# Engineering 1
## Group Assessment 1
## Requirements Document

## Cohort 2 Team 12:

Umer Fakher

William Walton

Richard Erasmus Liiv

Olly Wortley

James Frost

Joe Cambridge

# Eliciting requirements

A brief statement was asked for that describes the idea/issue the customer was facing. “I would like to have a game developed and this would be a single player game and should capture all the excitement of these events: the Dragon Boat Race that happens in York every year". This, combined with the listed requirements in the product brief, gave us our initial requirements to start the process of eliciting requirements.

Upon reading requirements listed in the brief, it was decided that some of the features were vague and could be easily misinterpreted in the development. Thus, questions were created such as:
“What specific penalties would you want when you said *‘Every boat must remain in its lane for the duration of the race. Leaving the lane may result in a penalty at the discretion of the chief race official.’*?”
These allowed the requirements of the client to be understood in a more definitive way. For example, the answer to the question above was to add time to the players run timer, decreasing the player’s “score”.
This was important because the type of penalty was unclear and it would be easy to have gone with a wrong implementation.

During the meeting with the customer, the importance of some of the aspects of the game that weren’t explicitly stated in the brief, such as the general appearance and UI of the game, were asked about. This allowed the priority of certain features to be categorised, helping us to plan accordingly. For example, animations and graphics are a nice thing to have but were stated as something that can be added later, implying that they are of low importance and shouldn’t be worked on unless there is adequate time spare.

Enquiries were made into environmental constraints, like the type of hardware the game is expected to run on (normal computers) as well as the expected input system (standard mouse and keyboard input). This confirmed assumptions, ensuring the requirements were accurate, leading to a game closer to the clients wishes than otherwise achievable.

The requirements were formatted as a spreadsheet because it was agreed that this would be a clear way of communicating requirements between the team members and the stakeholders. The spreadsheet was split into 3 sections; User Requirements, Functional Requirements and Non-Functional Requirements. To make sure these were done to a high enough standard, research was done into user requirements engineering[1].

# Environmental Assumptions, Associated Risks and Alternatives

Clarification was needed for the SSON due to it being vague and lacking detail. The customer was asked if they could clarify how one would go about “capturing the excitement of the event” as mentioned verbally by the customer and stated in the Document Brief. Questions were asked if this was related to the game difficulty and user reward trade-offs. This will be referred to this as UR_EXCITEMENT.

The customer expanded that he wants an increasing level of difficulty and that he wants the first level to be accessible and easy to the player so the player learns the dynamics of the game. As you progress through the game and legs of the race, the difficulty would increase. This ended up being an example of requirement negotiation and this clarification with the customer helped to develop the user requirement UR_ACCESSIBILITY.

**UR_TIRED_OVER_TIME** - There was a bit of confusion about one of the key mechanics in the game - stamina - which prompted us to send an email to the customer asking for clarification. Shortly after, the team received an email back explaining how the feature should be implemented. The risk was wrongly assuming how a key game mechanic would function.

**UR_PLAYABILITY** - the team clarified with the customer that the user would be using a personal computer to play the game and owns a keyboard and mouse which are connected to the computer. The risk is that they do not have these, in which case they will not be able to play the game. There are no alternatives?

**UR_OBSTACLES** - One of the user requirements in the brief was to have obstacles in the race that the user has to avoid, so this will be included as a must-have requirement. It will be assumed that the game is working as it should and that the user is able to join a race. The risk is that the obstacles do not spawn or function properly. An alternative would be not to have them.

**UR_LANE_PENALTY** - It was vague what kind of penalty is given to the player for crossing their lane so this was clarified with the customer for a specific definition of “penalty”. The response was that extra time should be added to the players “score”. This being, the time it took them to complete the leg.

**UR_PERFORMANCE** - It was important to have a game that runs well and looks great on screen so a frame rate of 36fps + is the goal. It is assumed that the user is using a suitably up to date computer with a modern operating system which runs at a good speed. The risk is that the game does not perform smoothly. An alternative is having a game that runs at a slower frame rate on purpose.

**UR_INFO_DISPLAY** - It is important that the user sees information about their boat and the leg that they are currently racing so that they are well informed when they make control decisions e.g. paddling faster/slower, so this is defined as a should-have requirement. It is assumed that the user has a display monitor and that their game is working correctly. The risk is that the information does not display and the user cannot see it. An alternative is to use audio to tell them info such as remaining distance, stamina and speed.

**UR_COLLISIONS** - Again, this was one of the requirements set out in the project brief, so it is included as a must-have requirement. It is very important that the game is realistic, so it makes sense that crashing into objects should damage the boat.

**UR_AWARDS** - At the end of the race, the winner and good performers should receive a reward, and it was said in the brief that the 1st place boat should receive a gold medal, 2nd place silver, and 3rd place bronze. It is assumed that the user finishes the race and their game is working correctly, and they have all the necessary controls (keyboard and mouse). The risk is that the awards do not work or display. An alternative is just to display the times and order of the boats at the end of each race.

**UR_FINALS_PLACING** -This was mentioned in the brief, so it had to become a user requirement. It is also exactly how it works in the real York Dragon Boat Race, so this captures the real event very well. An assumption would be that the user has started a race and finishes all 3 legs, so that the top 3 boats can be picked to go into the final. The risk is that the code does not work and picks the wrong boats to go into the final. An alternative would be to have all boats from the first 3 legs in the final.

**UR_BOATS_NO** - There was uncertainty on how many boats should be in the race. This was for two reasons, the more boats in the race the harder each race would be, but also being able to see all the boats on your screen, without the size of the boats being too small to be practical. The customer was asked about this problem, who responded explaining he wanted ~3-6 boats in each race. Some other options were to increase the number of boats the harder the difficulty setting, since you have to beat more boats to win.

**UR_BOAT_SPECS** - The speed spec of the boat wasn't very well defined, it is assumed it meant the terminal velocity of the boat, In the project brief it asked for unique stats for each boat, there are a few ways of solving this, either creating several boats that the user would be able to pick from, or let the user create their own boat allowing them to change things like the size, paddlers.

**UR_LEGS** - Each race that the user does in the game should have 3 heat legs, and the fastest boats from these 3 races are picked to go into the final. This requirement was in the brief, so it was decided to set it as a must-have requirement. Assuming that the user's game is working and that they can enter a race. The risk is that the race does not have 3 legs for some reason. An alternative would be to have a different number of legs in each race?

**UR_EXCITEMENT** - The customer was very clear during the questioning that the game needed to be exciting and engaging for the player. This seemed to be a very important aspect needed for the game to be complete.

**SSON: A single-player boat racing game that captures the excitement of the annual York Dragon Boat Race**

| ID | Description | Priority |
|---|---|---|
| UR_PLAYABILITY | The game must be playable with a keyboard and mouse. | M |
| UR_BOAT_SPECS | Every boat must have a unique spec in terms of speed, acceleration, maneuverability, and robustness. | M |
| UR_TIRED_OVER_TIME | Over time, paddlers in the team get tired, so speed, acceleration and maneuverability decrease progressively during every leg. | M |
| UR_LANE_PENALTY | Every boat must remain in its lane for the duration of the race. Leaving the lane should result in a penalty at the discretion of the chief race official. | M |
| UR_OBSTACLES | Teams should find obstacles in the river during the race, like clueless ducks and geese, or tree branches floating down the river | M |
| UR_COLLISIONS | Colliding against obstacles will progressively reduce the robustness of the boat, until it breaks down (resulting in the end of the game). | M |
| UR_DIFFICULTY | Every subsequent leg will increase in difficulty level. | M |
| UR_LEGS | The competition must consist of 3 "heat" legs and a final, with the fastest from the 3 heats racing in the final. | M |
| UR_AWARDS | In the final, the 1st place team will receive a Gold medal, the 2nd a Silver medal, and 3rd a Bronze medal. | M |
| UR_PERFORMANCE | The game must look smooth when played (30fps +) | M |
| UR_JAVA | Must be coded in Java programming language | M |
| UR_BOATS_NO | The number of teams should be consistent with no. of legs so that the races have an appropriate number of boats and the race should not be cluttered. | S |
| UR_INFO_DISPLAY | During races, the user should be able to see information such as position in race, distance to go, speed, acceleration, stamina and damage. | S |
| UR_ACCESSIBILITY | The first leg/level of each race should be more accessible so that beginners can learn the dynamics of the game. | S |
| UR_FINALS_PLACING | The fastest of the three legs will be used to place teams into the finals | M |
| UR_EXCITEMENT | The game must be exciting and engaging to for the player, simulating the experience of the real York dragon race | M |

*User Requirements Table (We use the MSCW Scheme for Priority listing)*

| ID | Description | User Requirements Ref |
|---|---|---|
| FR_PLAYABILITY | The game must take the user's key-presses as input for controlling their boat in-game. | UR_PLAYABILITY |
| FR_SPEED_STAT | The speed specification of each boat should refer to its terminal velocity | UR_BOAT_SPECS |
| FR_STATS | The game has different properties for boat specification in terms of speed, acceleration, stamina, maneuverability, and robustness. | UR_BOAT_SPECS |
| FR_STAM_DECR | Stamina decreases over the duration of the race | UR_TIRED_OVER_TIME |
| FR_STAM_USAGE | How much maneuvering and changes of speed (acceleration) a boat is doing should further contribute to the increase in tiredness | UR_TIRED_OVER_TIME |
| FR_STAM_EFFECT | Stamina would restrict movements (changes of speed or maneuverability) and this effect would increase as stamina decreases | UR_TIRED_OVER_TIME |
| FR_STAM_REGEN | Stamina replenishes if movements are conservative and stamina is fully restored between the legs | UR_TIRED_OVER_TIME |
| FR_LANE_PENALTY | Boats must remain in their lane for the duration of the race. Leaving the lane should result in a penalty at the discretion of the chief race official. | UR_LANE_PENALTY |
| FR_COLLISIONS | Colliding against obstacles will progressively reduce the robustness of the boat, until it breaks down (resulting in the end of the game). | UR_COLLISIONS |
| FR_MOVING_OBSTACLES | Some obstacles should start stationary, but then in later legs start becoming more dynamic and moving left to right. Some obstacles e.g. bridge pillars are permanently stationary. | UR_OBSTACLES |
| FR_OBSTACLE_CLUTTER | Game should not display too many obstacles in order to not clutter the screen. | UR_OBSTACLES |
| FR_OBSTACLES | Game should display obstacles in the river during the race, like clueless ducks and geese, or tree branches floating down the river | UR_OBSTACLES |
| FR_DIFFICULTY | The game should increasing the difficulty level with every subsequent leg by changing the number, speed and type of obstacle (dynamic or static). | UR_DIFFICULTY |
| FR_DIFFICULTY_DISPLAY | During gameplay the game should overlay/display the difficulty level clearly and this should update to show any changes to the difficulty level based on the leg of the race. | UR_DIFFICULTY |
| FR_CONTROLS | The user should be able to see the controls for how to move the boat by pressing a certain key. | UR_DIFFICULTY |
| FR_TUTORIAL | At the start of the race there should a tutorial and/or overlay of controls that are clearly visible and understandable to a new player. | UR_ACCESSIBILITY |
| FR_LEGS | The competition must consist of 3 "heat" legs and a final, with the fastest from the 3 heats racing in the final. | UR_LEGS |
| FR_AWARDS | In the final, the 1st place team will receive a Gold medal, the 2nd a Silver medal, and 3rd a Bronze medal. | UR_AWARDS |
| FR_BOATS_NO | Each race should have 3-6 boats. | UR_BOATS_NO |
| FR_POV | The game view should be fixed in a central position for the user's boat and shows boats around the user's boat (but not all of them). | UR_BOATS_NO |
| FR_INFO_DISPLAY | During race, screen should display user's position in race, distance remaining, stamina, speed, acceleration, and damage | UR_INFO_DISPLAY |
| FR_PEN_NOTIFICATION | When a penalty is incurred a notification should be displayed on screen | UR_LANE_PENALTY |

*Functional Requirements Table*

| ID | Description | Fit Criterion | User Requirements Ref |
|---|---|---|---|
| NFR_PERFORMANCE | The game must run smoothly enough for the game to be enjoyable. | 90% of the time the user does not experience dropped frames or stuttering | UR_PERFORMANCE |
| NFR_CLUNKINESS | No clunkiness in the movement of the objects/sprites. | 90% of the time the user does feel the controls are responsive and visuals reflect this with low latency. | UR_PERFORMANCE |
| NFR_RESILIENCE | The game should not crash at any point during use or impact performance of the computer. | 99% of the time the game does not crash. | UR_PERFORMANCE |
| NFR_LEARNABILITY | The game should be easy to pick up for beginners. | 90% of users feel that they can learn the game easily. | UR_ACCESSIBILITY |
| NFR_RELIABILITY | The game should be available to use all of the time the user's computer is on. | 99% of the time the game should be available to use. | UR_PERFORMANCE |
| NFR_ACCESSIBILITY | The game should be usable for people with impaired vision. | 90% of users with impaired vision find the game easy to use. | UR_ACCESSIBILITY |
| NFR_USABILITY | The game's messages and information should be in plain English and easy to comprehend. | 95% of users find the game information easy to understand. | UR_INFO_DISPLAY |
| NFR_RESPONSE_TIME | The game should respond quickly to any input from the user. | 90% of the time the game responds within 16ms. | UR_PERFORMANCE |
| NFR_MEMORABILITY | The game should be memorable so that users know how to use it when they go back to it. | 95% of the users remember how to use the game the second time round. | UR_PLAYABILITY |
| NFR_SATISFACTION | The game's design should be satisfying to look at and use. | 90% of users are satisfied with the design. | UR_PLAYABILITY |
| NFR_EXCITEMENT | The game must be exciting for the player. | When given a 1-5 question on how exciting the game is, on average, players should give 3 or above | UR_EXCITEMENT |

*Non-functional Requirements Table*

# Bibliography

[1] I. Sommerville, Software Engineering, Pearson Education, 2008, pp. 101-122.