SE 3A04: Software Design III: Large System Design

Group #5, Spaceship System Sabotage
Pareek Ravi 001407109
Pavle Arezina 001410366
David Hobson 001412317
Victoria Graff 001401451
Julian Cassano 001406891

February 13, 2017

Contents

List of Tables

1 Introduction

1.1 Purpose

The purpose of the software requirements specification for this project is to document the results of the analysis of the requirements pertaining to this project. This document will demonstrate the mutual understanding of the problem to be solved and what the eventual system will be tested against to ensure that the project fulfilled its objectives. The document has an important role in laying down the foundation of what the rest of the project will be built off of. The intended audience of the software requirements specification is for the software developers, researchers, and advanced users who would be involved and interested in the basis of how the project's development was determined. It allows the specified groups of people to understand the reasoning behind the decisions made in regards to the project.

1.2 Scope

The application being developed, called Spaceship System Sabotage, is a simulation of a fictional spaceship that the user will interact with to ensure the spaceship reaches the objective of the simulation. An engaging and entertaining experience will be provided to the user in a real-time simulator that will provide various challenges for the user to overcome. It will not simulate realistic physics or be restricted to non-fictional material. Spaceship System Sabotage is a purely entertainment application that responds to the users stimuli in response to various events. This will provide a satisfying experience to the user that will ensure the user's interest is held. It is not meant to frustrate the user into not utilizing the application anymore but to keep Spaceship System Sabotage challenging enough to ensure the user is not bored.

1.3 Definitions, Acronyms, and Abbreviations

Table 1: **Table of Acronyms, Abbreviations, and Symbols**

Table 2: **Table of Definitions**

Term	**Definition**
Sabotage	Deliberately destroy, damage, or obstruct (something)
Defector	A person who has abandoned their country or cause in favor of an opposing one
Lore	The story or reasoning behind occurances.
TA	Teaching Assistant to the professor of an university.
Android OS	The operating system on phones that run Android

1.4 References

none

1.5 Overview

The rest of the SRS is separated into three sections, the first being the overall description of the project. This section describes the general factors that affect the product and its requirements. It does not state specific requirements; it provides a background for those requirements and makes them easier to understand. Functional requirements are the next section which contain a sufficient level of detail to enable designers to design a system to satisfy all the requirements stated. These requirements specify what the project must do and describe the actions that the product must perform and works as intended. The last section of the SRS is the non-functional requirements which also assist in guiding developers in what the final state of the product is. An appendix is also included to the SRS that clearly states how the division of labour was between the software developers in the project.

2 Overall Description

2.1 Product Perspective

Spaceship System Sabotage is an independent and self-contained system that simulates a spaceship. The typical user experience with this application is to keep the system operating for as long as possible so that after a specified period of time the space craft will "reach its intended destination" and the user will reach an end state through a win condition. This perspective on the interaction of sub-systems will make the application feel more like a game than it does a simulation. When a sub-system is triggered with a stimulus, eventually leading to a sub-system failure, the application will tell the user that one of the crew members of the ship has **sabotaged** a sub-system. This acts as a challenge for the user to keep the crew members alive but attempt to determine who the **defector** of the crew is, providing **lore** and context for the simulated sub-system failure so that the user has motivation to keep the system functional.

2.2 Product Functions

- The user will be able to start a simulation of a system

- The different sub-systems will be displayed for the user
- The user will be able to switch views to choose the sub-system they are observing
- The system will allows the user to combine views
- Messages/alerts will be displayed for the user
- The sub-systems will be able to receive stimulus from the application to prompt a status change
- The sub-systems will be able to affect other sub-systems depending on their status
- The user will be able to input a stimulus that will have a specified effect on a sub-system, namely a status change back to the original state
- The user will be able to pause/resume the system
- The simulation will be able to reach an end state at any time and terminate the simulation
- The user will be able to end the simulation at any time if they so choose

2.3 User Characteristics

The intended user of this application is anyone who enjoys games or challenges and has a basic understanding of systems and how they can interact. No previous work or school experience with designing systems is required as this application is not a test on the user's understanding of software architecture. This application aims to provide a casual experience of simulating the negative effects of sub-systems failing in a larger system and allows the user to have a role in this scenario regardless of failure or not. Each end state is an acceptable level of completion in terms of the simulation as they both result in a demonstration of the interaction between sub-systems. This means that the application is accessible to anyone from casual players in elementary school to Dr. Khedri while he's killing time waiting for his **TAs** to mark these deliverables.

2.4 Constraints

The limitations of this project come in the form of development time. The allocated amounts of hours that we intend to dedicate to this project are in negative correlation with the amount of other school work we have to do during the time-frames of each deliverable. We would love to make this a full fleshed out game that is fun and challenging but at the moment we do not have the time to comfortably introduce enough additional features to make this a fun to play, sellable game. The rubric specified by the **TAs** will first and foremost be followed, and only then can additional features be added.

Another limitation of this project would be the programming language chosen for the application. Depending on what language the development takes place in, certain features would be more difficult to implement in the amount of time given. For example, our team is more experienced in Java and may be more proficient in meeting the functional requirements with this language but it may be harder to meet the non-functional requirements, especially in terms of visual interfacing compared to working in a language like C++.

2.5 Assumptions and Dependencies

The assumptions in the requirements section of this document are focused around the intended interface of the application. The proposed layout of the visual interface is that the sub-systems will be laid out in a representation of their location in physical space, meaning that there will appear to be a space craft in the user's view and all of the interactions will happen through this apparent view. This also means that sub-systems will be layered on top of each other and will be difficult to view all at once, hence the requirement that the user will be able to change views and customize the sub-systems being displayed at one time.

2.6 Apportioning of Requirements

Some non-functional requirements that could take too long to implement and are not explicitly required for a functional demonstration may be delayed until later versions of the application. Some areas of development where this is a concern are the low loading times and the "clean" interface design.

3 Functional Requirements

BE1. User wants to combine the views of multiple sub-systems

VP1.1 User

i. The system will provide an interface for the user to interact with when combing views
ii. The system will allow the user to select views to be combined
iii. The system will display the combined views in a layered format
iv. The system will be able to support any amount of views to be combined.

BE2. User wants to add a sub-system view

VP2.1 User

i. The system will provide an interface for the user to interact with when adding views
ii. The system will allow the user to select a view to be shown on the system

BE3. User wants to remove a sub-system

VP3.1 User

i. The system will provide an interface for the user to interact with when removing views
ii. The system will allow the user to select a view to be removed from the system

BE4. User wants to stimulate a subsystem

VP4.1 User

i. The system will provide an interface for the user to interact with different sub-systems and be able to select a particular sub-system
ii. The system must display the result of when the stimulus takes place

BE5. User wants to start the system

VP5.1 User

i. System must provide an interface to allow the user to start the overall system
ii. System must provide an option for selecting 3 or more sub-systems to have enabled
iii. System must display only the overall system and none of the sub systems

BE6. User wants to pause the system

VP6.1 User

i. System must provide an interface to pause the system
ii. System must provide a means to alter the game settings
iii. System must display the current state of all the sub-systems

BE7. User wants to resume the system

VP7.1 User

i. System must provide an interface to resume the system
ii. The system must resume to the exact state before it was paused
iii. The system must display the same set of sub systems as it was before pausing

BE8. User wants to end the system

VP8.1 User

i. System must provide an interface to allow the user to end the overall system
ii. System must show the final state of all the sub-systems
iii. System must provide an option for the user to restart the system from initial conditions

BE9. Random Time Stimuli to the System

VP9.1 User

i. System must stimulate a sub-system with a random event defined by parameters
ii. System must determine the event based on a weighted system to determine which event will be simulated from a set of events

iii. System must display the stimulus and display a text description of the stimulus
iv. System must clearly indicated the region of the system affected by the stimulus

4 Non-Functional Requirements

4.1 Look and Feel Requirements

4.1.1 Appearance Requirements

LF1. The application shall have a clean, minimalistic user interface. The systems on the ship will be colour coded and easy to differentiate on the map.

4.1.2 Style Requirements

LF2. The application shall have smooth transitions while responding to user input.

4.2 Usability and Humanity Requirements

4.2.1 Ease of Use Requirements

UH1. The application shall be simple to use for all users above the age of twelve. A new user shall be able to navigate through menus and play the game with ease.

4.2.2 Personalization and Internationalization Requirements

none

4.2.3 Learning Requirements

UH2. The learning time within the application shall take no longer than five minutes.

4.2.4 Understandability and Politeness Requirements

UH3. The application will be easy to understand and will refrain from inappropriate language or slang words.

4.2.5 Accessibility Requirements

UH4. The application will be playable for the hearing impaired.

4.3 Performance Requirements

4.3.1 Speed and Latency Requirements

PR1. The application shall update systems from user input in under 5 seconds, and load time between screens shall be under 5 second.

4.3.2 Safety-Critical Requirements

none

4.3.3 Precision or Accuracy Requirements

PR2. The touch user input on the application will have 90 percent accuracy.

4.3.4 Reliability and Availability Requirements

PR1. The application shall be available at all times.

4.3.5 Robustness or Fault-Tolerance Requirements

PR3. The application will be designed to handle all selection errors by the user.

4.3.6 Capacity Requirements

PR4. The application will be able to handle input from one user.

4.3.7 Scalability or Extensibility Requirements

PR5. The application will be created to be able to integrate more space ship systems in the future.

4.3.8 Longevity Requirements

PR6. The application will operate as long as the **Android OS** version it is created on, is supported.

4.4 Operational and Environmental Requirements

4.4.1 Expected Physical Environment

OE1. The application will run on Android smartphones.

OE2. The application shall be used at any location.

4.4.2 Requirements for Interfacing with Adjacent Systems

none

4.4.3 Productization Requirements

none

4.4.4 Release Requirements

OE3. The application will be downloadable through the Google Play Application store.

4.5 Maintainability and Support Requirements

4.5.1 Maintenance Requirements

MS1. The application shall be built through modular design for easy maintenance and for updating features in the future.

4.5.2 Supportability Requirements

MS2. The application will only run on android smartphones.

4.5.3 Adaptability Requirements

MS3. The application will be designed to be able to re-format to fit all android phone screen sizes.

4.6 Security Requirements

4.6.1 Access Requirements

SR1. The application will be availability to users that own an android smartphone.

4.6.2 Integrity Requirements

none

4.6.3 Privacy Requirements

none

4.6.4 Audit Requirements

none

4.6.5 Immunity Requirements

none

4.7 Cultural and Political Requirements

4.7.1 Cultural Requirements

CP1. The application will display language in the English language and will avoid mention or assumption of any specific culture.

4.7.2 Political Requirements

CP2. The application will not mention any political topics or issues.

4.8 Legal Requirements

4.8.1 Compliance Requirements

LR1. The application shall adhere to all compliance laws in Ontario, Canada.

4.8.2 Standards Requirements

LR2. The application shall adhere to all standards laws in Ontario, Canada.

A Division of Labour

Member	**Duties**	**Signature**
David Hobson	Functional Requirments Editing	
Pavle Arezina	Introduction Editing	
Pareek Ravi	Functional Requirments	
Victoria Graff	Non-functional Requirments	
Julian Cassano	Overall Description	