SE 3XA3: Module Guide Othello

Team #2, JMS Corporation
Mohammed Mirajkar, mirajkam
Jinesh Patel, patelj60
Sankar Renganathan, renganas

December 5, 2018

Contents

List of Tables

List of Figures

1 Introduction

JSM Corporation has implemented a web-based version of Othello that strives to provide a fresh new feel to playing the beloved board game. JSM Corporation has provided this module guide which contains information on the different modules that have been used to implement the game of Othello. The modules used in the project present a state of low coupling and high cohesion and this guide will run through how the project has been organized and how different functions interact with each other. This guide will touch on anticipated and unlikely changes, module decomposition, uses hierarchy and the tracability matrix to connect how each module depends on certain other modules.

2 Anticipated and Unlikely Changes

This section lists possible changes to the system to help implement unfinished tasks or to improve the game.

2.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The algorithm for moves made by the AI.

AC2: The animation time for valid moves .

AC3: The animation time for invalid moves.

AC4: The graphical user interface elements and format used to represent the game score.

AC5: The storage method used to save and load the game state.

Table 1: **Revision History**

Date	**Version**	**Notes**
06/11/2018	0.1	Worked on Introduction and Behaviour Modules
06/11/2018	0.1	Worked on Anticipated and Unlikely Changes as well as Module Hierachy
29/11/2018	1.0	Revised Document

2.2 Unlikely Changes

This sections lists changes that are not prioritized by the developers.

UC1: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

UC2: The input data will always come via the user.

UC3: The input format (will always be a click event).

UC4: The algorithms for determining a valid and invalid move.

UC5: The animations for making an invalid move and making a valid move.

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 Module

M2: Board Module

M3: UI Module

M4: Game Module

M5: Game History Module

M6: AI Module

Level 1	**Level 2**
None	
Behaviour-Hiding Module	Board Module Module UI Module Module Game Module Game History Module
Software Decision Module	AI 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

This section depicts how the project is organized into different modules and the main functionalities of the modules and how it contributes to the overall system

5.1 Hardware Hiding Modules (M1)

There is no hardware hiding module being implemented

5.2 Behaviour-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 Board Module (M2)

Secrets: Manages the state of the board

Services: Contains information of the state of the board and the validity of the board

Implemented By: HTML and JavaScript

5.2.2 UI Module (M3)

Secrets: How the game responds to user input

Services: Displays information and visuals in an organized manner to the user and reacts to input

Implemented By: HTML, JavaScript and CSS

5.2.3 Game Module (M4)

Secrets: How the game functions

Services: Calculates moves and score and checks the validity of the move

Implemented By: JavaScript

5.2.4 Game History Module (M5)

Secrets: Helps keep track of the past states of the game

Services: Stores the past states of the game board to be accessed later

Implemented By: JavaScript

5.3 Software Decision Module (M6)

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

5.3.1 AI Modules

Secrets: The AI Controller

Services: Chooses moves based on the state of the board in a time efficient manner

Implemented By: JavaScript

6 Traceability Matrix

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

AC	Modules
AC1	M6
AC2	M3
AC3	M3
AC4	M3
AC5	M4

Table 4: Trace Between Anticipated Changes and Modules

Req.	**Modules**
FR1	M3, M2
FR2	M3, M4
FR3	M3, M2
FR4	M4, M6
FR5	M3, M4, M2
FR6	M4, M3
FR7	M4, M2, M3
FR8	M4, M2, M5
FR9	M4, M2, M5
NF1	M3
NF2	M3
NF3	M3
NF4	M3
NF5	M3
NF6	M3
NF7	M3
NF8	M3, M6
NF9	M3, M4, M2
NF10	M3, M4, M2
NF11	M4
NF12	M4
NF13	N/A
NF14	M5
NF15	M5
NF16	N/A

Table 3: Trace Between Requirements and Modules

7 Use Hierarchy Between Modules

This section shows the relationship of each module, identifying leaf modules as well as the heirarchy of the modules. A DAG is used to portray this in order to show the which module use other modules and which modules are used by other modules.

This references Parnas (1978), and Figure 1

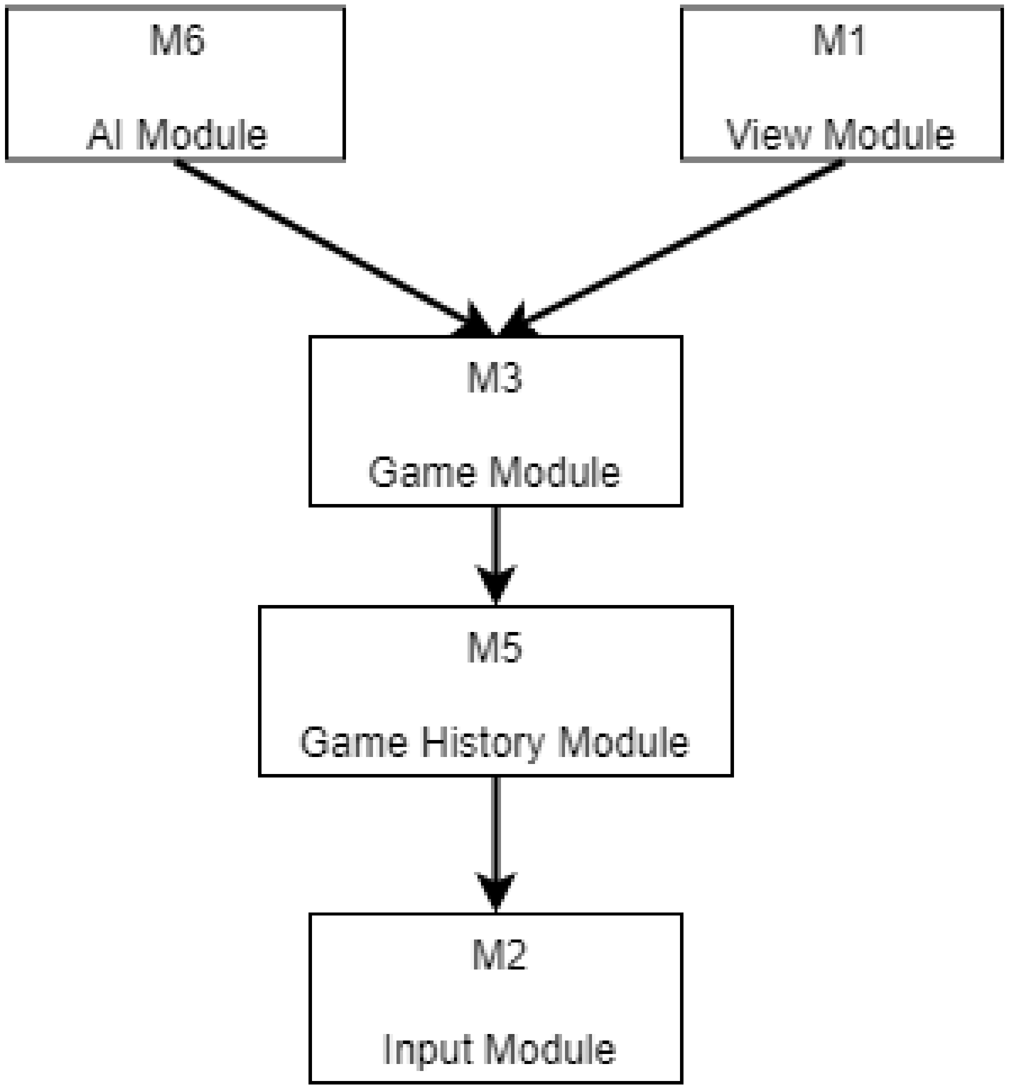

Figure 1: Use hierarchy among modules

8 Project Schedule

The project schedule is layed out by this Gantt Chart created by JSM Corp.

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.