SE 3XA3: Design Module Guide Document
ReTouch

Team #7, ReTouchers
Abrar Attia - attiaa1
Susan Fayez - fayezs
Mediha Munim - munimm

December 6, 2017

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
November 10	1.0	Added Module Hierarchy
November 10	1.1	Added Traceability Matrix between Modules and Requirements
November 10	1.2	Added Module Decomposition and Requirements/Design Connections
December 6	2.0	Added Button Listener Module, Updated Uses Hierarchy

1 Introduction

1.1 Overview

The ReTouch project is a re-implementation of the open source project K-Touch which allows users to track their speed and accuracy in typing, and results in improved typing skills through practice and repetition.

1.2 Context

The Module Guide document is created after the Software Requirements Specification (SRS) document. The purpose of this is to allow the project design to be based off a set structure of functional and non-functional requirements that are featured in the SRS. Therefore, this MG document is used to provide a modular decomposition of the system, and by following the SRS, all the featured modules and the system is ensured to meet the functional and non-functional requirements. Additionally, the Module Interface Specification (MIS) is created after the MG document which provides further details on all the modules specified in the MG document. The MIS specifies the externally observable behavior and explains the semantics and syntax of the functions for each module in the system.

1.3 Design Principles

The Design Principles being used to guide the decomposition of the system into modules include Information Hiding which supports design for change, and the criteria that the modules have low coupling and high cohesion. The principle of Information Hiding allows for all expected changes to be identified and encapsulates each expected change. In other words, all module secrets are hidden from the rest of the system and the module design interface will not change even if a a module's secret changes. Additionally, low-coupling would mean that the modules are independent and do not use one another. High cohesion within the modules would indicate that the elements of the module are strongly related.

1.4 Document Structure

This Module Guide 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 4 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 7 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.

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 specific hardware on which the software is running.

AC2: The format of the initial input data.

AC3: The Operating Systems of which the Software interfaces with.

AC4: The format of the final output and display.

AC5: The graphical user interface elements used to display effects of user input.

AC6: The calculation methods for determining user speed and accuracy.

AC7: The number of lessons (and their contents) that can be selected by the user.

AC8: The precision of the timer and whether it should measure milliseconds.

AC9: The data structure used to make the buttons to return to the lesson selection page.

2.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

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

UC2: There will always be a source of input data external to the software.

UC3: The goal/purpose of the system: To assist users in enhancing their typing abilities by tracking their speed and accuracy when completing lessons.

UC4: The methods and functions used to track whether inputted characters match the expected characters.

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: Display Characters Module

M3: Lesson Module

M4: Parser Module

M5: Selection Page Module

M6: User Input Module

M7: Make Results Module

M8: Results Module

M9: Scheduler Module

M10: Button Listener Module

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Display Characters Module Lesson Module Parser Module Selection Page Module User Input Module
Software Decision Module	Make Results Module Results Module Scheduler Module Button Listener 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

All of the modules in the creation of ReTouch were decomposed with Information Hiding in mind. With this system, each module carries one secret and has a set of functions (services). This section examines each module, defining its secret and the services/functions it provides.

5.1 Hardware Hiding Module (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

5.2 Behaviour-Hiding Modules

5.2.1 Display Characters Module (M2)

Secrets: The GUI elements used to structure and display the characters.

Services: Displays the characters in the lesson on screen and updates their colour, as well as displaying the current character.

Implemented By: ReTouch

5.2.2 Lesson Module (M3)

Secrets: The data structure used to construct the character input UI page.

Services: Constructs the UI page that displays the lesson characters and accepts user input.

Implemented By: ReTouch

5.2.3 Parser Module (M4)

Secrets: The format and structure of the input data.

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: ReTouch

5.2.4 Selection Page Module (M5)

Secrets: The data structure used to construct the selection page UI.

Services: Constructs the UI page that displays that allows users to select a lesson.

Implemented By: ReTouch

5.2.5 User Input Module (M6)

Secrets: The algorithms used to handle user input.

Services: Handles user input by comparing all the inputted characters to the characters in the lesson.

Implemented By: ReTouch

5.3 Software Decision Modules

5.3.1 Make Results Module (M7)

Secrets: The algorithm that produces the information for a results GUI component.

Services: Compiles the results of the lesson into a GUI.

Implemented By: –

5.3.2 Results Module (M8)

Secrets: The algorithm that produces/displays a GUI component.

Services: Displays the results of the lesson.

Implemented By: –

5.3.3 Scheduler Module (M9)

Secrets: The processes that calculate the mathematical formulas used in the lesson.

Services: Calculates and displays the calculated parts of the lesson.

Implemented By: –

5.3.4 Button Listener Module (M10)

Secrets: The data structure used to implement the buttons.

Services: Creates a button that returns to the lesson selection page.

Implemented By: –

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
FREQ1	M5
FREQ2	M4, M5
FREQ3	M2, M3, M5
FREQ4	M2, M6
FREQ5	M2, M6
FREQ6	M2, M6
FREQ7	M2, M6
FREQ8	M2, M6
FREQ9	M2, M6
FREQ10	M6
FREQ11	M3, M9
FREQ12	M3, M9
FREQ13	M3, M9
FREQ14	M6
FREQ15	M6, M7, M8
FREQ16	M10, M3
FREQ17	M10, M8

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M4
AC3	M3
AC4	M7, M8
AC5	M2, M6, M9
AC6	M9, M6
AC7	M3, M5
AC8	M5, M7
AC9	M10

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

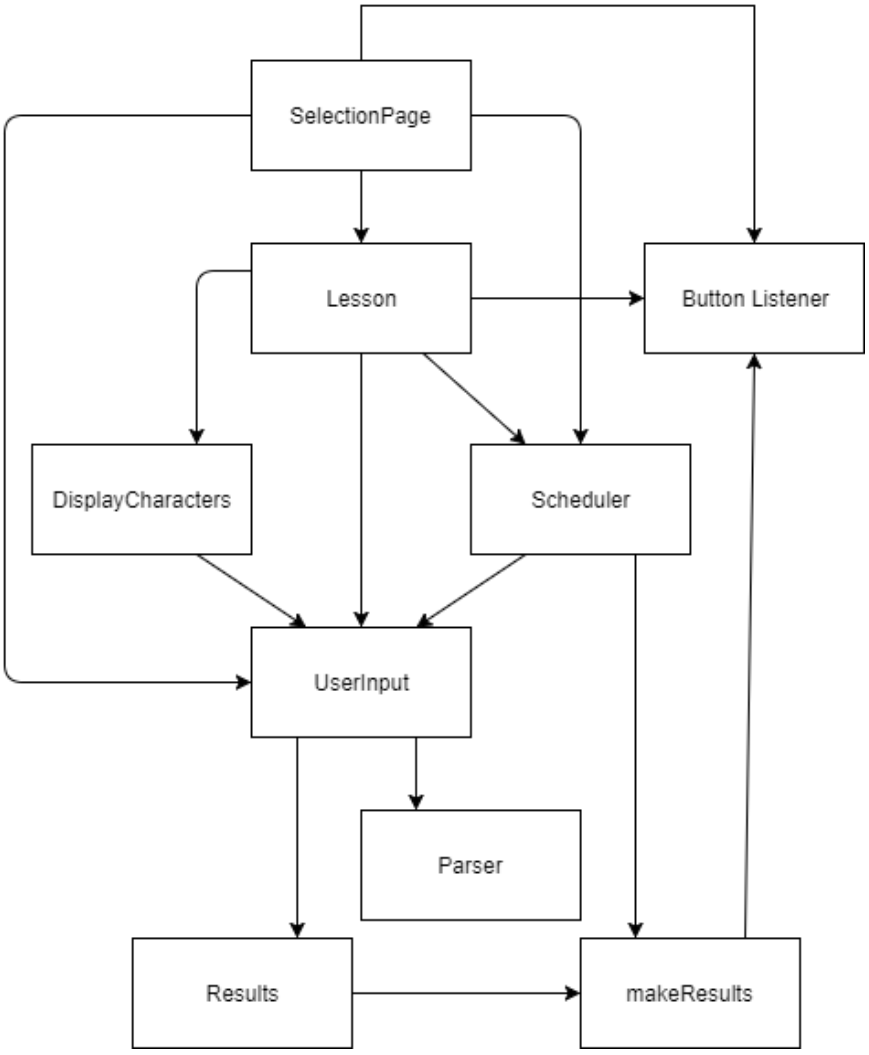

Figure 1: Use hierarchy among modules