McMaster University

Software Project Management

SFWR ENG 3XA3

Design Document

Authors:
Mohammad Naveed **1332196**
Josh Voskamp **1319352**
Stephan Arulthasan **1308004**

November 4, 2015

Contents

List of Tables

List of Figures

Revision History

Rev. No.	Rev. Date	Description	Author
0	Nov 2 2015	Created Document	Mohammad Naveed
0	Nov 2 2015	Added Module Hierarchy	Josh Voskamp
0	Nov 4 2015	Added Module Decomposition	Stephan Arulthasan
0	Nov 4 2015	Added Introduction	Mohammad Naveed
0	Nov 4 2015	Improved Module Hierarchy	Josh Voskamp
0	Nov 4 2015	Added Anticipated Changes	Mohammad Naveed

Table 1: Revision History

1   Introduction

According to Jane McGonigal, a well known and world renowned game designer; we spend 3 billion hours a week playing video games. That is a lot of time that many people argue could be spent better, and that is what 2048 aims to accomplish. More and more people are playing video games everyday and 2048 is a fun and challenging game that tests the users? mathematical as well as their spatial intelligence. This allows 2048 to be fun, yet still be brain enhancing. Since the target audience for this game is so large, we can take advantage of this by providing users an option to spend their gaming time in a way thats beneficial mentally while still being entertained.

The design pattern that will be used to implement 2048 is the Model View Controller (MVC) design pattern. This pattern is based on the decomposition of the software into three different modules. The decomposition is based on the concept of information hiding. The model, view and controller modules. Each module has a specific task that it needs to focus on, for example, the view module focuses on the GUI of the game whereas the model module focuses on the data structures and actual data used in the game.

In terms of other documentation, after completing the Software Requirements Specification (SRS), the first part of the design document is completed which is then followed by the development of the Module Interface Specification (MIS). The MIS specifies the externally observable behaviour of a module's access routines.

The rest of the design document is organized as follows, section 2 lists the anticipated and unlikely changes of the software requirements. Section 3 lists the module hierarchy that was constructed. Section 4 lists the connection between requirements and design. Section 5 includes two traceability matrices that check the completeness of the requirements provided in the SRS. Section 6 gives a detailed description of the modules. Section 7 describes the use relation between modules.

2 Anticipated and Unlikely Changes

2.1 Anticipated Changes

2.2 Unlikely Changes

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module
M2: Keyboard
M3: GameView
M4: Main
M5: Board
M6: Tile

Level 1	Level 2	Level 3
Hardware-Hiding Module		
Behaviour-Hiding Module	Keyboard GameView Main	
Software Decision Module	Board	Tile

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

Modules are decomposed according to the principle of "information hiding". Each module hides some design decision from the rest of the system. This is described in the *Secrets* field. The *Services* field specifies *what* the module will do without documenting *how* to do it. Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented. Whether or not this module is implemented depends on the programming language selected.

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: OS

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

Table 3: Trace Between Requirements and Modules

7   Use Hierarchy Between Modules

AC	Modules

Table 4: Trace Between Anticipated Changes and Modules