SE 3XA3: Module Guide
KingMe

Team 9, KingMe
Ardhendu Barge 400066133
Dylan Smith 001314410
Thaneegan Chandrasekara 400022748

April 13, 2021

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
2021/03/10	0	Added the modules of the system
2021/03/11	0	Added likely and unlikely changes
2021/03/14	0	Added the traceability tables
2021/03/18	0	Added the section introducitons
2021/04/12	1	Updated Menu, Game and Piece Modules in Module Decomposition section
2021/04/12	1	Removed FR5, NFR3 from traceability table
2021/04/12	1	Updated UsesHierarchy image

1 Introduction

A very important part of the development of our checkers application, King Me, is modular decomposition. This is the process of breaking our application into individual modules each containing one secret. In this document we will be discussing each of these different modules, the secret they contain and how they have been designed for change. We will be discussing the changes we anticipate being made in the future and how the modules have been designed to allow these changes to be easily made, and we will discuss the unlikely changes that we don't anticipate being made. We will be discussing the structure of the modules and how they comprise the system. Lastly we will demonstrate how the requirements are being met by the different modules of our system.

2 Anticipated and Unlikely Changes

This section will list possible changes that may be made to the system. They will be separated into anticipated and unlikely changes.

2.1 Anticipated Changes

Anticipated changes are changes that were anticipated during the design process. These changes influenced the modular decomposition of our system. They are changes that we anticipate will likely be made and should be easily made. We chose to decompose the system into modules containing secrets such that each of these changes would result in the modification of only one module, therefore making it much easier to implement the changes when the time comes.

AC1: Modifying the algorithm for generating computer moves.

AC2: Changing the graphical elements used to display the application.

AC3: The tutorial will need to be updated when new features are added or removed.

AC4: Changes to the implementation of the state of the board.

AC5: Changing how the valid moves are generated.

AC6: More advanced checkers rules may be added.

2.2 Unlikely Changes

Unlikely changes are those that we do not anticipate being made. The module structure was not be designed to accommodate these changes. They will likely require the modification of multiple modules, making these changes much more difficult to implement.

UC1: The input will come from the mouse and the screen will be used to output to.

UC2: The user is the source of the input.

3 Module Hierarchy

In this section we discuss our module design. The modules are separated into Hardware-Hiding, Behaviour-Hiding and Software Decision Hiding based on the secrets they contain.

M1: GUI Module

M2: Menu Module

M3: Board Module

M4: Game Module

M5: Piece Module

M6: AI Module

Level 1	**Level 2**
Hardware-Hiding Module	None
Behaviour-Hiding Module	GUI Module Menu Module Board Module Game Module Piece Module
Software Decision Module	AI Module

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

King Me has been designed to meet the functional and nonfunctional requirements as outlined in the SRS document. Table 3 shows all the requirements of the SRS document alongside the modules used to satisfy them.

5 Module Decomposition

In this section we will discuss what modules our system have been decomposed into. We will introduce the different modules and briefly explain their role in the overall system. Each module is broken up into a secret, a service and an implementation. The secret of the module is the design decision that has been hidden within the module , the service of the module is what the module does, and the implementation is how we implemented the module.

5.1 Hardware Hiding Modules

N/A - This implementation contains no hardware hiding modules.

5.2 Behaviour-Hiding Modules

5.2.1 GUI Module (M1)

Secrets: Manages the display of the application.

Services: Displays the UI to the screen. Displays the board, the pieces and the menu options in the desired configurations. Captures interactions on the UI and relays to appropriate modules.

Implemented By: GUI.py (Python)

5.2.2 Menu Module (M2)

Secrets: ~~Manages the 'New Game' and 'Tutorial' logic.~~ Manages the logic for the Main Menu and the In-Game Menu.

Services: ~~Provides a way to select restart the game and also provides instructions on how to play the game.~~ Provides functionality for selecting game mode (1-player or 2-player), selecting piece color (red or white), restarting the game (new game) and also provides instructions on how to play the game (tutorial).

Implemented By: menu.py (Python)

5.2.3 Board Module (M3)

Secrets: Manages the state of the game board.

Services: Provides details on the state of the game board, including where pieces are located on the board.

Implemented By: board.py (Python)

5.2.4 Game Module (M4)

Secrets: ~~The checkers logic.~~ Manages user input to play the game.

Services: ~~Determines the valid moves that can be made by any given piece. Tracks whether or not a win/loss state has been reached.~~ Determines what happens when user input is received.

Implemented By: game.py (Python)

5.2.5 Piece Module (M5)

Secrets: ~~Manages how the position of the piece.~~ Manages the position and state of a piece.

Services: ~~Contains the information on where the piece is located and the direction(s) it can move.~~ Stores information about the piece's state. Provides functionality for moving a piece and for changing piece from 'man' to 'king'

Implemented By: piece.py (Python)

5.3 Software Decision Modules

5.3.1 AI Module (M6)

Secrets: The logic of how the AI/Computer chooses it's move.

Services: Calculates the best move for the AI/Computer to make based on the given state of the game board. Utilizes an efficient algorithm to select the move while satisfying the NFR response times.

Implemented By: minmax.py (Python)

6 Traceability Matrix

This section shows two traceability matrices. The first shows the connection between the requirements of the SRS document and the modules used to satisfy them, and the second is

between the anticipated changes and the modules that will need to be modified to make the changes.

Req.	**Modules**
FR1	M1
FR2	M1, M2
FR3	M5
FR4	M5
~~FR5~~	~~*M1, M5*~~
FR6	M1, M3, M4
FR7	M1, M2
FR8	M3, M4
FR9	M3, M4
FR10	M1, M4
FR11	M5, M1, M3, M4
FR12	M1, M5
FR13	M3, M5
FR14	M3, M4, M5
FR15	M1, M4, M5
FR16	M1, M3, M5
FR17	M1, M4, M3
NFR1	M1, M2
NFR2	M1
~~NFR3~~	~~*M1*~~
NFR4	N/A
NFR5	M2
NFR6	M1, M3, M4, M5
NFR7	M1, M3, M6
NFR8	M1, M3, M4
NFR9	M1
NFR10	N/A
NFR11	N/A
NFR12	N/A
NFR13	N/A
NFR14	N/A
NFR15	N/A
NFR16	N/A
NFR17	M1, M2

Table 3: Trace Between Requirements and Modules

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

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section we provide the uses hierarchy between the modules. The uses hierarchy depicts the relationships between the different modules. An arrow from one module to another describes a uses relation where the first module depends on the module it points to as described by Parnas (1978).

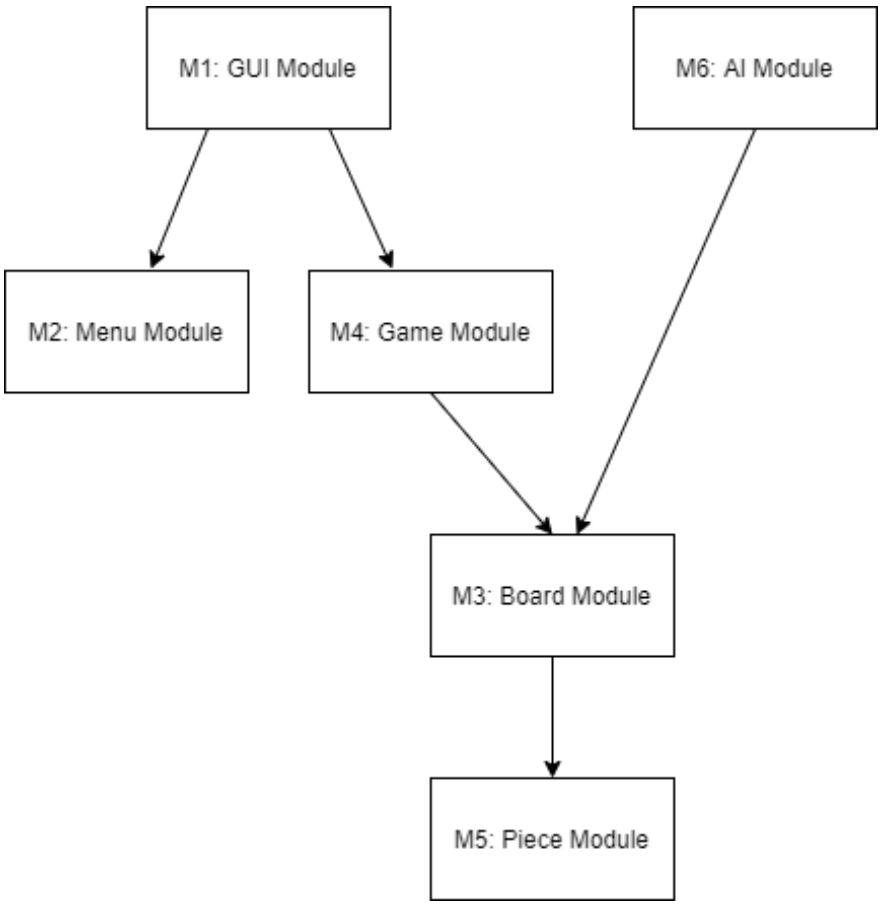

Figure 1: Use hierarchy among modules

References

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.