SE 3XA3: Module Guide
Group 309: 2048

Yanting, Cao	Chris, Fu	Shengchen, Zhou
400067576, caoy31	400084178, fus13	400050783, zhous20

March 13, 2020

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
2020.3.10	0	draft discussion and edition
2020.3.13	0	revised ideas and plans with consideration

1 Introduction

1.1 Overview

This is the module guide and design document for re-development of 2048 game project by group 309. This document is written for potential readers listed below:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

1.2 Context

The module guide(MG) is based on software requirement specification(SRS) document. With the purpose of develop project's components concurrently, a decomposition of system will be implemented and divide into different modules. Generally, the MG is supposed to:

- provide the view how system meets functional and non-functional requirements(FRs & NFRs)
- specify modular structure of system and allow designers and maintainers to easily identify parts of software.
- provide support and reference to module interface specification(MIS)
- make the modification of modules in the system easier

1.3 Document structure

The document structure is constructed below:

- Section 2 lists anticipated and unlikely changes to system implementation.
- Section 3 includes details of module hierarchy
- Section 4 includes explanation of connection between requirements and deisgn

- Section 5 introduces how the modules are decomposed based on the principle of information hiding to support high cohesion and low coupling.
- Section 6 includes traceability matrices to check the completeness of requirements in SRS.
- Section 7 includes use hierarchy between modules via a graph of modules' relationship

2 Anticipated and Unlikely Changes

2.1 Anticipated Changes

AC1: The format of user input for additional features in game.
AC2: The implementation of data structure that stores information of grid.
AC3: The module of user interface for game mode.
AC4: The format of output data.(score, record)
AC5: The graphical user interface elements to capture user input.
AC6: The ways to store user's record.

2.2 Unlikely Changes

UC1: Input/output devices of system, assumed via mouse, keyboard and screen.
UC2: The storage of output data(assumes that can store on local machine)
UC3: The operations on data structure that stores tile information, i.e. movement, collision.
UC4: The condition to check game stage - victory/loss

3 Module Hierarchy

See below in table 2 and 3.

4 Connection Between Requirements and Design

The system is designed to meet all functional requirements and non-functional requirements. The connection between rquirements and modules is listed in table 4 and 5.

5 Module Decomposition

Modules are decomposed according to the principle of information hiding proposed by David Parnas. The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without

Level 1	Level 2
Hardware Hiding Module	
Behaviour Hiding Module	Keyboard input module Game manager module Local storage module HTML actuator Module User interface Module
Software Decision Module	Grid module Tile module

Table 2: Module Hierarchy

Module Name	Module Number
Hardware Hiding Module	M1
Keyboard input module	M2
Game manager module	M3
Local storage module	M4
HTML actuator module	M5
Grid module	M6
Tile module	M7
User interface module	M8

Table 3: Module Number Format

documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title.

5.1 Hardware Hiding Module

Secrets: The implementation of JavaScript in OS.

Services: This module serves as the interface between software and hardware, which allows the system to take inputs and display outputs.

Implemented By: OS

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.

Implemented By: N/A

5.2.1 Keyboard input module

Secrets: User inputs

Services: Converts the user input into the operations and parameters in data structure inside system during user's interaction with system.

Implemented By: [keyboard_input_manager.js]

5.2.2 Game manager module

Secrets: Game interface design.

Services: Initiate the game interface and

Implemented By: [game_manager.js]

5.2.3 Local storage module

Secrets: Local information

Services: Stores records and game state information.

Implemented By: [local_storage_manager.js]

5.2.4 HTML actuator module

Secrets: HTML elements

Services: present information from other module to user and game stage.

Implemented By: [html_actuator.js]

5.2.5 User interface module

Secrets: user interface design

Services: present actual interface to user

Implemented By: [main.css]

5.3 Software Decision Module

Secrets: Data structures

Services: Stores the data structures and information which are used in system design.

Implemented By: N/A

5.3.1 Grid Module

Secrets: Grid

Services: Stores the information of grid and operations.

Implemented By: [grid.js]

5.3.2 Tile Module

Secrets: Tile

Services: Stores the information of tile and operations.

Implemented By: [tile.js]

6 Traceability Matrix

6.1 Modules and Requirements

see below in table 4.

Requirement	Modules
Functional Requirements	
FR1-2	M6
FR3-9	M7, M3
FR10-14	M3
FR15-19	M3
FR20-22	M5, M2
FR23-25	M5, M6
FR26-27	M4
FR27-31	M5
FR32-34	M4
Non-functional Requirements	
NF3.1.1	M8
NF3.1.2	M8
NF3.2.1	M8
NF3.2.2	M8
NF3.2.3	N/A
NF3.2.4	M8
NF3.2.5	M1,M2,M3,M4,M5,M6,M7,M8
NF3.3.1	M8, M3
NF3.3.2	N/A
NF3.3.3	M1,M2,M3,M4,M5,M6,M7,M8
NF3.3.4	M1,M2,M3,M4,M5,M6,M7,M8
NF3.3.5	N/A
NF3.3.6	M2,M3,M4,M5,M8
NF3.3.7	N/A
NF3.3.8	M2,M3,M4,M5,M6,M7,M8
NF3.4.1	M3,M5,M8
NF3.4.2	M3,M5,M8
NF3.4.3	M8
NF3.4.4	M2,M3,M4,M5,M6,M7,M8
NF3.5.1	M2,M3,M4,M5,M6,M7,M8
NF3.5.2	M2,M3,M4,M5,M6,M7,M8
NF3.5.3	M2,M3,M4,M5,M6,M7,M8
NF3.6.1	M2,M3,M4,M5,M6,M7,M8
NF3.6.2	M2
NF3.6.3	M2, M4
NF3.8	M8

Table 4: Trace Between Requirements and Modules

Anticipated Changes	Modules
AC1	M2,M3,M5,M8
AC2	M3,M6
AC3	M8
AC4	M5
AC5	M5,M8
AC6	M4

Table 5: Trace Between Anticipated changes and Modules

6.2 Modules and Anticipated Changes

see below in table 5.

7 Use Hierarchy Between Modules

See below in figure 1.

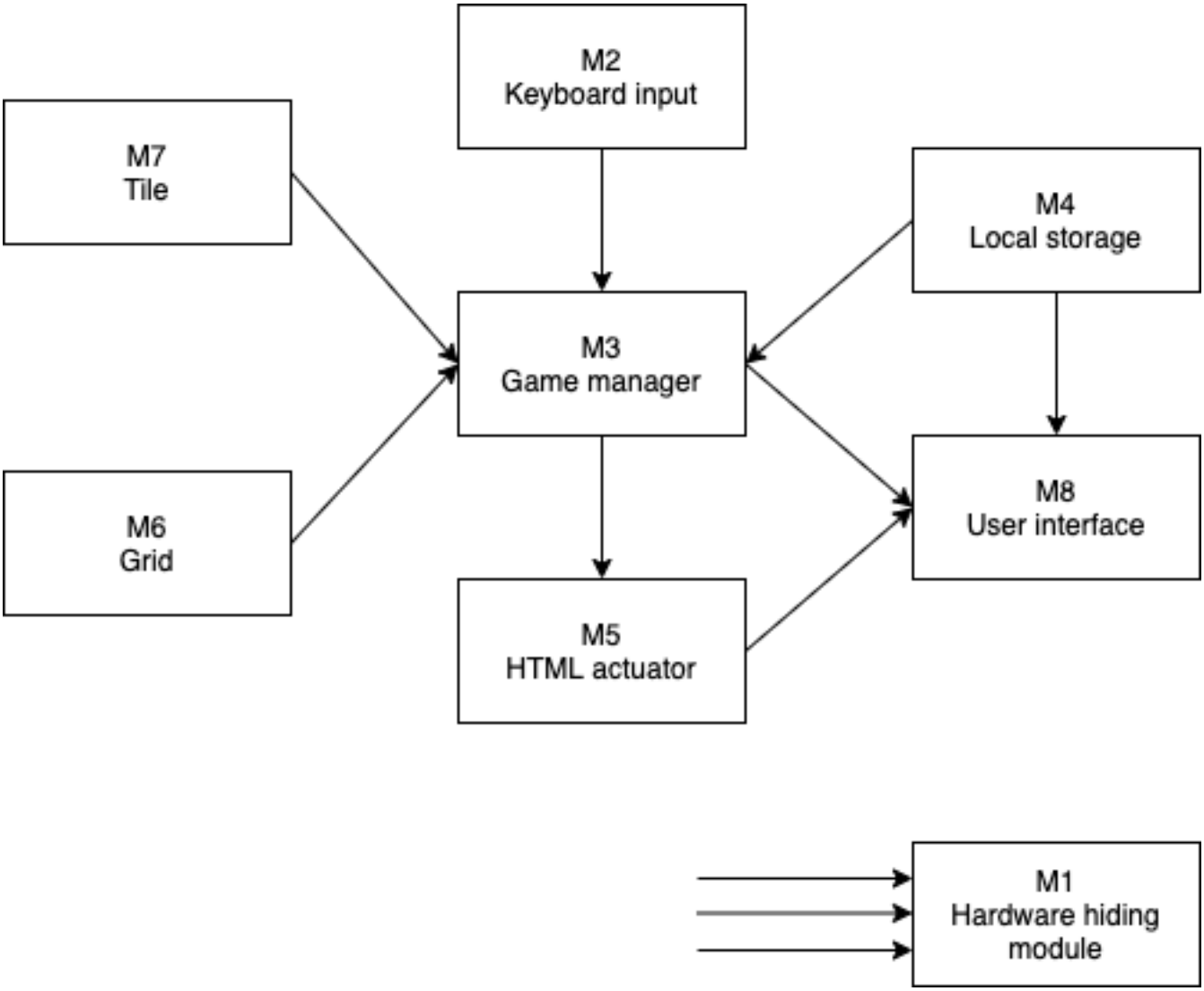

Figure 1: Use Hierarchy between modules