Section 1

Requirements

Additions: Green
Deletions: Red

1.1 Background

1.1.1 Elicitation and Negotiation

Requirements are fundamental to the software engineering process. In order for us to deliver a piece of software which is as close to our customers vision as possible, we needed to begin the requirement engineering process, starting with elicitation. We first analysed the brief as a team, and made a note of all the requirements which were clearly stated. We also highlighted parts of the brief which we thought were ambiguous and needed further input from the customer to ascertain what they required more precisely. We took these points and made a list of questions which we took to an interview with the customer to gain further insight into what they required. At the interview, the customer told us that we could decide what we thought was best for some of the details, such as the exact features of the map, and the amount of customisation that can be applied to a plot. He also cleared up certain ambiguities in the brief such as how resources are generated. Some negotiation took place over certain features. We looked at all of these as a team and decided on an initial plan of what these should be.

We took this information and created user stories to represent how a user would interact with the game. From there pulled out a series of functional and non-functional requirements that would allow those user stories to become a reality.

We compiled all of this into the first draft of our requirements, with a section for those things which would be nice to have, but were not essential according to the brief. This means that if we need to revisit our requirements during development because otherwise we would not be able to complete the project on time, we know which requirements can be changed or toned down to allow for this.

We took this first draft of the requirements back to the customer, and together we went through them all to ensure that we were agreed on what the requirements were.

1.1.2 Presentation

We will be using user stories, as detailed in Sommerville [1], these stories will detail the needs of the user for specific features and aspects of the system. We are using user stories as they are part of the agile development process and easily allow for us to iteratively adapt to change. This is useful the requirements themselves change or if we realise we need to update them due to contradictions. The user stories will also include acceptability criteria, this means that we will be easily laid out everything the system needs to be able to do. This will also feed into the test driven development we will be using later in the project as we able to see if a requirement has been fully implemented. We will not be using use cases as they are part of none agile development methods and more suited to transactional processes and are therefore inappropriate for this project.

We will then tabulate our requirements to make them more accessible to us and to the customer. In the table we will have the number and name of the user story to allow us to easily reference them. We will then have the user story itself. Finally, we will have the acceptance criteria split up into functional and non-functional requirements. When referencing requirements they will be referred to by

a series of 3 numbers separated by points. The first number will represent the position of the user story associated with the requirement in the requirements table. The next number will show if the requirement is functional, represented by a 1, or non-functional, represented by a 2. The final number will represent the position of the requirement in its cell in the table. We chose to tabulate our requirements after initially presenting our requirements to the customer as a user story followed by a list but changed them as he said it would be easier for both himself and us to read and reference.

1.2 User Stories

#	Title	Story	Functional	Non-functional
1	GUI	As a player I must be able to see a GUI consisting of a map subdivided into plots and should be able to gain information about the state of my freehold and my individual plots.	1.1.1. The entire map should be available to the user 1.1.2. Information about individual plots should be shown: (a) Which player owns a plot (b) Output of ore, energy, and food (c) Robiticons installed 1.1.3. The total amount of resources a user has must be shown on the GUI	1.2.1. The map must represent the university of York with at least 3 identifiable landmarks 1.2.2. The map must be split into multiple evenly sized plots 1.2.3. Each player must be uniquely identifiable on the map 1.2.4. The GUI should load in less than 5 seconds
2	Purchasing Land	As a player I must be able to purchase plots of land to increase the size and productivity of my freehold	2.1.1. The player must be able to exchange currency for more land during the acquisition phase of the round	2.2.1. Plots will have different strengths and weaknesses in terms of production based on location and terrain type 2.2.2. The player must be able to cancel a purchase to avoid accidental purchases
3	Plot Modification	As a player, I must be able to buy and sell various modifications to my plots to increase productivity and/or style.	3.1.1. The system must provide a number of possible modifications to plots	3.2.1. The player must be able to view all modifications and choose one to install 3.2.2. Installation must take less than a second

#	Title	Story	Functional	Non-functional
4	Multiplayer	As a player, I must be able to buy and sell various modifications to my plots to increase productivity and/or style.	4.1.1. A player must be able to chose whether to play against another human or the computer 4.1.2. At least two users must be able to play the game together 4.1.3. The players will take turns in playing	4.2.1. The simulated player should take no longer than 20 seconds to complete a round
5	Round Structure	As a player I must be able to play the game in a structured manner	5.1.1. The game must be split into multiple rounds 5.1.2. Each round should be made of 5 phases: 1. Purchase any unoccupied plots 2. Purchase and customise roboticons 3. Install roboticons on plots of land 4. The colony produces resources 5. The player can buy and sell resources 5.1.3. Phases 2 & 3 must be time limited.	5.2.1. It must be easy for the player to move between phases 5.2.2. Changes between phases must take no longer than 5 seconds
6	Roboticons	As a player I must be able to purchase and customise my roboticons so they can produce more of certain amounts of resources	6.1.1. The player must be able to purchase roboticons from the market 6.1.2. The market must have ore to produce roboticons 6.1.3. The user must be able to purchase modifications for the robiticon at the market 6.1.4. The user must be able to install modifications on roboticons 6.1.5. The user must have the option to install a roboticon on a plot of land they own.	6.2.1. At the start of the game, the market has 12 roboticons

#	Title	Story	Functional	Non-functional
7	Resources	As a player I must be able to produce resources from my plots	7.1.1. Roboticons are required to produce resources 7.1.2. During phase 4 the users roboticons will generate resources across the freehold 7.1.3. Food, energy and ore will be generated 7.1.4. Different amount of resources will affect the rate of production	7.2.1. The resource production should not take more than 5 seconds 7.2.2. The resource production should happen automatically
8	Buying/selling resources	As a player, I must be able to buy and sell resources to other players through an auction, or to the market at a fixed price so that I can maximise my wealth and productivity.	8.1.1. The system must provide an auction facility, where the other player and the market bid for resources 8.1.2. The system must choose a market price based on resource abundance 8.1.3. The player must be able to buy/sell resources from/to other players, or the market	8.2.1. At the start of the game, the market must have 16 units of food and energy and 0 units of ore 8.2.2. At the start of the game, the player must have a small amount of money
9	Gambling	As a player, I must be able to enter the bar and either win or lose money.	9.1.1. The system must provide a minigame where the player can gamble with their money	9.2.1. The minigame must give feedback on the money won or lost
10	Winning	As a player, I must be able to win or lose the game.	10.1.1. The system must assign a value to each resource at the end of the game, from which a player's final wealth is calculated 10.1.2. The game must end on the round in which the last plot of land has been allocated. 10.1.3. The player with the highest final wealth must be declared the winner, and Vice-Chancellor of the colony	

The main risks associated with these requirements are risks 3, 4, 5 & 6 as certain requirements may not be able to be implemented due to available tools or staff ability, however risks 8 and 11 must also be considered as they may change the requirements themselves.

Bibliography

[1] I. Sommerville, *Software Engineering*. Harlow, United Kingdom: Pearson Education, 10 ed., 2016.