University of York

Department of Computer Science

SEPR - Assessment 2

Updated

Requirements

Team Craig

Thomas Burroughs

Huw Christianson

Joseph Frankish

Isaac Lowe

Beatrix Vincze

Suleman Zaki

Requirement Elicitation Process

As specified in our Method Selection and Planning section, the team will be following an agile method, namely an adaption of the Scrum method. This method promotes face-to-face meetings and communication with the customer. Using this and information gathered from the reading of Ian Sommerville's Software engineering book [1] we devised a suitable process for requirement elicitation. Our process follows the four stages set by Sommerville [1]:

1. **Requirement discovery**- Involves interaction between the team and the customer to discover requirements.
2. **Requirement Classification/Organisation**- Involves the categorising of requirements e.g Function and non-functional requirements.
3. **Requirement Negotiation**- Involves further meetings with customer to resolve ambiguous requirements and discuss the degree of necessity of each requirement (whether they are essential or optional).
4. **Requirements specification-** Involves the documentation of the negotiated requirements.

Below outlines the first three stages of our elicitation process with the last stage, requirement specification, outlined in a separate section:

On receiving the written brief, the team gathered to brainstorm potential requirements that the game must fulfil and note questions about ambiguous statements in the brief that must be clarified on meeting with the customer. These requirements were often vague and required more elaboration. This brainstorm approach paid dividends, as it allowed all group members to engage with all parts of the brief, and fully understand how their future contributions to the project would connect with other areas of the project. During the first meeting with the customer the noted questions were asked and answers regarding the clarification of requirements were noted so further discussion with the team could commence. From these answers, the team discussed potential design ideas for the game and how these ideas would meet the requirements of the customer, allowing for all team members to engage on a design level, particularly group members that felt less confident about upcoming programming tasks. On research of requirements engineering it became clear that prototyping could help to elicit requirements (As discussed in the IEEE practice for system requirements specification [2]) , as well has give an insight into potential software architecture. Due to this each team member was asked to create and provide a paper prototype of a specific game feature. Once these prototypes were created, the group met once more, to create a unified paper prototype, allowing the group to consolidate their vision for the games finished product. This unified paper prototype was presented to the customer at the next meeting. Discussion and negotiation at the second meeting, prompted by the paper prototype, elicited more requirements for the project which may have previously been unanticipated. These requirements were agreed upon by both the team and client to be optional.

Once all feedback from the customer had been discussed, the team created a Single Statement of Need (SSON) which was later agreed upon with the customer. This statement gives a comprehensive understanding of the desired outcome of the game/project and can provide a simple testable measure of how successful the requirements process have been completed. For this project the agreed SSON is as follows: "The system will deliver a game about zombies, given in a top-down perspective, with a novel mechanic."

This approach allows us to stay focused on the project and keep ideas to an appropriate scope, although it also hampers creativity at certain points in the creation process.

Requirement Specification and Presentation

A software requirements specification (SRS), is a detailed description of a software system to be developed with its functional and non-functional requirements. Our SRS was developed based off the agreement that was formed in the meetings between our team and the customer, as described in the elicitation process detailed above. Using the recommended IEEE practice for system requirements specification, our SRS will adhere to the following: correct; unambiguous; complete; consistent; verifiable; modifiable and traceable [2]. We decided to follow IEEE's practice for system requirements specification as it allowed us to form more concise requirements which follow specific conventions as outlined by IEEE's practice. By following these conventions, the group can ensure that each modification of a requirement remains consistent with the previous and adheres to the industry's definition of a 'good' requirement. Previous requirements which did not follow the standard were shown to possess inconsistencies which led to requirements either not being testable, despite the fit criterion, or contained ambiguous natural language which led to the misinterpretation of the intent and degree of necessity (essential, conditional or option) of the requirement. By following the IEEE specification, other teams should be able to evolve the SRS with greater ease by adhering to the protocols written below . The IEEE practice ensures that our SRS has a suitable balance between comprehensiveness for the customer and enough precise detail for developers and testers [1].

Although the team tried to adhere to the standard set by IEEE it is clear that the convention set needs to be adapted for our specific use and project scope. Time and resource restrictions mean that the team cannot enforce all the practices. Some requirements laid out by the IEEE practice are not suitable for our small scope project. For example, safety requirements, which take into account safety certifications and security requirements, which take into account privacy issues and data protection, were omitted from our SRS as we could not elicit any requirements of these types.

Introduction

Purpose: This software requirement specification outlines the requirements for the game created by Team Craig for the SEPR Module at the University of York. The customer has been declared as the head of the SEPR module.

Document Conventions:

- **Fit Criterion**: Each requirement within the table has an accompanied fit criterion. The fit criterion is used to quantify or measure the requirement which makes it testable, which will allow the team to determine whether a specific implementation actually meets the requirement [3].
- **Risks, Environmental Assumptions, Alternatives:** Alongside each requirement are environmental assumptions, risks and alternative for that specific requirement. This will help to assess and minimise potential risks that may arise. Alternatives provide evidence of further requirement elicitation and is indicative of the decision process.
- **Requirement ID:** Alongside each given requirement within a table is a distinct requirement ID. This requirement ID allows the requirement to be traceable throughout all of the project's documentation.
- **Requirement Category:** Each requirement has been categorised as either 'functional' (F) or 'non-functional'(NF), and further categorisation of the non-functional category has occurred with, for example, performance (P) , constraint(C), User Interface (UI) and maintainability (M) requirements. More non-functional requirement categories are likely to expand as more requirements are developed.
- **Key Words:** 'Must' and 'Should' describe the degree of necessity for each requirement. 'Must' is used in requirements which are essential, meaning the game will not be accepted if these requirements are not met. 'Should' is used in requirements which are conditional, meaning that they should enhance the game but the game will be accepted by the customers without these requirements being met.

SSON: "The system will deliver a game about zombies, given in a top-down perspective, with a novel mechanic."

Our requirements are represented in a table to improve legibility and minimise documentation to adhere to our agile method.

Requirement ID	Requirement Description	Fit Criterion	Environmental Assumptions, Risks or Alternatives
F1	Game should incorporate a novel mechanic	The game will contain an aspect to the game not included in the brief, and not conceptualised by the client - a novel mechanic. This aspect will function, and function without adversely affecting the aspects of the game explicit in the client's brief.	As is, it should not be a higher priority than making a functional product that meets client requirements, but if the mechanic(s) are crucial to system use, then neglecting them at an early stage of the project can make it difficult to work with down the line
F2	Game must include a minigame, distinct from the main game.	The game contains a mini-game with self-contained mechanics. This can be entered by clicking on a graphic on the world map. The mini-game can be replayed as many times as the player wants.	Assume that like the main game, that the mini-game is not too graphically intensive and can be run of the pc's in the software lab.
F3	Game must contain 5 powerups	The game will contain the following 5 power-ups: health boost; damage boost; speed up; nuke; rapid fire. These will can be acquired using virtual currency. The power-ups will noticeably change the state of the game for a limited time.	Trying to reach too far with the powerups risks wasted time making unnecessarily complicated. Risk that the powerups do not provide enough trait enhancements to make it worthwhile
F4	Game must have 6 six different locations.	The game will have 6 distinct locations based on real-life locations at the University of York. They are as follow: Ron Cooke Hub with Lake, Langwith, Goodricke, Constantine and the Retail Park. These locations should be distinct and distinguishable in a player's memory.	Risk that some locations may be too similar to each other making gameplay seem repetitive.
F5	Game must contain 3 different characters	The player can select from 3 characters, all of which have a distinct visual appearance and mechanical differences.	Attempting to implement character traits that are too complicated could be time-consuming for minimal payoff. Character traits will be kept simple. Risk that characters' traits are not evenly matched or distinct enough leading to the players only using one player.
F6	Game must contain 2 bosses	The player will have to engage in combat first with a massive goose living below the Ron Cooke Hub Lake and then Koen Lamberts (with a goose's head) on entering the Computer Science building. Bosses will have enhanced characteristics, such as health and attack power, that make them feel distinct from fighting regular enemies	There is a risk that bosses posing an improper level of challenge will restrict player engagement, i.e. an easy boss is boring, and an overly hard boss is frustrating.

F7	Player must be able to win the game after visiting all 6 locations and defeating all bosses	The player can complete the game once the player has unlocked all location achievements by visiting each location and defeating Koen Lamberts and the giant goose.	Assume that all 6 locations are visitable and that both bosses are beatable. Assume that the player understands that completion is the aim of the game. Risk that the player does not attempt to finish the game through lack of interest.
F8	The game must contain regular enemies	Game contains zombies (all of the same graphic design and attributes) which will roam the world map and distribute from the edges of the map. Zombies will approach the player when they enter the players viewing frame. Players will then be able to attack zombies by clicking in the direction of the zombie using the selected weapon.	Assume that the machine will be powerful enough to support an arbitrary number of these enemies. Assume the player is of a suitable age to be exposed to 'violent' themes.
F9	Game should contain a way to navigate between locations	The game has distinct pathways between the 6 locations similar to those on the Hes East Campus which players cannot stray from. Players can move characters within locations using the w,a,s,d or arrow keys	Assume the machine has supported periphery. Alternatives include modelling for different keys and controller support
F10	Player must be able to control the character during gameplay	The player will be in control of the player whilst the game is running using keyboard controls. Players will only not be in control of players when the game is paused, the player is on the menu screen or during cutscenes..	Assumption that player is not away from keyboard whilst game is running. Alternatively, player is in control using a different peripheral such as a controller. Assume inputs given while the game is an active program are meant for the game
F11	Use of the lake location on campus should allow the player to access a fishing minigame	Game contains a graphic at the boundary of the Ron Cooke Hub lake which allows the player to enter the mini-game whilst pausing the main game. Alternatively, this can just be accessible through menus either when the game starts or through the menu which appears when the game is paused	Risk that the game does not communicate the lake's purpose or entice players to play the minigame.
F12	Game should include virtual currency	The player can collect coins on route to the 6 locations and the two bosses and use these to trade for Items at the retail park.	Risk that tradable items do not provide a clear benefit for the player, resulting in currency being an effectively redundant mechanic. Alternative: players forage for food/health packs to improve their stamina and health- Risk that this could be too tedious.
F13	Game should contain a way for	The player can collect items/travel to some location which will unlock a new	Risk that collecting items unlocks new sections of the map is not

	players to progress through locations	location progressing the player through the game.	obvious and players become confused on how to progress. This could mean players lose interest in game's story.
F14	Player should be able to pause the game.	The game includes a key binding which allows the game to be paused and resumed during gameplay.	Risks: Players are not aware of key bindings and hence, do not use feature. Alternatives: Minimising game window using OS procedures.
F15	Player should be able to exit the game.	The game includes an exit button on the main menu which closes the game executable.	Alternatives: Game exited in windows mode using operating system procedures to close programs. Risk: Accidental loss of saved data due to unclear exiting procedure.
C1	Project must be completed in its entirety by Wednesday 01/05/2019	Implementation and deliverables will be finished by 01/05/19. A Gantt chart has been created given specifics on time allocations and future scheduling.	Risk D1 and D2 (See Risk Assessment and Mitigation)
C2	Game must run on Windows 10	Game successfully runs with all added functionality on PCs in the software lab running on the Windows 10 operating system.	Assume players will choose to run Windows 10 on the Software lab PCs. Risk that PCs in the software lab are not using the Windows 10 operating system, making the game unplayable.
M1	Game must be structured such that a transition to another software engineering team can be completed.	Game runs from fully commented code with suitable well defined architecture diagrams. Game code abides by the quality standards set at the beginning of game's implementation.	Assumes that quality standards set by both teams are agreeable and consistent. Risk that commented code is insufficient and no teams choose to transition with our game.
P1	Games must run smoothly on software lab computers	Game can run at a minimum of 30 FPS throughout the entirety of gameplay on the PCs in the Computer Science software labs.	Assume that the pc's in the have the minimum hardware to run the game at 30 FPS and can use this as a benchmark for other pc's. Risk that changes to the hardware of the pc's in the software lab will negatively affect performance of the game.
UI1	Game Menu should be intuitive and easy to navigate	New users shall be able to select a character and enter the world map on the first attempt at using the game within 1 minute.	Assume that the player is not impaired and capable of operating a valid control scheme. Assume that the loading sequence of the game, including cutscenes, does not exceed 1 minute.

References

[1] I. Sommerville. *Software Engineering*. Pearson, tenth ed., 2016

[2] IEEE, 830-1998. *IEEE Recommended Practice for Software Requirements Specifications*. Institute of Electrical and Electronics Engineers, 1998.

[3]- S.Robertson et al. *Mastering the Requirements Process*. Sep 2012. Available at: http://www.informit.com/articles/article.aspx?p=1929849&seqNum=7