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

March 9, 2017

Contents

List of Tables

List of Figures

1　Introduction

1.1　Purpose

The Large System Design document is responsible for the high level architecture design of the system and the results produced from this analysis will be utilized in the design phase. The document is derived from the requirements and specifications of the overall system, determined in the previous software requirement specification document. The intended audience of the document is mainly for the developers who will be utilizing the large system design document to understand the overall design of the program to help structure the responsibilities assigned to each class. Developers who are maintaining the program may also use the document to ensure new additions to the program follow the system design properly to ensure a smooth experience for the user.

1.2　System Description

Space system sabotage is a simulation of a fictional spaceship that is travelling to a pre-determined location in a fixed amount of time. The user will interact with the system through random events based on a function of the running time of the simulation. Random events will involve negative situations that occur, such as comets damaging the ship or part of the ship being sabotaged, where the user will have to respond to fix them in a timely manner with a selection of tools. If successful, the space ship continues toward its goal but if an incorrect solution was chosen then the problem is spread to other systems. If enough events happen with incorrect solutions to them, then the space ship will slowly spiral out of control and the user will fail the simulation. This system offers an entertaining environment to the user where they will be challenged to keep the spaceship intact until it dock import at the space station.

1.3　Overview

The rest of the Large System Design document is separated into four sections, the first being the use case diagram. This diagram models the functionality of the system using actors, the users of the system, and use cases of the system. The next section is the analysis class diagram which describes the key classes of a system and their interrelationship through boundary class, entity classes, and controller classes. These classes of the system are identified by extracting through the description of the use case descriptions. Section three should provide an overview of the overall architectural design of the application where the division of the system into subsystems follows high cohesion and low coupling. Last section is the Class Responsibility Collaborator (CRC) cards on the classes in the analysis class diagram where each card identifies and assigns responsibilities to classes required to build the system. After these four sections there is an appendix that includes the division of labour.

2 Use Case Diagram
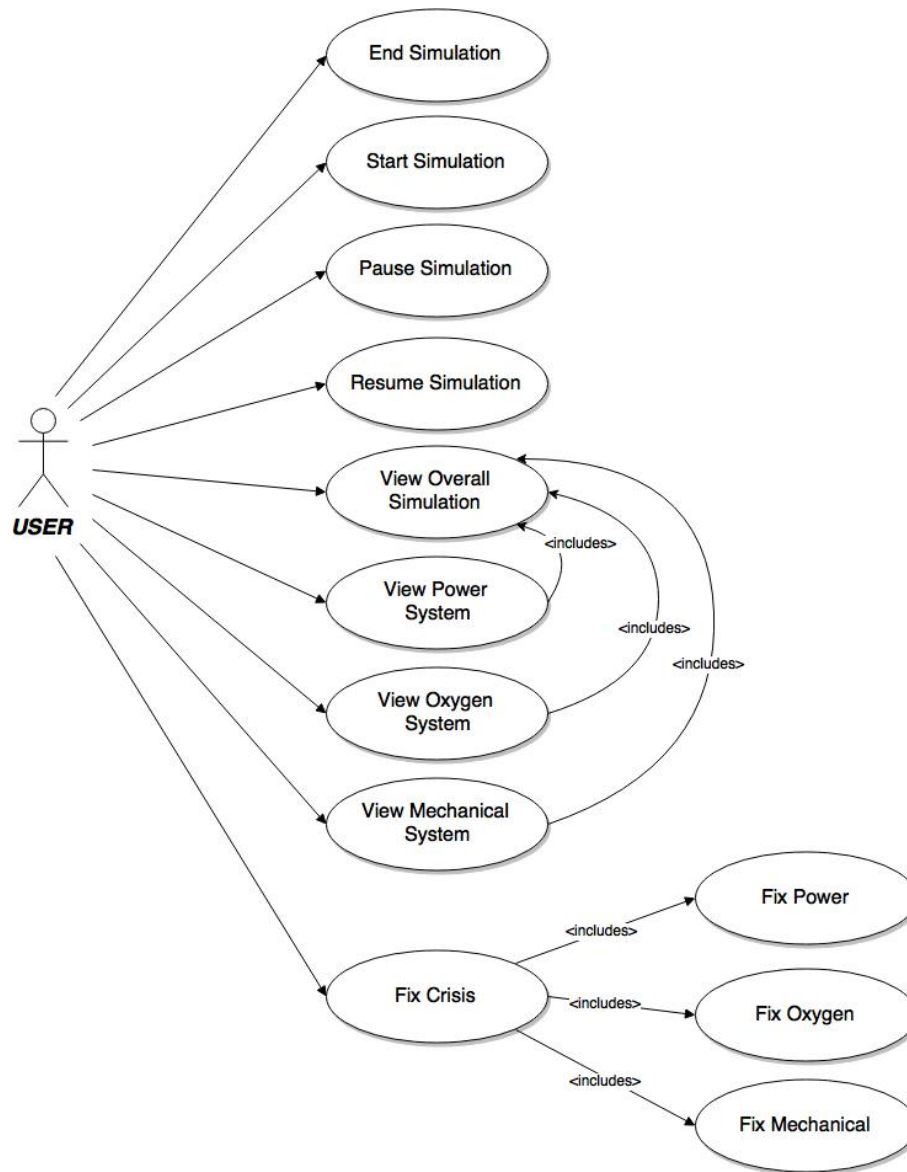

Figure 1: Use Case Diagram

Use Case Definitions

a) End Simulation: The user is in a play session and intends to indefinitely stop the simulation and return to the main menu.

b) Start Simulation: The user is in the main menu and intends to begin a play session.

c) Pause Simulation: The user is in a play session and intends to pause the simulation but intends to resume at a later time.

d) Resume Simulation: The user is in a paused play session and intends to resume and continue the simulation.

e) View Overall System: The user is in a play session and intends to display the status of all the subsystems at once.

f) View Power System: The user is in a play session and intends to include the status of the Power System in the displayed view.

g) View Oxygen System: The user is in a play session and intends to include the status of the Oxygen System in the displayed view.

h) View Mechanical System: The user is in a play session and intends to include the status of the Mechanical System in the displayed view.

i) Fix Crisis: The user is in a play session and intends to resolve an event that is negatively affecting one of the subsystems.

j) Fix Power: The user is in a fix crisis event and intends to resolve an event affecting the power system.

k) Fix Oxygen: The user is in a fix crisis event and intends to resolve an event affecting the oxygen system.

l) Fix Mechanical: The user is in a fix crisis event and intends to resolve an event affecting the mechanical system.

3 Analysis Class Diagram

Figure 2: Analysis Class Diagram of System

4 Architectural Design

This section should provide an overview of the overall architectural design of your application. You overall architecture should show the division of the system into subsystems with high cohesion and low coupling.

4.1 System Architecture

The main controller of the system is the primary interaction point for all classes. Every time a use case event is instantiated, the main controller will be prompted to interact with the specific class that is associated with the attempted performed action. This means that each class that can return values will be dependent on the main controller in order to receive a prompt at the correct time. The system is a presentation-abstraction-control (PAC) architecture, being similar to MVC but not identical. The reason this architecture style was chosen is because we wanted each section of the architecture to only interact through the

controller, to reduce conflict and clutter in the classes. The following diagram shows the main controller interacting with each set of classes individually, with no interaction directly between classes.
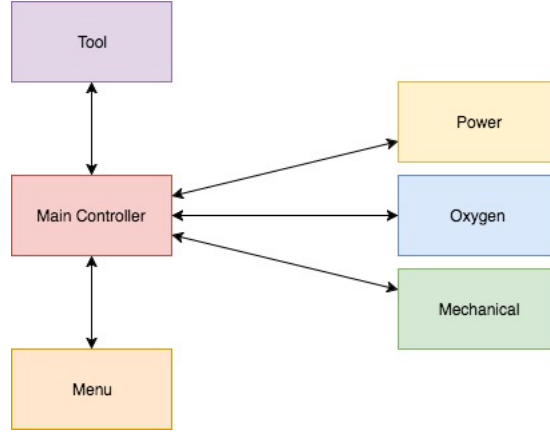

Figure 3: Structural architecture diagram for the PAC system.

4.2 Subsystems

- The power subsystem controls all of the all of the interactions that relate to the simulated amount of electrical power in the system and is dependent on the main controller. This subsystem can "malfunction" and must be stimulated to return to base state.

- The oxygen subsystem controls all of the interactions that relate to the simulated amount of air in the system and is dependent on the main controller. This subsystem can "malfunction" and must be stimulated to return to base state.

- The mechanical subsystem controls all of the interactions that relate to the simulated physical moving parts in the system and is dependent on the main controller. This subsystem can "malfunction" and must be stimulated to return to base state.

- The tool subsystem controls all of the interactions that will "fix" the other simulated subsystems and return them to normal state.

- The menu subsystem controls all interactions outside of the play session and will allow the user to performa actions such as pausing, resuming, or quitting the play session.

- The main controller subsystem is the main hub of interaction between all other subsystems and will apporpriately stimulate each subsystem at the correct time. This subsystem receives input from all other subsystems and returns the output back to each system to determine what will happen in the simulation.

5 Class Responsibility Collaboration (CRC) Cards

This section should contain all of your CRC cards.

a) Provide a CRC Card for each identified class

b) Please use the format outlined in tutorial, i.e.,

Class Name:	
Responsibility:	**Collaborators:**

Class Name: Start Screen	
Responsibility:	**Collaborators:**
Receive request to display a prompt to start game	Menu Controller
Display screen message to user	
Respond to prompt being pressed by user	
Send request to Menu Controller to start game	Menu Controller

Class Name: End Screen	
Responsibility:	**Collaborators:**
Receive request to display a prompt to end game	Menu Controller
Display screen message to user	
Respond to prompt being pressed by user	
Send request to Menu Controller to end game	Menu Controller

Class Name: Pause Screen	
Responsibility:	Collaborators:
Receive request to display a prompt to pause game	Menu Controller
Display screen message to user	
Respond to prompt being pressed by user	
Send request to Menu Controller to pause and unpause game	Menu Controller

Class Name: Success Screen	
Responsibility:	Collaborators:
Receive request to display a screen message when game has been won	Menu Controller
Display screen message to user	

Class Name: Failure Screen	
Responsibility:	Collaborators:
Receive request to display a screen message when game has been lost	Menu Controller
Display screen message to user	

Class Name: Choose Tool	
Responsibility:	Collaborators:
A view for the user the select the tool they wish to use to fix the issue on the spaceship	Tool Controller

Class Name: Tool Fail	
Responsibility:	**Collaborators:**
A view to show the user that they chose the wrong tool to fix the issue	Tool Controller

Class Name: Tool Success	
Responsibility:	**Collaborators:**
A view to show the user that they chose the correct tool to fix the issue	Tool Controller

Class Name: Event Alert	
Responsibility:	**Collaborators:**
A view to show the user that there is an issue with the spaceship	Tool Controller

Class Name: Event Timer	
Responsibility:	**Collaborators:**
A timer to handle the duration of an event	Tool Controller

Class Name: Tool Controller	
Responsibility:	**Collaborators:**
Get user's tool choice	Choose Tool
Determine if it was the correct tool to use	Tool Success, Tool Fail
Inform the user of an issue	Event Alert
Know the event has occured	Overall Controller

Class Name: Power Model	
Responsibility:	**Collaborators:**
Hold the information of the power system	

Class Name: Power View	
Responsibility:	Collaborators:
DIsplay the power system to the user	Power Controller

Class Name: Power Controller	
Responsibility:	Collaborators:
Updates subsystem information after tool usage	Power Model, Overall Controller
Tells system of subsystem information	Power Model, Overall Controller
Indicate to the view what to display	Power View, Power Model
Generate the stimulation based on a time	Overall Controller

Class Name: Oxygen Model	
Responsibility:	Collaborators:
Hold the information of the oxygen system	

Class Name: Oxygen View	
Responsibility:	Collaborators:
DIsplay the oxygen system to the user	Oxygen Controller

Class Name: Oxygen Controller	
Responsibility:	Collaborators:
Updates subsystem information after tool usage	Oxygen Model, Overall Controller
Tells system of subsystem information	Oxygen Model, Overall Controller
Indicate to the view what to display	Oxygen Model, Oxygen View
Generate the stimulation based on a time	Overall Controller

Class Name: Mechanical Model	
Responsibility:	**Collaborators:**
Hold the information of the mechanical system	

Class Name: Mechanical View	
Responsibility:	**Collaborators:**
DIsplay the mechanical system to the user	Mechanical Controller

Class Name: Mechanical Controller	
Responsibility:	**Collaborators:**
Updates subsystem information after tool usage	Mechanical Model, Overall Controller
Tells system of subsystem information	Mechanical Model, Overall Controller
Indicate to the view what to display	Mechanical Model, Mechanical View
Generate the stimulation based on a time	Overall Controller

A Division of Labour

Include a Division of Labour sheet which indicates the contributions of each team member. This sheet must be signed by all team members.

Member	Duties	Signature
David Hobson		
Pavle Arezina		
Pareek Ravi		
Victoria Graff		
Julian Cassano		

IMPORTANT NOTES

- Please document any non-standard notations that you may have used

  - *Rule of Thumb*: if you feel there is any doubt surrounding the meaning of your notations, document them

- Some diagrams may be difficult to fit into one page

  - It is OK if the text is small but please ensure that it is readable when printed
  - If you need to break a diagram onto multiple pages, please adopt a system of doing so and thoroughly explain how it can be reconnected from one page to the next; if you are unsure about this, please ask about it

- Please submit the latest version of Deliverable 1 with Deliverable 2

  - It does not have to be a freshly printed version; the latest marked version is OK

- If you do <u>NOT</u> have a Division of Labour sheet, your deliverable will <u>NOT</u> be marked