SE 3XA3 Module Guide: Revision 0

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

November 14, 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

1 Introduction

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. Design 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 classes. The model class is responsible for managing the data, logic and rules of application of the project. View is responsible for the output representation of the information. Control is responsible for the implementations of commands from users and manipulates the model.

2. Each data structure is implemented in only one model.

3. If any other program requires the data structure implemented in a module, it calls on that particular module to access the data.

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

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.

Designers: 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. Maintainers: 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. The rest of the document is arranged as follows. The second section (2.1 and 2.2) of this document provides details about anticipated and unlikely changes of the document. The third section contains the breakdown of the module hierarchy, per the likely changes. The forth section shows the connections between the software requirements and the modules. The fifth section shows a detailed breakdown of the module description. The sixth section includes the tractability matrix. The seventh section describes the use hierarchy between various modules.

2 Anticipated and Unlikely Changes

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.

M1 Hardware hiding modules

M2 Input control module

M3 Output control module

M4 Game frame control module

M5 Player details controlling module

M1 is not a required module in our project, because the implementation is software based.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Paddle control module Ball control module Start and end of game control module Input control module Output control module High score controlling module
Software Decision Module	Game frame controlling module Player details controlling module

Table 2: 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 3.

5 Module Decomposition

5.1 Hardware Hiding Modules (M1)

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: Windows

5.2 Behavior-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: -

5.2.1 Input Control Module M2

Secrets: The structure and forms of the input data.

Services: Converts the input data provided by the user into a command for the movement and other optional services within the game.

Implemented By: The classes within the model folder that takes in account of the various stages and input control factors of the game, such as various speed levels.

5.2.2 Output Control Module M3

Secrets: The structure and forms of the output data.

Services: Based on the input data provided by the user it controls the movement and other services, based on users' choice for the game. This module is used to hide the implementation of the various outputs that the users get during the game.

Implemented By: The classes within the view folder that takes in account of the various output of the game, such as the movement of the paddle in the game frame based on the keyboard keys movement by the user.

5.3 Software Decision Module

5.3.1 Game Frame Control Module M4

Secrets: The structure and format of the game frame control.

Services: This module controls the outer frame of the ping pong game, such as the frame size, the switching between one frame to another etc.

Implemented By: The game view class within the view folder.

5.3.2 Player Details Controlling Module M5

Secrets: The structure and format of the player object implementation.

Services: This module controls the information and details about each user and provides the user with the services and details associated with the player class while they are playing the game.

Implemented By: The player class within the model folder.

6 Traceability Matrix

Requirements	**Modules**
R1	M2, M4, M5
R2	M1, M2, M3, M4, M5
R3	M1, M2, M4, M5
R4	M1, M3, M5, M6
R5	M2, M3, M5
R6	M2, M3, M4, M5, M6
R7	M2, M3, M4, M5, M6
R9	M1, M3, M4, M5
R10	M1, M4, M6
R11	M1, M3, M4, M6
R12	M1, M2, M3, M5
R13	M1, M3, M4, M6
R14	M1, M3, M4, M6
R15	M1, M4, M5
R16	M1, M4, M5
R17	M1, M2, M3, M4, M5
R18	M1, M3, M4, M6
R20	M1, M4, M5
R21	M1, M2, M4, M5, M6
R22	M3, M4, M5

Table 3: Trace Between Requirements and Modules

AC	**Modules**
AC1	M1
AC2	M2, M4
AC3	M4, M6
AC4	M4, M6
AC5	M2, M3, M4
AC6	M3, M4, M5

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

GameModel.java class uses ball.java, paddle.java and player.java.

GameView.java uses welcome.java, mode.java, pongGameDisplay.java and tutorial.java.

Tutorial.java uses ImageIcon.img.

GameController.java uses GameView.java, GameModel.java, Welcome.java, mode.java, highScore.java and Tutorial.java.

StartGame.java uses GameView.java, GameModel.java and GameController.java.