SE 3XA3: Module Guide
Scrabble

Team #214, The Trifecta
Kanakabha Choudhri, Choudhrk
Lucia Cristiano, Cristial
Raymond Tu, Tur1

April 6, 2020

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
13/2/20	~~1.0~~ 0.0	—
3/4/20	1.0	—

1 Introduction

1.1 Overview

The Scrabble project is a python language implementation of the classic family game Scrabble ~~in python~~ that had been implemented with an ASCII based user interface on the command line terminal. The goal of the project is to take the original implementation that exists on Github and improve on various aspects outlined in the SRS. The primary focus of these additional features is to implement the game as a graphical user interface using the Tkinter library in python. This would make the game more accessible to users not familiar with use of the command line terminal, in an easily understood and aesthetically pleasing format. Additionally, in the back-end modules will be re-implemented to allow for modularity and ease of maintainability for new features in the future. Overall, this would lead to better software design that is easier to maintain and update for the future.

1.2 Scope

The scope of the project is based around the original Scrabble repository, any new features or functionality added is to make it closer to the original game. For example, there are several new modules added for background checks that are in line with the original rules of the game but omitted from the original Github implementation such as enforcement of scrabble words in a crossword fashion, checking for other possible adjacent word combinations that add onto a players score, etc. The Tkinter library in Python will be the ~~main~~ primary tool used to create the GUI for Scrabble.

1.3 Module Guide Purpose

The purpose of this document, the Module Guide (MG) is to outline the modular structure of the system and allow any future contributors to easily identify parts of the program. New contributors will be able to refer to this document in the future to find relevant documentation for the modules they are working on. Additionally, the module guide will be helpful to check for consistency in the overall design of the game, the cohesion between modules and their coupling to ensure good software design. The module guide will outline how the modules in the back-end and front-end are broken up to have high cohesion and low coupling. Overall, the module guide will ensure developers and future contributors have good documentation to gain an understanding of the software, ~~and~~ ensure consistency between the relationships of the modules, and allow for identification of opportunities for addition of newer changes to the planned implementation.

The rest of the document is organized as follows. Section 2 lists the anticipated and unlikely changes of the software requirements. Section 3 summarizes the module decomposition that was constructed according to the likely changes. Section **??** specifies the connections between the software requirements and the modules. Section 5 gives a detailed description of the modules. Section 6 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section **??** describes the use relation between modules.

2 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 2.1, and unlikely changes are listed in Section 2.2. Anticipated changes are feasible changes that will be implemented in the future, while unlikely changes are those that may improve the quality of the user experience but are not likely to be implemented.

2.1 Anticipated Changes

AC1: ~~Implementation of a Skip Turn button. This would prevent deadlock in the game if a player is unable to make a move.~~ Change has been implemented

AC2: ~~Implementation of an Exchange Tiles button. This would help prevent deadlock if available tiles do not create a word that can be placed.~~ Change has been implemented

AC3: Scoreboard at the end of a match. Display player total score and who won at the end of a match.

AC4: Improved user interface, more adjustable window resolution and default size. Make the interface more friendly to different monitor sizes, and more readable text/font sizes.

2.2 Unlikely Changes

UC1: Input/Output devices. It is possible that the input may be changed from the default keyboard and mouse to other forms such as a touch screen, but given the scope of the project this is unlikely to occur.

UC2: Different environments/platforms. It is possible for portability that additional environments/platforms other than Windows OS are supported, for example the Scrabble game as a mobile application on Android of Apple OS, but given the scope of project this is unlikely.

UC3: Different numbers of players. It is possible to enhance the user experience by allowing more than 2 players to play at once and balance the game accordingly with varying types of scoring and number of tiles in the bag to adjust game length. However, given the scope and timeframe of the project this is unlikely to occur.

UC4: Singleplayer options. It would be possible to create a singleplayer option where the player faces the computer designed with machine learning. However, this is outside of the scope of the project and is unlikely to be implemented.

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: ~~Game Window~~ Main Game

M2: ~~Gameboard Boolean Checks~~ BoardChecks

M3: EndTurn ~~Boolean Checks~~

M4: ~~Valid Word Boolean Checks~~ WordChecks

M5: ~~Game View~~ GameController

M6: Board

M7: Player

M8: Rack

M9: Tiles

M10: Bag

M11: WidgetCreation

Level 1	**Level 2**
Hardware-Hiding Module	Main Game
Behaviour-Hiding Module	Game ~~View~~ Controller ~~End Turn Boolean Checks~~ WidgetCreation
Software Decision Module	Bag Board Player Rack Tiles ~~Valid~~ Word ~~Boolean~~ Checks EndTurn ~~Gameboard Boolean Checks~~ BoardChecks

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

Module	Requirement Module Meets
Tile	FR10, MS1, MS2, MS3, LR1
Bag	FR5, MS1, MS2, MS3, LR1
Rack	FR11, FR14, MS1, MS2, MS3, LR1
Player	FR4, MS1, MS2, MS3, LR1
Board	FR2, MS1, MS2, MS3, LR1
EndTurn	FR10, MS1, MS2, MS3, LR1
WordChecks	FR6, MS1, MS2, MS3, CR1, LR1
BoardChecks	FR6, FR8, FR9, MS1, MS2, MS3, LR1
WidgetCreation	FR8, LF1, UH2, MS1, MS2, MS3
GameController	FR6, FR7, FR8, FR9, FR10, FR12, PR2, MS1, MS2, MS3
~~MainGame~~ Main	FR2, FR4, FR7, ~~FR8~~, FR10, ~~FR12~~ FR13, FR15, FR16, FR17 LF1, UH1, UH2, UH3, UH4, UH5, UH7, PR1, PR2, OE1, OE2, CR1, LR1, HSR1

Table 3: Requirements and Design Connection

5 Module Decomposition

The purpose of having modules is to decompose the system into manageable pieces of code that can be more easily developed and maintained ~~than~~ as compared to a monolithic piece of software. The way the modules are decomposed is based around the principle of information hiding, which states that each module has one secret that is hidden from the other modules. These secrets are parts of the implementation that are likely to change. Each module also has a service which is the functionality that the module provides without describing the implementation of these services. For each of the modules given in the hierarchy their secret and service is listed as well as how the module will be implemented.

5.1 Hardware Hiding Modules - Main Game (M1)

Secrets: The data structures and algorithms used to handle virtual input and output, for example the screen, keyboard and mouse.

Services: Serves as a virtual hardware for the screen, keyboard, mouse used by the rest of the system to accept input and display output to the screen. This module provides the interface between the hardware and the software.

Implemented By: OS

5.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours of the player and the game board interface.

Services: Includes programs that provide externally visible behaviour of the game board as outlined in the software requirements specification (SRS) document. These modules serve as a middle layer that provides communication layer between the hardware-hiding module and the software decision modules. The programs in this module will need to change if there are changes in the SRS as these modules deal with the implementation of the functional requirements.

Implemented By: Scrabble Project

5.2.1 End Turn Boolean Checks (M3)

~~**Secrets:** The logic for when a move is completed by a player.~~

~~**Services:** Provides the behaviour necessary for updating the interface based on changes in data once the player has made a move.~~

~~**Implemented By:** Scrabble Project~~

5.2.2 WidgetCreation (M11)

Secrets: The creation of the tile and label widgets of the game board.

Services: Provides the behaviour necessary for creating tiles and the various labels on different components of the game board.

Implemented By: Scrabble Project

5.2.3 Game Controller (M5)

Secrets: The interface of the game that the player interacts with.

Services: Displays the game board, input boxes and buttons necessary for the player to interact with the game.

Implemented By: Scrabble Project

5.3 Software Decision Module

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. Changes to these modules are more likely to be motivated by a desire to improve performance than by externally imposed changes.

Implemented By: –

5.3.1 Gameboard Boolean Checks (M2)

Secrets: The algorithms dealing with the correct placement of words on the board.

Services: Provides the behaviour associated with ensuring that a word is correctly placed on the board.

Implemented By: Scrabble Project

5.3.2 Valid Word Boolean Checks (M4)

Secrets: The algorithms to determine if a valid word is entered by the player

Services: Provides the behaviour necessary for ensuring that only valid words are used in the game.

Implemented By: Scrabble Project

5.3.3 End Turn Boolean Checks (M3)

Secrets: The logic for when a move is completed by a player.

Services: Provides the implementation necessary for checking and alerting the players on the end of a certain players turn in the interface based on changes in data once a certain player has made a move.

Implemented By: Scrabble Project

5.3.4 Board (M6)

Secrets: The data structure that represent the Scrabble game board.

Services: Includes data structure, setters and getters used to model the board of a game of Scrabble. Used by checks to ensure correct placement of words.

Implemented By: Scrabble Project

5.3.5 Player (M7)

Secrets: The data structure that represents a player participating in a Scrabble game.

Services: Includes data structure, setters and getters used to model the data associated with a player, such as score and rack of tiles.

Implemented By: Scrabble Project

5.3.6 Rack (M8)

Secrets: The data structure that represents a rack of tiles in a Scrabble game.

Services: Includes data structure, setters and getters used to model a generic rack of game tiles.

Implemented By: Scrabble Project

5.3.7 Tiles (M9)

Secrets: The data structure that represents a tile in a Scrabble game.

Services: Includes data structure, setters and getters used to model the tile such as score and letter.

Implemented By: Scrabble Project

5.3.8 Bag (M11)

Secrets: The data structure that represent the bag of tiles.

Services: Includes data structure, setters and getters used to model the bag aspect of a game of Scrabble.

Implemented By: Scrabble Project

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes. Some functional requirements, namely FR1 and FR3 have been deemed obsolete based on feedback from teaching assistants and review of the requirements by the team.

Req.	**Modules**
FR1	Obsolete
FR2	M1, M5, M6
FR3	Obsolete
FR4	M1, M5
FR5	M1, M5, M8, M9
FR6	M1, M5, M4
FR7	M1, M5
FR8	M1, M5, M2
FR9	M1, M5, M2, M6
FR10	M1, M5, M9
FR11	M1, M5
FR12	M1, M5, M3, M11
FR13	M1
FR14	M1, M8
FR15	M1
FR16	M1
FR17	M1

Table 4: Trace Between Requirements and Modules

AC	**Modules**
AC1	~~M5~~ Change implemented
AC2	~~M5~~ Change implemented
	~~M3~~ Change implemented
AC3	M5
AC4	M5
	M1

Table 5: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

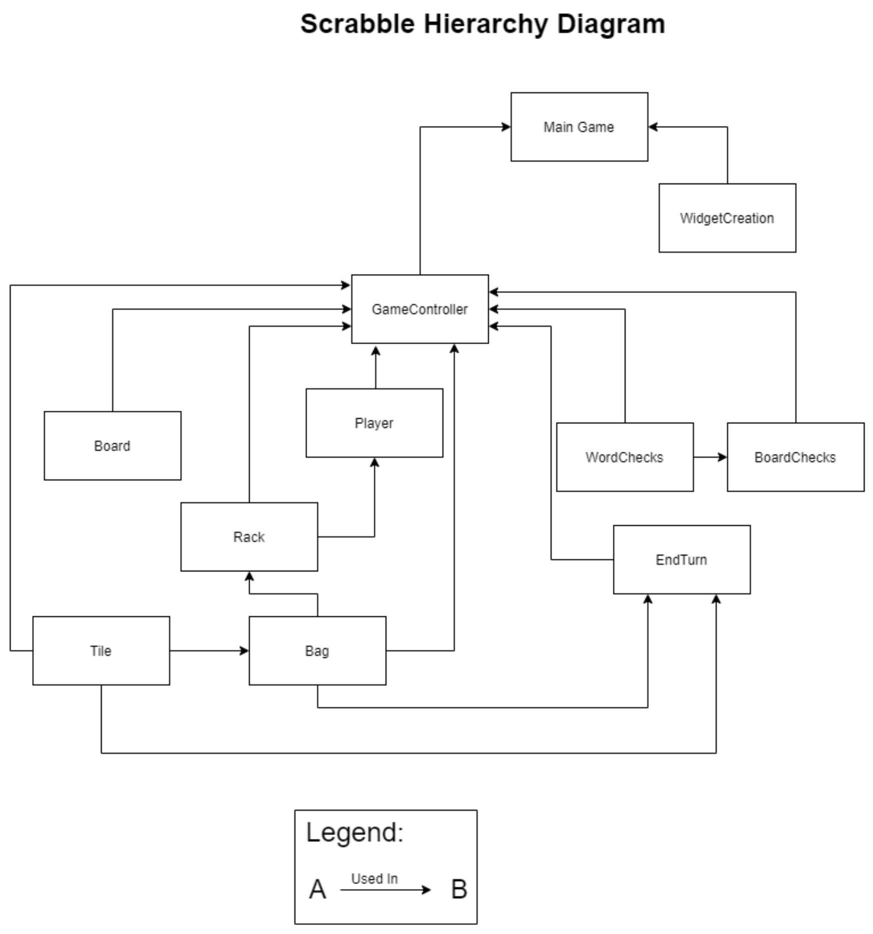

Figure 1: Use hierarchy among modules