blob/website/docs/assets/assessment2/deliverables/Risk2.pdf

As the final scope of the project was adjusted inline with the brief for Assessment 2, additional risks would be required to reflect this.

Changes to Risk Format

We adopted a similar process stated in the original, but defined this as the Risk Management Process, as this provides a concrete way for us to re-analyse existing Risks and further develop the existing risks iteratively. Further clarification was added for what information each risk must state and how we will use the process to continually review and manage risks.These changes to the Risk Management Process (highlighted in the Update document) replaced highlighted text in the Original document.

Few changes were made to the formatting of the risks as they are presented in a logical order. However, the risks were initially placed into a single table. When distinguishing amongst Project, Product and Business Risks, we anticipated that confusion may arise when regularly reviewing risks, so these were separated into their distinct categories in the Updated Risk Assessment (highlighted). The last change to the structure involved adding an additional column ‘Rank’; Rank enabled us to prioritize certain risks over others during the monitoring for risk development, and if several risks come to light, more impactful risks can then be prioritized.

The number of people assigned to a risk was heavily reduced. We didn’t see the benefit in assigning everyone to the same risk on multiple occasions as this can lead to confusion and team members not actively monitoring their assigned risks. In some places, all team members remained assigned to some risk because its scope was large, however in most risks this was reduced to two

team members. These changes to Owner are highlighted in the original document and also highlighted in the updated document.

Changes to Risks

A number of changes to the risks were made. As a team, we felt that some risks seemed trivial, irrelevant or could be grouped under a single risk that applies to all. For example, 'UI fails to react to the events in the game in a timely manner' or 'Player controls don't work as expected' are already defined in the Requirements deliverable and they are tested for in BlackBox and WhiteBox testing. These Risks were removed. These changes are highlighted in the Original document.

- Risk 7 was removed as this was not a requirement the customer had identified and was therefore not relevant to the project's outcome and would not impact its delivery
- Risk 10, 11 ,12, 13, 14, 15, 16, 17 were removed and condensed into one overarching Risk 15 within the Update deliverable: 'Features requested by the stakeholder are not implemented'.

Furthermore, we felt that a number of critical risks had not been noticed in the original deliverable. Covering these unnoticed areas was important, as each risk is given a mitigation enabling the team to quickly respond to risks throughout the project. The new risks added were highlighted in these colours based on the type of risk they are:

- Project Risks
- Product Risks
- Business Risks

Project Risks:

- Risk 2 was expanded to include situations where a team member's current situation changes unpredictably, for example due to Covid, or they are unavailable for periods of time during the Christmas holidays. Therefore, this risk's mitigation strategy was extended to be a more viable strategy that would allow us, as a team, to quickly recover and reallocate tasks deemed more important. To reflect this change in Mitigation, the Severity of this risk was deemed to be Medium, not High.
- Risk 7 & 10 were added as the risks of 'longer task length' and the 'impact of not completing a task' were not initially recognized. Both risks were assigned a High likelihood, to reflect the short Project timeframe, but Medium severity, because the mitigation plan allowed us to quickly adapt and reassign priority amongst the tasks to be completed. The mitigations of these risks involve frequently monitoring tasks and being adaptable by focusing on or dropping tasks that are falling behind.
- Risk 12 was added to raise the risk of poor project planning. We realised this risk during the second sprint session (14th - 20th Dec) when we analysed the existing codebase. It was assigned a Medium Severity and Likelihood.
- As testing was a new element to the project's timeline that we did not have experience in, we were unsure of the time requirements and how time consuming it may be to conduct BlackBox and WhiteBox testing throughout the Assessment. Therefore, Risk 11 was added and assigned a high severity because testing is important to guaranteeing a final product usability. We agreed that a sensible mitigation would be to focus on WhiteBox Testing as this would at least help us to guarantee the core functionality of the game.

Product Risks

- Risk 5 was adjusted to a Low likelihood because we have all gathered experience working with the LibGDX and know that detailed documentation exists. Its mitigation remains the same as the risk itself has not changed, just its likelihood.
- Risk 15 was added because new requirements would be given at the start of the Assessment. Therefore, this risk was assigned a medium likelihood to reflect the Assessment being released during the Christmas period. As we could not maintain communication with our Customer during this period, if we were not confident in our understanding of these new requirements, this could impact the final product. Therefore, it was also assigned a High Severity. Regarding mitigation, we felt that early and frequent communication with the stakeholder was the only appropriate strategy.
- Risk 13 was added because if LibGDX was no longer supported it could result in a number of unpredictable situations out of our control, such as much-needed documentation being missing or bugs being present, to occur in the codebase. We felt that the only mitigation that could prevent this risk heavily impacting the final product, would be to change to the next best alternative framework.
- Risk 16 was added because although we were confident in each other's programming abilities, we would be implementing different features which require different functionality. The likelihood was set to Low because we felt confident in our abilities, however, the severity was set to Medium because it could lead to a requirement not being implemented in the final version. We believe our mitigation is justified so long as team members are honest about their capabilities.
- Risk 14 was partially covered in the Project table adjustments, Risk 11, but BlackBox and WhiteBox testing raises an additional risk if not conducted properly. The final product, if not tested properly, could cause the game to be inadequate, based on the customer requirements. As both tests help to highlight small errors and bugs the potential impact could be severe thus it was assigned a High severity. Our mitigation is appropriate considering testing starts early in our project plan.

Business Risks

- Risk 17 reflected the risk with taking on board a new project, with a medium Likelihood and High severity. By inheriting a new project, the risk of the codebase not being modular enough to accommodate the new requirements could impact additional work done on the game beyond the scope of this project. To mitigate this potential risk, it would be important to spend sufficient time understanding the current limitations of code dependency, and then plan additional time that would be needed to overcome these.
- Risk 19 was added to reflect the impact of an incomplete game if it is to be sold or used in demonstrations. This could cause unneeded hassle for the business. The mitigation strategy was written to reflect how we could avoid missing functionality but provide the support necessary through a written up report, to help quickly implement these features in the future.