Requirement

Team 14 - Taken On By Team 15

Joe Wrieden

Benji Garment

Marcin Mleczko

Kingsley Edore

Abir Rizwanullah

Sal Ahmed

Introduction

Preface

This document is intended to be read by the System Engineers as well as the stakeholder. The document is added to as new requirements arise (refer to version history for this document's various iterations).

1.1 Project

This is a requirement specification document for a single player game based on the Dragon Boat Challenge. The player will participate in the Dragon Boat Challenge race as one of six teams, where they will compete for the title of the York Dragon Race and receive a gold medal. There will be four legs in total, with the fastest team in each leg going through to the next leg. In the final leg, the first-place team will receive a gold medal, the second-place team will receive a silver medal and the third-place team will receive a bronze medal. There will also be obstacles that the user will have to avoid that will slow down the users speed and the robustness of their boat. 1.2 Development.

1.2 Development

As a team, after our initial research, we decided that our first main priority was to set out all of the requirements before developing the code and producing the project, as per Software Engineering practices. All further iterations of the artefact would stem from these requirements. This would ensure our staying aware of the stakeholders' and customers' goals, as well as guiding us through the rest of the development process - with the final product being of a greater quality than if we had not used the Requirements Engineering approaches we did.

After analysing the product brief, we set up a list of all the different possibilities we wanted to clear up with the customer, which we did in the team-customer meeting. From the product brief and initial meeting between the development team and the customer, we extracted User Requirements. From this, we defined our System Requirements, which consist of both Functional Requirements and Non-Functional Requirements. [1] We aimed to encapsulate the software in its entire scope, with all the requirements cumulatively describes the game in enough detail to meet all of the client's needs to keep our requirements as specific as possible, and not vague such that it would cause problems later on in the project. [2] We also made sure no requirements conflicted with one another, and that they were feasible, necessary, clear and verifiable, in order to ensure our requirements were of high standards. [3] This would ensure that the rest of the iterations of the product, which would be based upon this initial iteration, would also be of good quality.

2.1 Presentation

To distinguish between the types of requirements, we had a header which separated the User and System requirements as well as the functional and non-functional requirements. Each requirement has a unique ID, which will help us to identify and refer to different requirements, and a priority rating to help the Development Team understand which requirements are most important. We will also include any assumptions, risks and alternatives for each requirement (Mitigation). By doing it this way, the reflection process was made more efficient.

The following requirements are presented without technical jargon and in natural language, such that the customer can easily understand it. [4] They are written in a concise format so that confusion caused by ambiguity will be avoided, but have enough detail so the software developers have enough guidance. Each record in the tables represents an individual requirement, referred to by its ID. This format will be useful for when we later map our requirements to our architectures. A tabular approach was decided upon as it allowed for easy adaptation when changes are added and also allows for easier readability.

2 Requirements - Part B

2.2 User Requirements

Below is a table presenting the user requirements - what tasks the user should be able to carry out during the game process.

ID	Description	Priority	Assumptions, Risks and Alternatives(Notes)
UR_Main_Menu	The user should be presented with a main menu, where from the user can start or exit the game	Shall	Assumed that the boat choice area will be done automatically after pressing start The start and exit buttons should be clear and obvious to the user
UR_Option	The user can pick a type of boat from a predetermined selection following from the main menu screen	Shall	Boats must have unique specs, such as: speed, acceleration, weight, strength and handling
UR_Damage	The system shall allow obstacles to damage the users boat and slow it down proportionately	Shall	If the boat's health decreases too far, the boat will break and the game will end
UR_Records	The users will be able to see the time it took them to complete the current leg Users will also be able to see their current stats	Shall	Users might feel more excited when they take a short time to finish the leg/game. Users need to see why their boats may be behaving worse than before or they'd get annoyed at the game
UR_Race_Track	The users can navigate their racetrack, as well as neighbouring tracks, by steering left and right	Shall	Assumed that the boats might be damaged and slowed down when they collide boundary of the racetracks
UR_Endurance	The speed, acceleration and maneuverability of the user will decrease progressively during every leg - this can be recovered by not moving the boat left or right	Shall	Assumed that the stats for the users boat will be reset at each leg Obstacles will be needed to prevent the user keeping the boat in one direction and not having their stats decrease
UR_Difficulty_Level	The user will be able to choose the difficulty of the game from easy to normal to hard before they start the game. This option should be present to the user in the main menu screen.	Shall	Assume that the option to choose difficulty will be present as soon as the game is executed. Difficulty option chosen via "1- easy", "2-medium" and "3-hard".

UR_Save _Game	The user will be able to save the game anytime during the game via “F1”, “F2” and “F3” for each of the three save slots . They can reload the game in the main menu screen with the same buttons.	Shall	Assume that the option to save the game will be present when the game is running . The save should save all obstacles, the positions etc of the current race and the load button should present all of these again when the game is loaded up.
UR_Powe r_Ups	The user will be able to navigate their boat to pick up power ups which can be used to boost the player’s boat and help them win etc .	Shall	Assume that the power ups will all be different and all have different abilities such as a speed power up, health power up , stamina regain etc.
UR_Penal ty	If the users enter a neighbouring racetrack, they will receive a proportionate penalty	Shall	Risk that user may not know which is their lane and constantly receive a penalty - need to make it clear which lane is users
UR_Medal _ Screen	If the user has one of the top 3 times in the final leg, a medal screen will show with either a gold or silver or bronze medal Otherwise, a medal screen will show but with no medals	Shall	Alternatively the medal could be presented over the top of the gameplay screen

2.3 System Requirements - Functional Requirements

ID	Description	User Requirement
FR_Boat_Health	The system will keep a health variable for each boat that may have different values depending on the type of boat selected.	UR_Damage
FR_Damage	The system will reduce the health of the boat when they collide with an obstacle. When the health of the boat reaches 0, the boat will break .	UR_Damage
FR_Boat_Move	The system will allow the boats to move left or right to a avoid the damage from the traps by using the left and right arrow keys only	UR_Race_Trac k
FR_Boat_Slow ed _Down	The system shall reduce the velocity of the boats when they collide with any obstacles	UR_Race_Trac k UR_Damage
FR_Selection	Each boat type will have unique, predetermined stats that are set within the class	UR_Option
FR_Compare	The system must compare the users time with the generated times of the other boats	UR_Medal_ Screen

FR_Time_Stats	The system must keep track of the users time during the current leg. The system must keep track of the preset stats of each boat and then how they decline/increase throughout the game	UR_Records UR_Penalty UR_Endurane

2.4 System Requirements - Non-Functional Requirements

Non-Functional Requirements are qualities that a system must have

ID	Description	User requirements	Fit Criteria
NFR_ Respon se _Time	The system will be respond quickly to the user, i.e. when they use the arrow keys to move left and right	UR_Race_ Track	Respond in <milisecond
NFR_ Control	The system must allow the boat to be easily controlled by the user	UR_Race_ Track	Controls will be pointed out in the manual .

References

[1] “Requirements engineering — requirements specification (part 3)”, by Omar Elgabry, at https://medium.com/omarelgabrys-blog/requirements-engineering-elicitation-analysis-part-5-2dd9cffafae8

[2] “Software Engineering, 10th Global Edition”, by Ian Sommerville, page 102

[3] “Writing good requirements,” by Ivy F. Hooks, pages 1247-1253

[4] “Requirements Engineering: An Overview”, by Birgit Penzenstadler, at https://www.youtube.com/watch?v=qENBiYaAXNE