SE 3XA3 Module Guide: Revision 1

Team 03, Pongthusiastics
Adwity Sharma - sharma78
Arfa Butt - buttaa3
Jie Luo - luoj3

December 8, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
November 9, 2016	1.0	Created Module Guide
November 11, 2016	2.0	Divided sections between group members
November 13, 2016	3.0	Created format for Module Guide and added sections 2 and 4
November 13, 2016	4.0	Sections 1, 3 and 6 added
November 14, 2016	5.0	Final version with all sections added
December 3, 2016	6.0	Section 1 to 4 revised for Revision 1
December 8, 2016	7.0	Section 5 to 7 revised for Revision 1

1 Introduction

1.1 Project Introduction

This project is the redevelopment of a Pong game found on GitHub. The new game FaultInOurPong developed by the PongThusiastics Team would not only fix bugs currently discovered in the previous project, but also add more entertaining features in order to maximize the satisfactions of potential players. Apart from the executable, several formal documents and test cases that are crutial in the process of software development process and management are made for the public and internal developers.

1.2 Document Overview

This document indicates the Module Guides for the implementation of the "Fault in Our Pong" project. This document is intended to facilitate the design and maintenance of the project. The main purpose of the Module Guide (MG) is to give an overview of each module in a project after the system decomposition. The Module Guide is formulated after the completion of Software Requirement Specifications (SRS). The Software Requirement Specification describes all the functional and non-functional requirements after project research and interviews with stakeholders.
The completion of the Module Guide will facilitate the production of Module Interface Specification (MIS). The Module Interface Specification exposes the secrets in each module, and it describes the detailed constructions of the modules in words. Compared to the Module Guide as the black box of modules in a system, the Module Interface Specification serves as a white box for users and developers to understand how the modules are composed.
The major purpose of this document is to provide a detailed information for the concerned parties about how and why a certain implementation has been carried out. The potential readers of the document are as follows: New project members: If new project members are added to the project then this document, along with the document about the MIS implementation, would help the new members understand how and why the functionalities have been implemented. It will also help them understand the features that must be preserved. This document provides the designers with a means of communication about the module specifications. It also helps determine if the

requirements have been met. It can also show the flexibility and feasibility of various modules. It is important for the people responsible for maintaining the modules to understand the hierarchical structure of the modules. This document helps people responsible for updating this project to understand the way the implementation has been done for the project.

1.3 Design Decision

The design for this project FaultInOurPong follows the following rules:

1. MVC model: MVC model has been implemented in rigorously in the project. The design has been separated in model, view and control frameworks. The model class is responsible for managing the data, logic and rules of application of the project. The view is responsible for the output representation of the information. The controller is responsible for the implementations of commands from users and manipulates the model.

2. Each data structure is implemented in only one model.And they would be exported to another modules for interactions.

3. The implementations that are likely to change are stored in separate modules.

4. The concepts of separation of concern (SC), information hiding (IH), and abstractions are used to further organize the code.

1.4 Document Structure

The rest of the document is arranged as follows: Section 2 provides details about anticipated and unlikely changes of the project. Anticipated changes are listed in Section 2.1, and unlikely changes are listed in Section 2.2. Section 3 contains the breakdown of the module hierarchy, per the likely changes. Section 4 shows the connections between the software requirements and the modules. Section 5 shows a detailed breakdown of the module description. Section 6 includes the tractability matrix. Section 7 describes the use hierarchy between various modules.

1.5 Acronyms and Definitions

Table 2: **Acronyms**

Acronym	Definition
AC	Anticipated Change
DAG	Directed Acyclic Graph
FR	Functional Requirement
IH	Information Hiding
MG	Module Guide
MIS	Module Interface Specification
NFR	Non-Functional Requirement
SRS	Software Requirements Specification
UC	Unlikely Change

Table 3: **Definitions**

Term	Definition
Welcome page	The first window shown on the screen when the program starts

2 Anticipated and Unlikely Changes

All the possible changes in listed in the first secion 2.1, and the unlikely changes are listed in the second section 2.2.

2.1 Anticipated Changes

AC1: The specific hardware on which the game is running.

AC2: The format of the input data. (left and right keys can be changed to different keys inside the GameController class without it affecting the rest of the project)

AC3: The constraints on the input parameters.

AC4: Game features. (Number of people added on the highscores list, number of lives given to the user)

AC5: Additional features. (Advanced single player mode with obstacles added, different speeds of the ball)

AC6: Magnitude of game controls and media (size of the buttons, ball etc.).

2.2 Unlikely Changes

UC1: Input and output devices. (Input: mouse clicks and keyboard presses, Output: screen/console)

UC2: There will always be a source of input data external to the software.

UC3: Game mechanics. (Formulas to calculate when ball should change direction)

UC4: Execution environment. (Must be java-based)

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

Level 1	Level 2
Hardware-Hiding Module	PongGame M1
Behaviour-Hiding Module	Ball M2 GameModel M3 Paddle M4 Player M5 GameView M6 HighScore M7 Mode M8 PongGameDisplay M9 Tutorial M10 Welcome M11
Software Decision Module	GameController M12

Table 4: Module Hierarchy

4 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 5.

5 Module Decomposition

5.1 Hardware Hiding Modules

Name: PongGame M1

Secrets: This module starts running in Java Development Environment.

Services: This module invokes the computer to display game windows.

Implemented By: Windows

5.2 Behavior-Hiding Module

5.2.1 Ball M2

Secrets: This data structure stores all the information of a ball.

Services: This module contains all the operations for a ball, including its postions, and size.

Implemented By: Ball.java

5.2.2 GameModel M3

Secrets: This model is the interface of the model framework in the MVC design pattern.

Services: It coroperates other data models such that other models can interact with the controller framework.

Implemented By: GameModel.java

5.2.3 Paddle M4

Secrets: This data structure stores all the information of a paddle.

Services: This module contains all the operations for a paddle, including its postions, height, and width.

Implemented By: Paddle.java

5.2.4 Player M5

Secrets: This data structure stores all the information of a player.

Services: This module contains all the operations for a player, including his/her score and methods to increase/decrease score.

Implemented By: Player.java

5.2.5 GameView M6

Secrets: This module acts as an interface for all other view modules.

Services: This module cooperates with other view modules such that they can interact with the controller framework.

Implemented By: GameView.java

5.2.6 HighScore M7

Secrets: This module displays the high scores of a player from a text file.

Services: It sets up a window/frame for display the scores for top 20 players.

Implemented By: HighScore.java

5.2.7 Mode M8

Secrets: This module displays different game modes for the player.

Services: It sets up a window/frame with buttons for the user to choose differet game modes.

Implemented By: Mode.java

5.2.8 PongGameDisplay M9

Secrets: This module displays the actual game panel.

Services: It sets up the game panel by drawing objects such as paddles, ball, and current scores on the screen.

Implemented By: PongGameDisplay.java

5.2.9 Tutorial M10

Secrets: This module displays the game instruction.

Services: It sets up the window for the tutorial page by displaying a picture of the instruction

Implemented By: Tutorial.java

5.2.10 Welcome M11

Secrets: This module displays the first window when the program starts.

Services: It creates different buttons for options so that a user can choose an option from it and start the game

Implemented By: Welcome.java

5.3 Software Decision Module

5.3.1 GameController M12

Secrets: This module contains part of the logic of the game.

Services: It performs some calculations to determine the winning/losing of a player; it also takes in hardware/environment variables and pass them into model and view framework.

Implemented By: Ball.java

6 Traceability Matrix

Requirements	Modules
R1	M1, M11, M6, M12
R2	M1, M11, M6, M12
R3	M5, M6, M12
R4	M1, M7, M11, M12
R5	M1, M6, M10, M12
R6	M8, M12
R7	M8, M12
R8	M3, M5
R9	M3, M5
R10	M4, M9, M12
R11	M5, M12
R12	M6, M12
R13	M6, M12
R14	M6, M12
R15	M6, M12
R16	M11, M11
R17	M7, M12
R18	M6, M12

Table 5: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M12
AC3	M12
AC4	M5, M7
AC5	M2, M6, M8, M12
AC6	M2, M4, M11

Table 6: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

User hierarchy can be depicted below Figure 1 in the graph. The modules listed in the document form a directed acyclic graph (DAG).

Figure 1: **Use Hierarchy**