SE 3XA3: Module Guide
CamRuler

Team 10,
Kshitij Mehta, mehtak1
Meet Patel, patelm16
Prince Kowser, kowserm

December 6, 2017

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
Nov.11,2017	1.0	Revision 0 Complete
Dec.6,2017	1.1r	Revision 1 Complete

1 Introduction

The CamRuler application is a re-implementation of an existing open-source Android application which enables a user to get measurements of an object through taking a picture.

This document is the Module Guide (MG) and is created after the Software Requirements Specification (SRS). The main purpose of the SRS is to specify all the functional and non-functional requirements for the project, compared to the MG which provides a modular decomposition of the system to display the modular structure of it. The MG also documents how the system meets both functional and non-functional requirements specified in the SRS. The Module Interface Specification (MIS) is created after the MG which MIS explains the semantics (state and environment variables, assumptions, and access routines) and syntax of exported functions (input, output, and exceptions) for each module. In other words, it essentially provides more detail on each module specified in the MG.

Design Principles being used to create the decomposition of the system into modules are Information Hiding and Encapsulation, as well as the principle that a Uses Relation (Hierarchy) should contain no cycles, has low coupling, and high cohesion. Firstly, Information Hiding is when each module hides a secret (a design decision) from the rest of the system. Encapsulation is explained to be when changeable information is in the implementation of the module, but the module interface should not change when the implementation changes; the services which it provides are accessed in a consistent way regardless of changes in implementation. Low coupling in the Uses Relation indicates that the modules are independent and do not use many other modules. High cohesion means that the elements within a module are strongly related. Lastly, having cycles in the Uses Relation indicates that module A uses module B, which uses module A. This is considered poor design because it can cause infinite loops and there is often a more efficient decomposition of the modules. Potential readers of this document are new project members, maintainers, and designers.

Our design follows the rules layed out as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is used in only one module.

- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

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 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 describes all the possible changes to the system. The Anticipated and Unlikely changes are listed in Section 2.1, and Section 2.2 respectively.

2.1 Anticipated Changes

AC1: The format of the input data for the reference object (e.g input measurements in different units).

AC2: The format of the output data (e.g displaying the measurements in different units).

AC3: How the user draws points on the picture (e.g allowing users to drag points or lines).

AC4: How the image is displayed on the application (e.g setting the image to full screen size, allowing user to zoom in).

AC5: The number of objects a user can select.

2.2 Unlikely Changes

UC1: The hardware platform (Android phones).

UC2: The language of the interface (i.e Java, Android).

UC3: The purpose of the system which is to display the measurements of an object.

UC4: How the desired measurements are calculated using the reference object's measurements and ratios.

3 Module Hierarchy

M1: Hardware Hiding Module

M2: Input Dialog Module

M3: ImageView Module

M4: DrawView Module

M5: Calculation Module

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Dialog Module ImageView Module DrawView Module
Software Decision Module	Calculation Module

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

The CamRuler system is designed so that all the requirements listed in the SRS are satisfied. A more detailed illustration of which modules satisfy which requirements is given in Table 4.

In order to satisfy the requirements listed in the SRS, several design decisions had to be made. For example:

- Ease-of-Use Requirement (NF-EU): An instruction set will be added to the application where a user can read the instructions on how to use the application before using the main functions of the application.
- Integrity Requirement (NF-SIR)-Security: An exception needs to created where upon an invalid input (e.g entering letters for the measurements of the reference object), an error message is displayed and the user is prompted to re-enter the measurements.
- Robustness Requirement (NF-RFTR): The user needs to be given the option to select an image from their gallery along with the option to take a picture. Therefore, the Picture Taking Module needs to incorporate a function that can access the gallery application on the phone. This way, if the phone's camera is not working, the user can select a picture from their gallery provided that they took a picture of the object beforehand.

5 Module Decomposition

The modules specified in the Module Hierarchy section are decomposed using the concept of information hiding. The Secrets part in a module decomposition states the design decision(s) hidden by the module. The Services part indicates what the module will do without actually documenting how to do perform that certain task. The Implemented By section suggests an implementing software for each module. For instance, if the entry is OS, the module is provided by the operating system or by standard programming language libraries. Description should also indicate if the module will be implemented specifically for the software.

If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented as only leaf modules have to be implemented. Whether or not this module is implemented depends on the programming language being used.

5.1 Hardware Hiding Modules

Secrets: Data structure and/or algorithm used to implement the virtual hardware

Services: Serves as interface between software and hardware so system can use it to display outputs or to accept inputs.

Implemented By: Java Virtual Machine and Operating System

5.2 Behaviour Hiding Module

Secrets: Contents of required behaviours

Services: This module acts as an interpreter between Hardware Hiding Module and Software Decision Module while also describing the visible behaviour of the system. The modules in this section are written in terms of the SRS document to make sure system behaves as described in SRS.

Implemented By: N/A

5.2.1 MainActivity Module

Secrets: Allows user to pick number of object

Services: This module runs the code by combining all modules.

Implemented By: Java Libraries and all other classes

5.2.2 Image Module

Secrets: User Input of Reference Points Measurements

Services: This module allows user to input reference point measurements.

Implemented By: Java Libraries and Android Studio

5.2.3 Utilities Module

Secrets: Converts between input and output units

Services: This module allows user to input reference point measurements.

Implemented By: Java Libraries and Android Studio

5.2.4 SurfaceImage Module

Secrets: Background Image

Services: This module allows user to take a picture which is then set as the background of the application.

Implemented By: Java Libraries

5.2.5 DrawingOnImage Module

Secrets: User Selection of Reference Points

Services: This module allows user to select reference points on picture taken.

Implemented By: Java Libraries

5.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: N/A

5.3.1 Calculation Module

Secrets: ~~Mathematical Calculations~~ Conversion Algorithm

Services: Calculates measurement of picture taken using mathematical ratios in various kinds of metrics.

Implemented By: Java Libraries

6 Traceability Matrix

6.1 Modules and Requirements

Req.	**Modules**
FR1	M2
FR2	M2
FR3	M2, M3
FR4	M3, M4
FR5	M4
FR6	M3, M4
FR7	M2
FR8	M5

Table 3: Trace Between Functional Requirements and Modules

Req.	Modules
NF-AP	M2, M3, M4, M5
NF-EU	M2, M3, M4, M5
NF-UPR	M2, M5
NF-AR	M1
NF-SLR	M2, M3, M4, M5
NF-SCR	M3
NF-PAR	M5
NF-RAR	M1
NF-RFTR	M3
NF-CR	M4
NF-SER	M4
NF-RIA	M1
NF-SR	M1
NF-SIR	M2
NF-SPR	M4
NF-SCR	M2, M5
NF-LSR	M1, M2, M3, M4, M5

Table 4: Trace Between Non-Functional Requirements and Modules

6.2 Modules and Anticipated Changes

AC	Modules
AC1	M2
AC2	M5
AC3	M4
AC4	M3
AC5	M4

Table 5: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

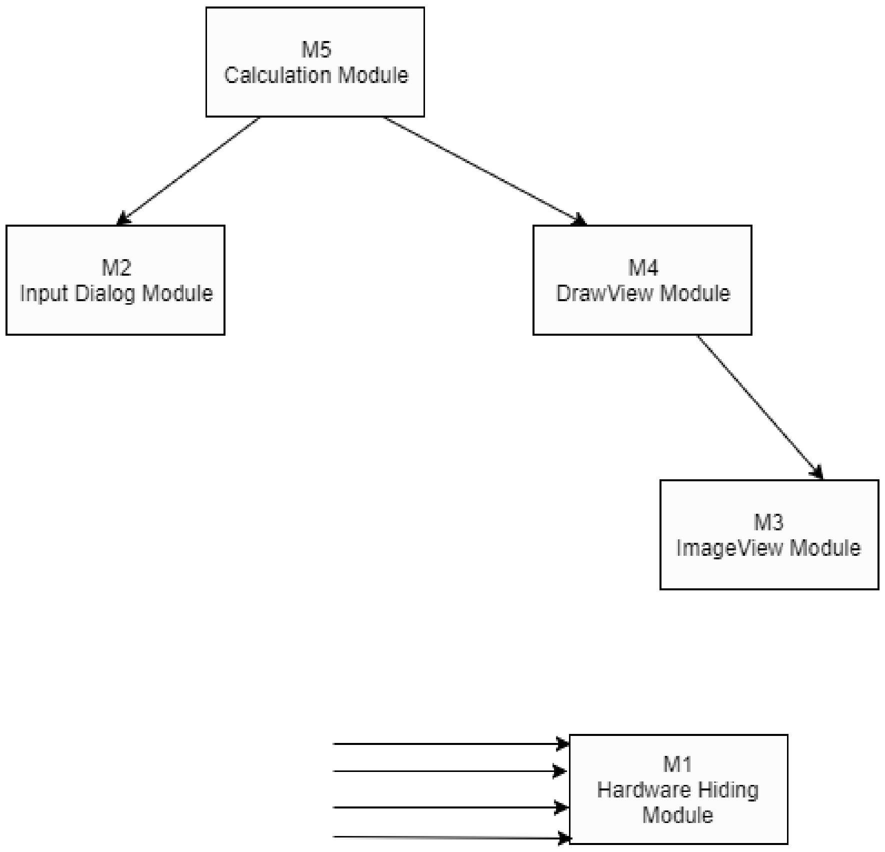

Figure 1: Use hierarchy among modules