Zop Design Documentation

Ratna Emani, Cole Blanchard, Akshay Mantha

December 7, 2015

Revision History

Developer	**Date**	**Change**	**Revision Number**
Cole Blanchard	December 07 2015	Introduction update, secrets are nouns, one secret per module.	4
Cole Blanchard	November 06 2015	Introduction, Anticipated and Unlikely Changes, Connection Between Requirements and Design, Pert Chart, Traceability Matrices	3
Ratna Emani	November 06 2015	Module Hierarchy, Module Decomposition, Use Hierarchy, Gantt Chart	2
Akshay Mantha	November 04 2015	MIS, Pert Chart	1

Contents

1 Introduction

This document details the design specifications for the implemented modules in Zop. Again, this document is to be used for Zop, the game where 2 the goal is to match as many tiles as possible within a minute. This document is to be used to simplify navigation through the program for design and maintenance purposes. This document complements the System Requirement Specifications, the Test Plan, and the provided MIS. In regards, to design principles we used the Model View Controller design to implement the Zop game.

2 Anticipated and Unlikely Changes

2.1 Anticipated Changes

AC1: Add a GUI.

AC2: Add a score counter.

AC3: Add a timer.

AC4: Add a mouse listener.

AC5: Change how the tiles are selected in a turn

AC6: Eliminate keyboard functionality

AC7: Change the algorithm of how turns are made in the game.

2.2 Unlikely Changes

UC1: Add a High Scores List.

UC2: Start, restart, play again buttons.

UC3: tile data stored differently.

3 Module Hierarchy

Module Hierarchy provides the guidelines of the module design. Modules are sorted in a hierarchy decomposed by secrets in Table **??**. The modules referred below are the modules that are being implemented and serve as the 'Leaf-nodes' in the hierarchy tree.

M1: Hardware-Hiding Module

M2: Main Module

M3: userInput Module

M4: Board Module

M5: Tile Module

M6: colourMatch Module

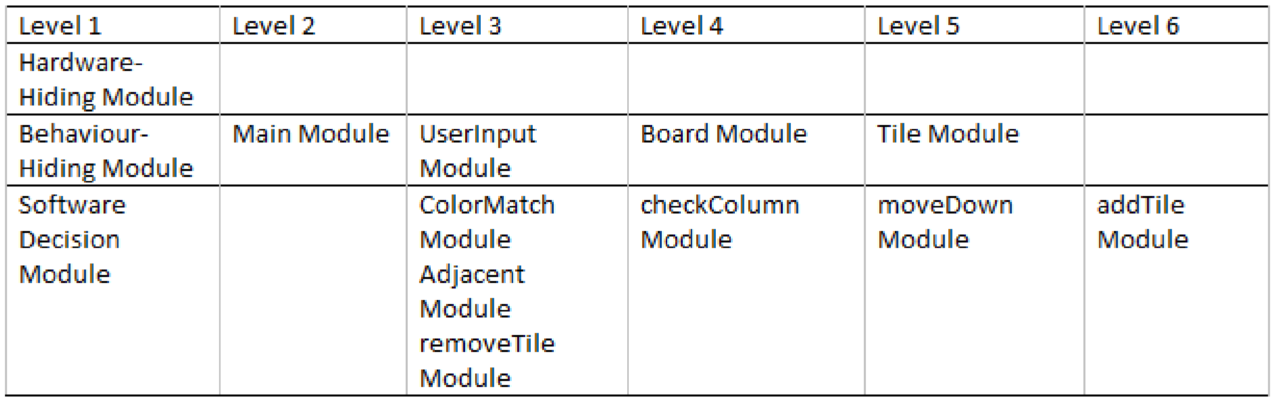
M7: adjacent Module

M8: removeTile Module

M9: checkColumn Module

M10: moveDown Module

M11: addTile Module



Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
Hardware-Hiding Module					
Behaviour-Hiding Module	Main Module	UserInput Module	Board Module	Tile Module	
Software Decision Module		ColorMatch Module Adjacent Module removeTile Module	checkColumn Module	moveDown Module	addTile Module

Table 1: Modules

The Operating System (OS) must implement some modules in order to perform other functions. M1 is a commonly used to refer to modules that have been implemented by the OS. Therefore there is no requirement for them to be reimplemented. All the other modules listen below serve to provide an input/output or make logical decisions to implement the rules of Zop.

4 Connection Between Requirements and Design

1. The product must allow the user to select game pieces. This requirement is satisfied in the userInput function of the design where the user inputs the coordinates of the piece they want to select.

2. The product must delete selected pieces of the same colour. This requirement is implemented with the colourMatch function in the design.

3. The product must delete selected pieces the are adjacent to each other. This requirement is met in the adjacent function in the design.
4. The design fulfils non functional requirements such as security (no password required), cultural, and legal requirements.
5. Look and feel requirement: zop consists of coloured squares on a grid. Although we implemented a visual "board" in the console, we have not met this requirement yet. This is an anticipated change in the future when the GUI is implemented.
6. Many requirement have not been yet implemented into the design, however these are anticipated changes that will be made in the future.

5 Module Decomposition

5.1 Hardware Hiding Modules (M1)

Secrets: the primary secret hidden by this module is the hardware/software interfaces. ~~Also the data structures and algorithms used to implement the virtual hardware~~
Service: controls the procedures that are used to handle the hardware/software interface changes with the same general capabilities. This is achieved by using virtual hardware, a digital hardware/software connection it can rely on.
Implemented By: OS

5.2 Main Module (M2)

Secrets: ~~Hides everything from~~ the game mechanics ~~to the sensitive design decisions~~.
Service: Is used to execute the game, called by the user or the front-end implementation when a GUI is developed.
Implemented By: -

5.3 userInput Module (M3)

Secrets: ~~Hides all~~ the software decision modules.
Service: serves as the only gateway in the behaviour-hiding module to use the software decision modules, reinforcing encapsulation and information hiding.
Implemented By: Main Module

5.4 Board Module (M4)

Secrets: ~~Hides~~ the Tile object ~~from the rest of the game logic~~.
Service: creates a six index by six index two-dimensional array, and populates the fields with the Tile object.
Implemented By: Main Module

5.5 Tile Module (M5)

Secrets: ~~generated randomly~~ random tiles
Service: creates a tile object with a random selection made from a set array of colors.
Implemented By: Board Module

5.6 colorMatch Module (M6)

Secrets: ~~hides part of~~ the data Ignores the index location
Service: Given a set of Tile objects, the module is responsible for checking the colors of the tile and confirm if they are all same.
Implemented By: userInput Module

5.7 adjacent Module (M7)

Secrets: ~~hides part of~~ the data Ignores the index value
Service: Given a set of Tile objects, the module is responsible for checking if the tiles are all adjacent to each other horizontal and/or vertical (not diagonal).
Implemented By: userInput Module

5.8 removeTile Module (M8)

Secrets: ~~secures the rest of~~ the board tiles ~~to avoid any loss of data~~
Service: Isolates the selected tiles from the remaining pieces to keep track and update the states of the game board. Finally marks the removed tile with a flag.
Implemented By: userInput Module

5.9 checkColumn Module (M9)

Secrets: ~~secures the order of the rest of~~ the columns on the board ~~to avoid any loss of data~~
Service: Isolates the selected columns and counts the number of flagged tiles and removes them.
Implemented By: removeTile Module

5.10 moveDown Module (M10)

Secrets: ~~works only with~~ the columns being updated. ~~hiding the rest of the board for being changed~~
Service: with the given column number and the number of missing tiles, moves all the tiles down. Implements the law of gravity within the board.
Implemented By: checkColumn Module

5.11 addTile Module (M11)

Secrets: ~~accesses only the very~~ top row of the matrix. ~~hiding the rest of the board from being changed~~
Service: with the given column number and the number of missing tiles, addTile fills the missing fields with new Tile objects.
Implemented By: moveDown Module

6 Traceability Matrix

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

6.1 Requirements and modules

Req.	**Modules**
Select game pieces	M3
Display Board	M4
Same Colour Tiles Removed	M6, M8, M11, M9
Adjacent Tiles Removed	M7, M8, M11, M9
Start Game	M2

Table 2: Trace Between Requirements and Modules

6.2 Anticipated Changes and modules

AC	**Modules**
AC1	M4, M5
AC2	M3, M4
AC3	M3, M4
AC4	M4
AC5	M3
AC6	M4
AC7	M4

Table 3: Trace Between Anticipated Changes and Modules

7 Use Hierarchy

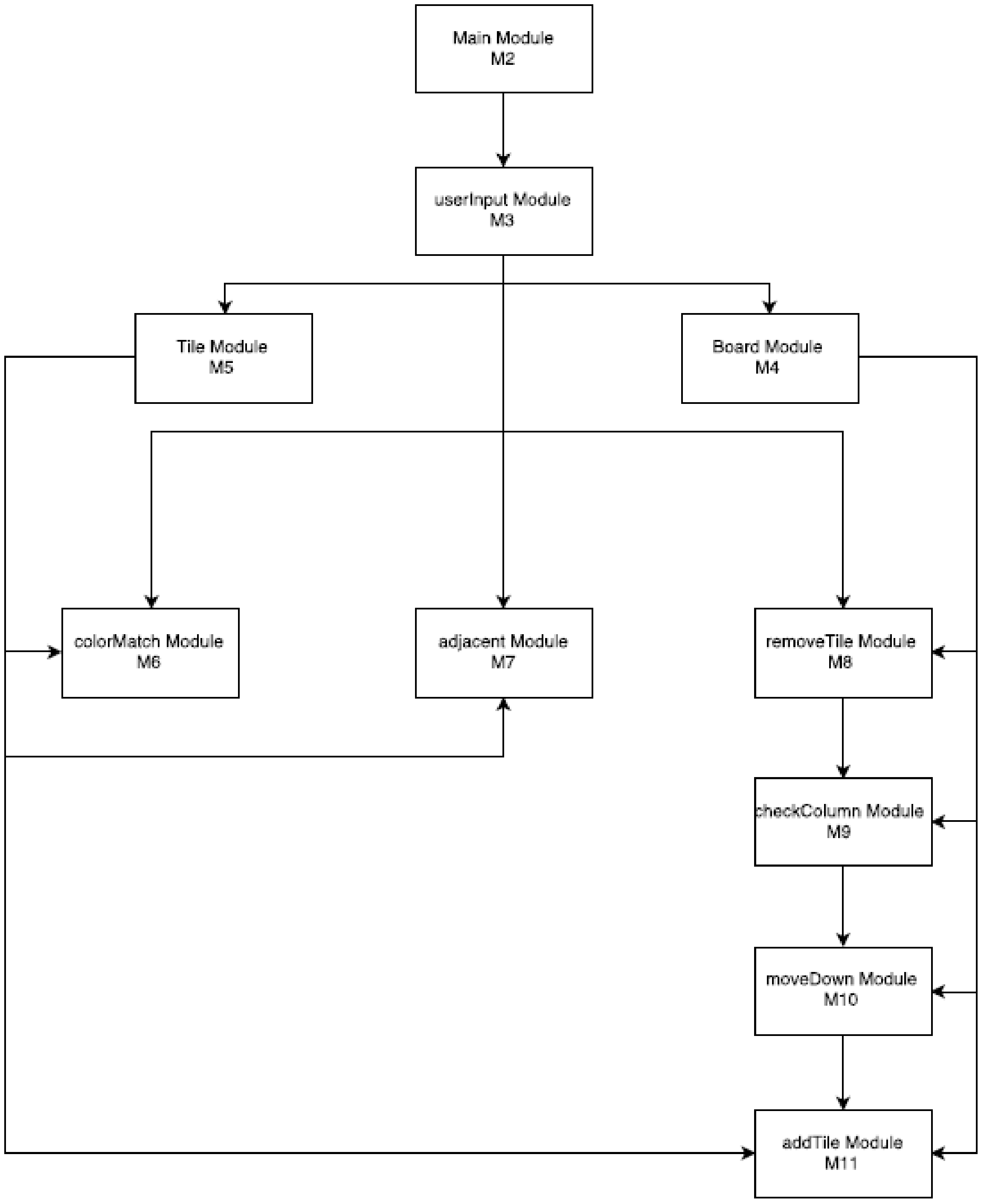
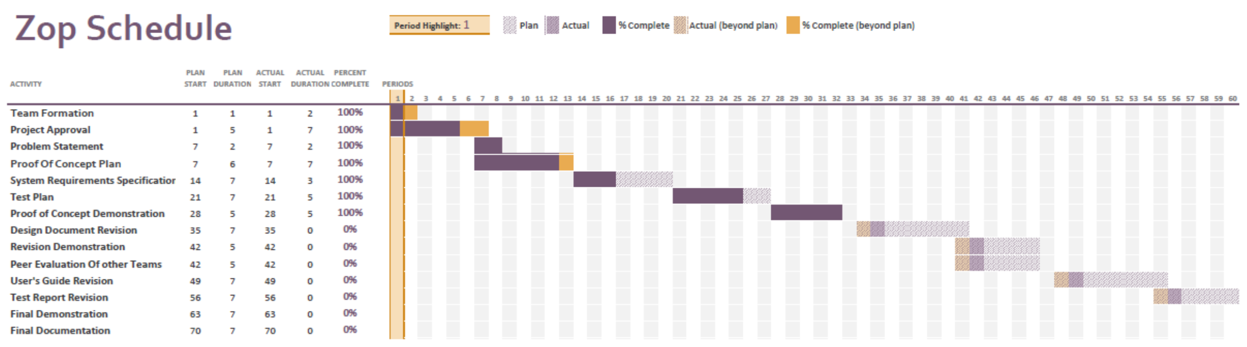


Figure 1: Use Table

8 Schedule

8.1 Gantt Chart



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Team Formation	1	1	1	2	100%
Project Approval	1	5	1	7	100%
Problem Statement	7	2	7	2	100%
Proof Of Concept Plan	7	6	7	7	100%
System Requirements Specificatior	14	7	14	3	100%
Test Plan	21	7	21	5	100%
Proof of Concept Demonstration	28	5	28	5	100%
Design Document Revision	35	7	35	0	0%
Revision Demonstration	42	5	42	0	0%
Peer Evaluation Of other Teams	42	5	42	0	0%
User's Guide Revision	49	7	49	0	0%
Test Report Revision	56	7	56	0	0%
Final Demonstration	63	7	63	0	0%
Final Documentation	70	7	70	0	0%

Figure 2: Gantt Chart

8.2 Pert Chart

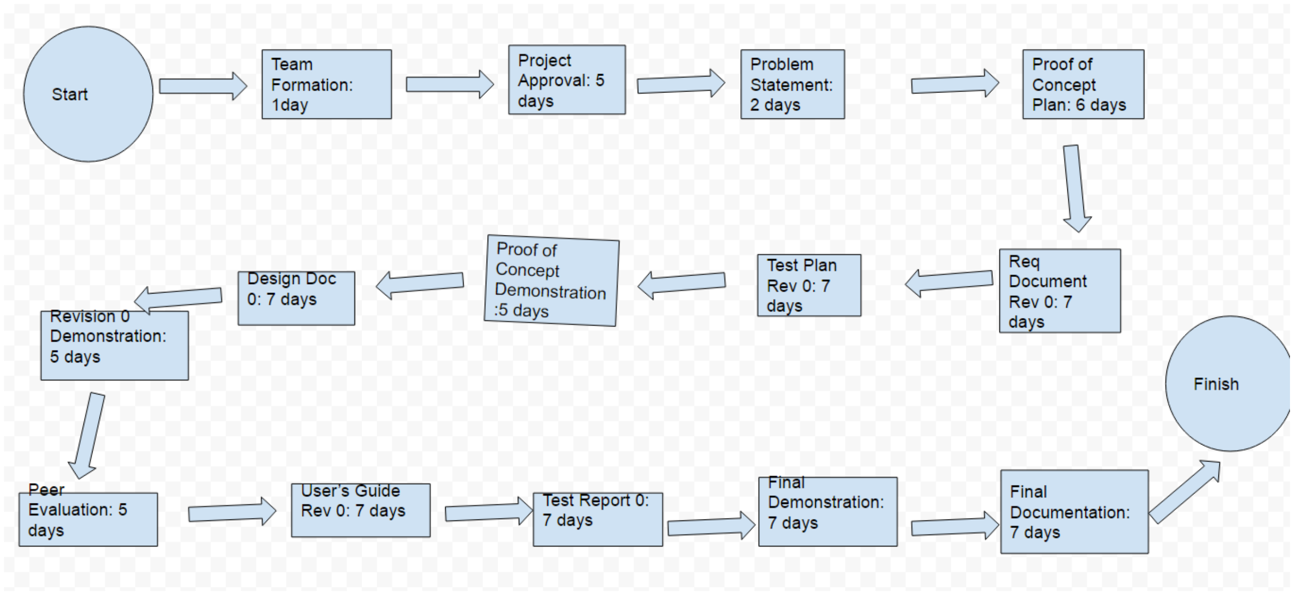

Start
Team Formation: 1day
Project Approval: 5 days
Problem Statement: 2 days
Proof of Concept Plan: 6 days
Req Document Rev 0: 7 days
Test Plan Rev 0: 7 days
Proof of Concept Demonstration :5 days
Design Doc 0: 7 days
Revision 0 Demonstration: 5 days
Peer Evaluation: 5 days
User's Guide Rev 0: 7 days
Test Report 0: 7 days
Final Demonstration: 7 days
Final Documentation: 7 days
Finish

Figure 3: Pert Chart