

# **Change Report**

## **Team CASEUS**

**(Team 7)**

**Eliot Sheehan**

**Matthew Turner**

**Daniel Atkinson**

**Hannah Pope**

**Mayan Lamont**

**Divyansh Pandey**

**Part (a)**

In order to ensure that we are making changes where necessary, we came up with our formal approach to change management. This was decided as a team and it covers change to all different aspects of the project, including documentation, deliverables and code. By formally stating our approach to change management, this means the whole team is clear on how to go about making changes and what steps they need to take so that the change can be successfully implemented and correctly documented.

The first step in this process is to identify where a change needs to be made and what the necessary change is. This will usually be done by a single team member and will then be discussed in the next meeting. The next step will be to have a group discussion about the proposed change, where we will address the need for this change, in addition to how it will be implemented. Once everyone in the team agrees with the proposed change, we will then assign people to implement it. Usually this will be the person that initially proposed the change, but if it requires a lot of work or is particularly difficult, we will assign multiple people to the task. The final step in this process is to review the implementation for the change. This will involve every team member reviewing and checking over all aspects of the change, in order to ensure it is correct and fully functional. Once this final step is completed, the change will have been successfully implemented and the team can move onto the next one.

By following these steps to manage change, we will be able to track the progress we are making each week in a much more efficient way, because everyone in the team will be aware of each change that is being made. Furthermore, by involving the whole group in the process, this means that people can give their opinions regarding how to implement a specific change to the project. This could be useful if someone has a specialised skill or is just familiar with implementing a problem of a similar type.

### **Why do we need to update/change the requirements?**

The team recognised that updating the requirements is a key step in auditing work done and providing a future pathway for the development of the game. The requirements tables are useful aids to implementing new and modifying old code. It allows an “audit” trail for new requirements that the team has agreed upon to add to the game and also allows us to check what requirements were successfully implemented by the original team. A key element of this stage of the development was a thorough review of the requirements. Using the requirements document it became evident that there were some inaccuracies i.e. missed recording requirements that had actually been implemented in Assessment 1, so we conducted a full review of requirement tables, adding these missing elements in addition to adding our new requirements.


### **What did we not change from Assessment 1?**


We decided to use the original team’s table templates, we thought this was important for continuity so that we could clearly show what we had amended vs their original work. Their use of the Risks and Assumptions column within some of the tables was a very useful way of keeping track of problems that may arise with each requirement. We believed that this could prove useful when considering and assessing the game's development.


### **How did we go about identifying changes/updates?**

The team audited the requirements document along with the code. This allowed us to identify what requirements had and had not been successfully implemented by the previous team, and in some cases which requirements that had been met but had not been recorded within the requirements tables. We decided to remove the introduction and the constraints table from the original document to allow us more space to update the tables. The constraints had remained the same between both Assessments so the information was redundant. We also decided to remove the Non-Functional Requirements table as we believed that there were no new ones to be added. This would give us the freedom to amend and add requirements that we deemed more important to this step of the project. We amended any requirements that had not been recorded by the original team in Assessment 1. And added our new requirements. Using the updated tables we should minimise the risk of duplicating any elements of the implementation delivered by the previous team. We believed this would add clarity for the team {Caseus} and any other additional people (potential customers) who would wish to view progress, and verify that we have met the requirements set. To aid clarity we colour coded the requirement tables using the following key to clearly show what we had updated within the tables, and allow us to readily identify what needed to be changed/updated within the code and in other deliverables.[1]

#### **Key:**

 = Requirements that were already implemented by the previous team during Assessment 1

 = Requirements that we added and implemented during Assessment 2

 = Requirements that were apart of Assessment 1 that were not properly implemented by the original team that we have finished implementing during Assessment 2

 = Amended Requirements that were implemented by the original team but were not in the original Requirements deliverable

### **What did we change?**

#### **User Requirements (UR):**

We added six new requirements that had not been implemented or recorded within assessment 1 these included the following: [1]

Requirements ID	Description	Risk and Assumptions	Priority
UR_DIFFICULTY_SELECTION	In the game menu the users can select the difficulty of their game	The user should have an idea of how difficult the game will be by using words like easy, normal, hard as to avoid them selecting overly difficult races	Must
UR_BOOST_ORBS	Each boat will be able to pick up set of boosts that can give the players boat special abilities including: immunity, speed, health, maneuverability, acceleration	The boost orbs should be balanced as to not make the game too easy or too difficult for the user	Should
UR_QUALITY_CHECK	Every step we take when we are developing the game we must make sure that it is at a good quality for our client and keeps up with the modern standards of game development	If we do not quality check it could result in us developing a poor game that doesn't work properly	Must
UR_HEALTH_RESET	The user will have their health reset every race	Assumes that the races will be balanced enough to damage them at least a bit	Should
UR_STAMINA_REGENERATION	The user will be able to gain stamina back by allowing their boats rowers to rest (not holding forward)	The stamina bar should not be too powerful or too weak it should be balanced	Should
UR_PAUSE_MENU	The users will be able to pause the game and edit the sound levels and be able to save their game	If the menu is overly complicated the user will find it difficult to navigate	Should

**Example of a requirement implemented but not documented in original requirement tables by the previous team: [1]**

UR_OBSTACLES	During each race the boats shall face obstacles that float down the river where if hit the boats shall take damage	Obstacles should be balanced as to avoid them doing too much damage or too little	Should
--------------	--	---	--------

Finally we updated all the priorities using the MoSCoW method ensuring a universal standard. [1]

#### **Functional Requirements (FR):**

We added six new requirements that we deemed important: [1]

ID	Description	Risks and Assumptions (if necessary)	User/Design Requirements ID
FR_OPTIONS	The game should have an options menu for the user to be able to edit the sound etc.	If the options menu isn't clear it can be too hard for the player to navigate it	UR_OPTIONS_MENU, UR_PLAYABLE
FR_PAUSE	The game should allow the user to pause during the race and allow the user to save	If the pause menu isn't clear it can be too hard for the player to navigate it	UR_PAUSE_MENU, UR_PLAYABLE
FR_SAVE	The game should allow the user to save during the game	Assumes there is a function that allows the user to save their progress within the game. Risk that the save file could end up being corrupted	UR_PAUSE_MENU, UR_PLAYABLE, UR_CONVENTIONS

FR_BOOST_ORBS	The game contains a series of orbs that allow the player to gain special abilities such as: immunity, speed, health, maneuverability, acceleration	The boost orbs should be balanced otherwise the game could become too easy or too difficult for the user	UR_PLAYABLE, UR_BOOST_ORBS
FR_HEALTH_REGEN	Health will be regenerated only after every race	Health regeneration should reset at the end of every race to help the user get to the final race	UR_PLAYABLE, UR_HEALTH_RESET
FR_DIFFICULTY_IMPLEMENTATION	The difficulty of the game will increase with each new race, by making the system controlled boats more competitive and increasing the number of obstacles. With the final race being at maximum difficulty	If the difficulty of implementation is too great, it can result in either the game being too hard or too easy meaning that the game will be not be enjoyable	UR_DIFFICULTY

**Example of an FR that had been implemented by the previous team but hadn't been recorded:**

[1]

FR_CONTROLS	The user will be able to control the boat using the "WASD" and arrow keys. They will also be able to speed up the boat using the "Shift" key.	If the controls aren't simple or linked to classic controls it could be too hard for players grasp and learn quick enough	UR_CONVENTIONS, UR_CONTROLS, UR_PLAYABLE
-------------	---	---	--

We finally updated the reference column of the FR table with our new UR's we had added to show which FR's were linked to them.

## **Abstract and concrete architecture**

### **Abstract Behavioural View**

Due to the change in requirements as a result of the additional features, we had to adapt the abstract behavioural view so that it fully represents the new changes.

The first change was to check whether the user has loaded a game from the menu screen. If the user does not load a game, the chart continues as normal. If the user does load a game, the chart will skip straight to the “Render Boats, Obstacles, Progress Bar” as there will be no more decisions necessary and the game content will need to be loaded. Next, we added a user decision with regards to the difficulty of the game, because the user must now select between three different difficulty options that are available on the menu screen.

The final change that we added is to facilitate the saving system; allowing the user to save their progress, exit the game, and then resume at a later time. In order to achieve this, we check whether the pause menu has been opened whilst the game is running. If it hasn't been opened, the chart continues like normal. If it has been opened, the chart then checks whether the player has chosen to save their progress. If they have, the game data is saved and the game exits. If they haven't, the chart then checks whether they have chosen to resume the game and if they have, it goes back to the main loop. If they haven't, the player must have chosen to exit the game and so the chart stops.

The only other additional requirement is to do with the power-up packs. This has not been represented in the chart as we plan to implement these packs as an “obstacle” and so the game will treat them the same way as with a log or goose obstacle. Therefore there was no need to add anything else to the chart.

### **Abstract Structure**

The only change that needed to be made to the abstract structure was the addition of the “Boost” class. This is implemented as a child of the “Obstacle” class and so as can be seen in the diagram, it inherits the methods from “Obstacle”.

### **UML Diagram:**

The UML diagram needed quite a few changes. Firstly, the classes: EndGameScreen, StartGame, Boost and IO needed to be added. These classes are used in some of the new features like saving/loading, pausing and power-ups. We then had to update the existing classes to reflect the changes we've made, and this included updating the relationship between them. Then we elected to make some modifications to the preferences of the previous team. For instance, they preferred not to include getters and setters and some unused inherited methods but we decided to include them for the sake of clarity, we also elected to include some types of attributes that don't really need to be on the diagram like BitmapFonts. Finally, we elected to omit the default values of some of the attributes as there was already enough information.

### **(iii) Methods and plans**

#### **Software Engineering Methods**

In the assessment 1 deliverable for methods and planning, the previous team decided on the Agile methodology with the Scrum framework as their chosen development methodology. We decided to continue using the Agile methodology for assessment 2 because we feel it is best suited to the size and timescale of the project and it is also something we are familiar with from the previous assessment. However we decided against using the Scrum framework because we have no experience using it and felt that trying to adapt to a new approach would reduce our productivity, since we would be spending more time trying to learn the new framework. Instead, we decided to use the Iterative development framework primarily because we are familiar with it after using it to complete the first assessment. This will consist of us quickly developing an initial prototype for the changes that we need to make, and then iteratively improving them until we reach the finished state of the project. Furthermore, we felt this framework suited the second assessment as the new features needed to be implemented in a specific order and iterative development allows us to do this as efficiently as possible.

#### **Development and Collaboration Tools**

In terms of the development and collaboration tools, we didn't have to make many changes as the previous team used a lot of the same tools that we used for assessment 1. For example, the previous team used Google Drive in order to create and manage the documentation and since we also used this for the first assessment, no change was necessary. The previous team also used Discord as one of their tools for communication and because we used this for the first assessment, we were happy to continue using it. For version control, we used GitHub for the first assessment and we were glad to see that the other team had done the same as this allowed us to easily clone their project and continue using the version control features that we are familiar with. Finally for the actual implementation of the game, the previous team used Java and the LibGDX Module which is the same as what we used. This was particularly useful because it meant we were familiar with a lot of the module specific functions from LibGDX and also helped us to quickly understand the code and get working on implementing the new features. The only tool that we changed was Jira as the previous team had used this in order to manage task assignment to fit their Sprint planning and since we decided against this, we felt it would be better if we assigned tasks in a way that we are more familiar with.

#### **Team Management**

Moving forward from assessment 1, we decided that we are going to split up tasks more evenly by allocating more team members to the same task. In the first assessment, tasks were a lot more individual and this resulted in the rest of the team being unfamiliar with the work that had been done. As a consequence of this, it became more difficult for new members to assist with a specific task and so progress was slower than it could have been. We firmly believe that by allocating more team members per task, we will be able to work more efficiently as we will all be more familiar with the various different tasks and so we will be able to assist where necessary. Tasks will still be assigned based on experience, meaning the team members who feel the most confident towards programming will be completing these tasks and the same goes for team members who feel more confident towards documentation.

## Assessment 2 Plan

Tasks	Week 1							Week 2							Week 3							Week 4							Week 5						
	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
Clone the Assessment 1 project																																			
Setup the new GitHub project																																			
Update the website																																			
Outline the changes to be made (unimplemented /new features and fixes)																																			
Formal approach to change management																																			
Change report: requirements																																			
Change report: architecture abstract																																			
Change report: architecture concrete																																			
Change report: methods and plans																																			
Change report: risk assessment																																			
Implementation																																			
Testing methods																																			
Testing																																			
Continuous Integration: methods																																			
Continuous Integration: infrastructure																																			

Above is the Gantt chart that we created to help us plan and manage the second assessment. The team is planning to meet every Tuesday and Thursday in order to discuss the progress that everyone has made and clear up any bigger issues. As can be seen on the left hand side, we split up the various deliverables that we have to produce into several tasks, therefore making it easier for us to manage how long each task should take to complete. The amount of time assigned to each task was based on the priority, as well as the estimated length it would take to complete. For example, some tasks are allocated a week to complete (such as Change report: requirements) and other tasks are allocated a single day, as we believe they can be completed by the whole team during one of our meetings.

Whilst this Gantt chart is different from the assessment 1 version for this project, we believe that it provides enough detail to clearly indicate how we plan to complete it. Furthermore, it is in the same format as the assessment 1 Gantt chart for our original project and so the whole team is familiar with this style of project plan. By following the timescales laid out in this plan, we are confident that the project will be completed to a high standard and on time.



#### (iv) Risk assessment and mitigation

The first change we made to the risk assessment was a re-review of all likelihoods, impacts and mitigations for each risk. For each risk we assessed each column of the table, taking into consideration our team and situation, before updating it with our changes.

We also considered any additional risks that we felt had not been covered in the risk assessment, and appended them. For example, the risk of a team member dropping out was not covered in the risk assessment, and therefore we added it. In addition to being a possible risk, it was also one that we experienced.

We also re-allocated risk monitoring roles to members of our team, allowing for an organised risk supervision system.

Lastly, we renamed the IDs to alphabetical representations to increase understandability (as before they were numbered - which means they could not be identified by their ID alone).

*Extract of risk assessment table with changes highlighted in yellow (full table is linked in the appendix) [2]*

#### **Project Risks**

ID	Description	Likelihood	Severity	Rank	Mitigation	Owner
PRJ_TASK_BAL	Tasks are not divided up evenly and certain members end up carrying the project	M	M	4	Assign each member specific pre-agreed tasks which are reasonably divided ensuring tasks with a large workload are shared.	Hannah
PRJ_TM_OUT	Team member stops communicating / drops out	L	H	3	Try to regain contact with team members. Alert module leaders if there is no response after a prolonged period of time. Be prepared to take on extra tasks and share uncompleted tasks with the remaining team.	Daniel
PJR_UPT_CODE	Sections of the code in the GitHub repository are not up to date resulting in inconsistent code.	L	M	2	Team members to commit and pull from GitHub frequently (at least daily). Create branches, if needed.	Matthew & Daniel
PJR_TEAM_CORD	Poor team coordination	L	M	2	Start holding more regular SCRUM stand-ups during Sprints. Set clear goals for each member to improve	Eliot

					coordination and productivity	
PRJ_DOC_DELETED	Documentation deleted	L	M	2	Ensure multiple up to date backups are stored so that documentation can be recovered	Hannah

### Product Risks

ID	Description	Likelihood	Severity	Rank	Mitigation	Owner
PRD_CODE_TST	The code is not subject to rigorous testing for bugs/issues leading to the submitted version being inadequate.	M	H	6	Create a time plan and review of adherence to schedule at weekly SCRUM meetings  Create a test plan to use during the testing phase. Ensure test classes are reliable and well developed	Mayan
PRD_CODE_FTRS	Lack of programming ability leads to features missing from the final product	L	H	3	Allocate coding tasks according to team ability Create a collection of relevant resources in case further skills need to be learnt	Mayan & Matthew

### Business Risks

ID	Description	Likelihood	Severity	Rank	Mitigation	Owner
BSN_POOR_CODE	The code is not iterable or able to be further developed due to poorly designed code and being undocumented.	L	H	3	Hold SCRUM meetings to review code for quality. Deobfuscate code before deployment  Ensure code is commented frequently for clear understandability	Eliot

## **Appendix**

### **Requirements section:**

#### **Full Requirement Tables**

[1]:Section: Assessment 2, Updated Assessment 1 Deliverables (Published 09-02-21), (Writer: Team Caseus) Online Link: <https://caseus7.github.io/Dragon-Boat-Z/docs/deliverables2/Updated%20Requirements.pdf>

### **Abstract and Concrete architecture Section:**

#### **Architecture Deliverable:**

Section: Assessment 2, Updated Assessment 1 Deliverables (Published 09-02-21), (Writer: Team Caseus) Online Link: <https://caseus7.github.io/Dragon-Boat-Z/docs/deliverables2/Arch1%20new.pdf>

#### **Architecture Diagrams:**

Section: Assessment 2, Architecture Diagrams (Published 09-02-21), (Writer: Team Caseus) Online Link: [https://caseus7.github.io/Dragon-Boat-Z/docs/deliverables2/Abstract\\_Behavioural\\_View.pdf](https://caseus7.github.io/Dragon-Boat-Z/docs/deliverables2/Abstract_Behavioural_View.pdf)

Section: Assessment 2, Architecture Diagrams (Published 09-02-21), (Writer: Team Caseus) Online Link: <https://caseus7.github.io/Dragon-Boat-Z/docs/deliverables2/Abstract%20Structure.pdf>

Section: Assessment 2, Architecture Diagrams (Published 09-02-21), (Writer: Team Caseus) Online Link: <https://caseus7.github.io/Dragon-Boat-Z/docs/deliverables2/UML%20Diagram.pdf>

### **Method and Planning Section:**

#### **Method and Planning Deliverable:**

Section: Assessment 2, Updated Assessment 1 Deliverables (Published 09-02-21), (Writer: Team Caseus) Online Link: <https://caseus7.github.io/Dragon-Boat-Z/docs/deliverables2/Plan1%20new.pdf>

### **Risk Management:**

#### **Full Risk Assessment Tables**

[2]:Section: Assessment 2, Updated Assessment 1 Deliverables (Published 09-02-21), (Writer: Team Caseus) Online Link: <https://caseus7.github.io/Dragon-Boat-Z/docs/deliverables2/Risk%20Assessment.pdf>

