SE 3XA3: Software Requirements Specification
Title of Project

Team #204, Trident Inc.
Harshil Modi, modih1
Justin Prez, prezj
Justin Rosner, rosnej1

April 5, 2020

Contents

List of Tables

List of Figures

Table 1: Revision History

Date	Developer(s)	Change
2020-03-09	Justin Rosner, Justin Prez, Harshil Modi	Initial write-up of MG document
2020-04-05	Justin Rosner, Justin Prez, Harshil Modi	Revision 1 of MG document

1 Introduction

1.1 Overview

Mastermind is a famous 1970's Mordecai Meirowitz game implemented as an android application ~~implemented~~ using Dart and Flutter. The goal of our project is the bring Mastermind back as the game that can be enjoyed by people of all ages. By developing an app version of the game, people will be able to play the game in their free time, or on-the-go. The Mastermind development team have added numerous requirements that the open source version failed to address, as seen in the SRS. The MG aims to separate these requirements into modules of similar functionality. The primary focus of the team is code modularization, maintainability, interoperability, and robustness. For a breakdown of the modules the team has deemed important, see the Module Interface Specification.

1.2 Scope

The scope of the project is to create a more interactive implementation of the Mastermind game. As the rules of the game of Mastermind are fairly straightforward, the development team will have no issues recreating the game in its entirety. The app will be developed with Dart/Flutter, allowing for deployment on both iOS and Android platforms through the Apple Store and Google Play Store.

1.3 Module Guide Purpose

When developing a mobile game, there are numerous components that fit together to form the final product. The purpose of the Module Guide is to provide an overview of how the team decided to decompose the components of Mastermind and provide an explanation on how the components interact with each other. Furthermore, the module guide will be used by the design team to ensure the consistency between the implementation and specification entitled through the documentation. Finally, the Module guide will provide developers with a way of understanding the implementation structure of Mastermind.

2 Anticipated and Unlikely Changes

This section of the Module Guide outlines possible changes to the design that the development team has considered but not yet implemented. These changes are broken down into two categories; Anticipated changes (2.1) that the development team will start to work on in the near future, and Unlikely Changes (2.2) that are more theory based and not realistic for the development team.

2.1 Anticipated Changes

AC1: The specific hardware on which the software is running (i.e. a new phone or tablet is released).

AC2: The operating system that the software runs on.

AC3: The format and layout of the game board.

AC4: The format and layout of the game menus.

AC5: The appearance of the buttons.

2.2 Unlikely Changes

UC1: Input devices. Given the rise of virtual and augmented reality in the last couple of years, it is possible that the team may wish to make this the focus of the user's interactions with Mastermind. Given the scope and the time required to make this change, it is unlikely that the development team will choose to replace the touch screen based interaction system currently in place.

UC2: Changes to the rules of the game. Changing the rules of the game would cause for changes to occur in the game board module, menu module, and both the game logic controller and the game state controller. Due to changes occurring in more than one module, it becomes a very unlikely change for the development team to make.

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: Android Emulator

M2: Game Board Module

M3: Menu Module

M4: Button Module

M5: Game Board Controller

M6: Types Module

Level 1	Level 2
Hardware-Hiding Module	Android Emulator
Behaviour-Hiding Module	Game Board Module Menu Module Button Module Types Module
Software Decision Module	Game Board Controller

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

This application was designed to satisfy all the requirements outlined in the SRS Document. In this stage, Mastermind is decomposed into modules to fulfill the functional requirements, such as displaying the game board on an Android/iOS device. The connection between requirements and modules is listed in Table 3.

5 Module Decomposition

Module decomposition is critical for improving the principle of "information hiding". Information hiding significantly reduces the complexity, while improving the maintainability and scalability of the program. Each module below contains fields for *Secrets*, *Services* and *Implemented By*. The *Secrets* field describes the design decisions hidden by the module. The *Services* field details what the module will do not documenting how to do it. The *Implemented By* field suggests the implementing software.

5.1 Hardware Hiding Module: Android Emulator (M1)

Secrets: The data structures and algorithms used to implement the virtual android emulator, track and manipulate the windows displayed to the screen.

Services: Serves as virtual hardware to emulate an android smartphone, able to run applications, accept user input and display output to the screen.

Implemented By: *OS*

5.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours for the Game Board, Menu, Button and Types Modules.

Services: The programs included affect the visible appearance of the game as described in the software requirements specification (SRS) documents. These modules serve as a communication layer between the hardware-hiding module and the software decision module.

Implemented By: Mastermind

5.2.1 Game Board Module (M2)

Secrets: The functions and state variables used to create an instance of the game board and draw it to the screen.

Services: Draws the game board to the screen, and updates the visible game board based on user input.

Implemented By: Mastermind

5.2.2 Menu Module (M3)

Secrets: The functions and state variables used to create the menu screen.

Services: Accepts user input for selections on the menu screen.

Implemented By: Mastermind

5.2.3 Button Module (M4)

Secrets: The functions and state variables used draw the button to the screen.

Services: Draws the buttons of different colours to the screen.

Implemented By: *OS*

5.2.4 Types Module (M6)

Secrets: ~~N/A.~~ The implementation of various types needed for the rest of the modules.

Services: The exported types from this module are to be used in other modules.

Implemented By: Mastermind

5.3 Software Decision Module

Secrets: Algorithms and data structures used for implementing the game rules and tracking the state of the Game Board.

Services: Includes data structure and algorithms used in to implement the game that the user does not directly interact with.

Implemented By: –

5.3.1 Game Board Controller (M5)

Secrets: Algorithm used to implement the rules of the game and logic of the game board.

Services: Includes algorithms that are used for interpreting user input, displaying correct output, and checking for win/lose conditions.

Implemented By: Mastermind

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
REQ1	M1
REQ2	M2, M4
REQ3	M2, M4
REQ4	M2, M4
REQ5	M2, M4
REQ6	M2, M4
REQ7	M2, M4
REQ8	M2, M5, M1
REQ9	M3, M1
REQ10	M5, M1
REQ11	M2, M5
REQ12	M2, M5
REQ13	M2, M5
REQ14	M2, M5

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M1
AC3	M2
AC4	M3
AC5	M4

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section, the uses hierarchy between the interacting modules is provided. It can be seen that the hierarchy represented below is an example of a directed acyclic graph (DAG). Meaning that if the Game Board Controller is dependent on the Game Board Module (and on a higher level in the hierarchy), then the Game Board Module cannot be dependent on the Game Board Controller. Modules in the higher level of the hierarchy have simpler implementations because they use 'secrets' from the lower levels.

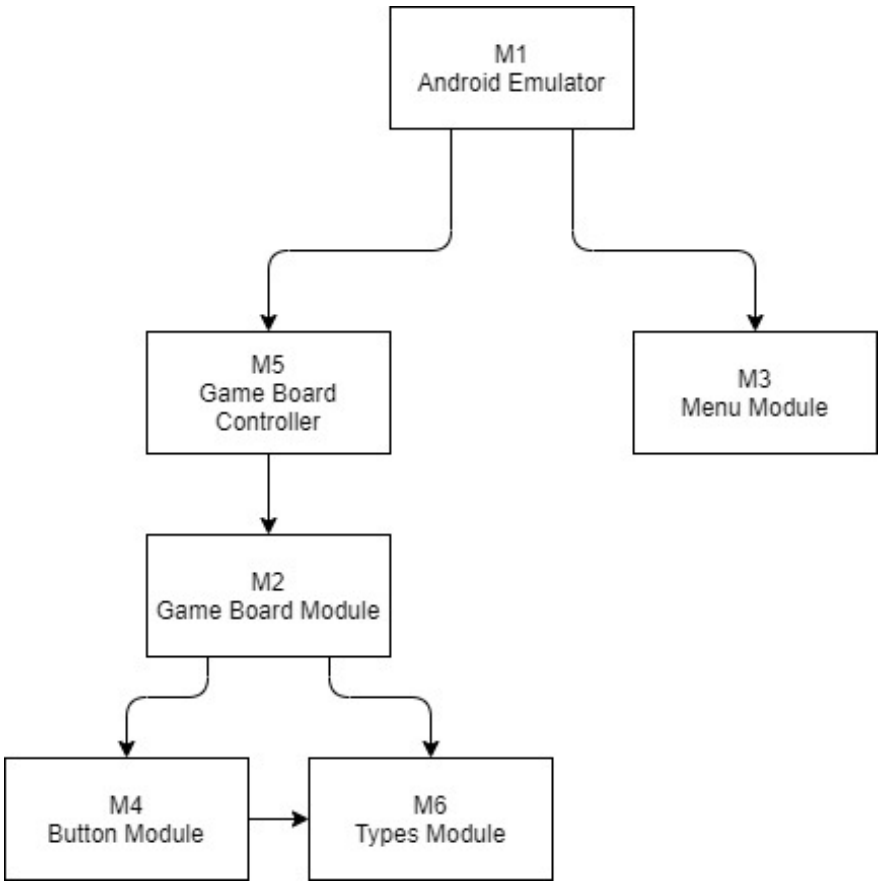

Figure 1: Use hierarchy among modules