SE 3XA3: Module Guide
OpenCameraRefined

Team # 211, Camera Crew
Faisal Jaffer, jaffem1
Dominik Buszowiecki, buszowid
Pedram Yazdinia, yazdinip
Zayed Sheet, sheetz

April 7, 2020

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
3/13/2020	1.0	Initial documentation
4/6/2020	1.2	Revision 1

1 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by Parnas et al. (1984), 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.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (Parnas et al., 1984). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

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.

1.1 Software Overview

OpenCameraRefined is designed to change the way we capture pictures through our phones. Using image processing algorithms and machine learning, the team is focused on enabling the software to process and recognize a certain gesture in real time. In addition, the software is designed to offer the user with real time filters, mainly using existing packages such as OpenCV. Some initial gestures recognized by the software can include Smile, Thumbs up or Wave.

OpenCamera is an open-source Andorid application that provides Camera functionality outside the stock camera applications in varous phones. Android system initiates its program within an Activity starting with a call on onCreate() callback method. There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity. This model is loosely based on MVC architecture with the controller being the Activity class, View being the resources and widgets and model being the entities or Classes with main Business Logic. Consequently, the modifications made follow the same principles.

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.~~ The gestures (smiling, thumbs up, wave, ...) recognized at any time by the system.

AC3: The number of gestures that can be detected by the model concurrently at any given time.

AC4: ~~The gestures recognized by the algorithm.~~ Addition of new filters that allow for different changes to the live preview.

AC5: The number of filters available to be applied.

AC6: Trigger gestures used to trigger different actions such as filter change or image capture.

AC7: Modifications to the UI, including extra functionality buttons that manually trigger different functions.

AC8: The render protocols used to draw the real time filters.

AC9: The write protocols used to capture and save the live preview with the filter applied.

2.2 Unlikely Changes

UC1: Input/Output devices (Input: Camera, Output: Memory, Screen).

UC2: Constant presence of a source input external to the system

UC3: ~~The implementation and output of gesture recognition will remain the same as this is used throughout several modules, and will require many modifications to change.~~ The implementation of the trained model used to detect different gestures.

UC4: ~~The user interface and settings will remain the same as the original Open Source Camera application in order to keep the application consistent and straightforward.~~ The original user interface from Open Camera, excluding the added filter button.

UC5: ~~Zoom and slow motion features will remain the same as this is outside the scope of this project.~~ Advanced functionalities of the camera including Zoom, contrast change and etc.

UC6: The actual training of the gesture model to detect new gestures.

UC7: Render protocols used to draw unfiltered preview.

UC8: Read and Write protocols used to edit old pictures or save new pictures.

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: Gesture Controller

M2: Recognition

M3: Classifier

M4: Classifier Constants

M5: ~~Filter~~ Image Filter Controller

M6: Filter Constants

M7: Hardware Hiding Module

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Modules	Classifier Constants Recognition Filter Constants
Software Decision Modules	Classifier Gesture Controller ~~Filter~~ Image Filter Controller

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" proposed by Parnas et al. (1984). The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* OpenCameraRefined. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. Also indicate if the module will be implemented specifically for the software.

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 (M7)

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 Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

5.2.1 Recognition Module(M2)

Secrets: The data and the state stored in the object.

Services: Provides access to the location, title and confidence of self.

Implemented By: OpenCameraRefined

5.2.2 Filter Constants Module(M6)

Secrets: The predefined stored filter matrices

Services: Provides access to the predefined filters.

Implemented By: OpenCameraRefined

5.2.3 Classifier Constants Module (M4)

Secrets: The file location of the model inference and the label.

Services: Provides access to the file location strings.

Implemented By: OpenCameraRefined

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.

Implemented By: –

5.3.1 Classifier Module(M3)

Secrets: The current sate of the running TensforFlow model.

Services: Provides methods to run classifications on images.

Implemented By: OpenCameraRefined

5.3.2 ~~Filter Module~~ Image Filter Controller Module(M5)

Secrets: The state of the current filter in view.

Services: Provides method to change the current filter.

Implemented By: OpenCameraRefined

5.3.3 Gesture Controller Module (M1)

Secrets: The flow of the information from module to module.

Services: Responsible for dispatching classification and actions based on the classification.

Implemented By: OpenCameraRefined

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
REQ1	M1, M2, M3, M4, M7
REQ2	M1, M2, M3
REQ3	M7
REQ4	M7
REQ5	M1, M2, M3, M4, M7
REQ6	M5 M7
REQ7	M1, M2, M3, M5, M7
REQ8	M1, M5, M7
REQ9	M1, M5, M7
REQ10	M5, M6, M7
REQ11	M7
REQ12	M1, M5, M7
REQ13	M1, M5, M7

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M7
AC2	M2
AC3	M1
AC4	M6
AC5	M6
AC6	M1
AC7	M5
AC8	M5
AC9	M7

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas (1978) said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. 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.

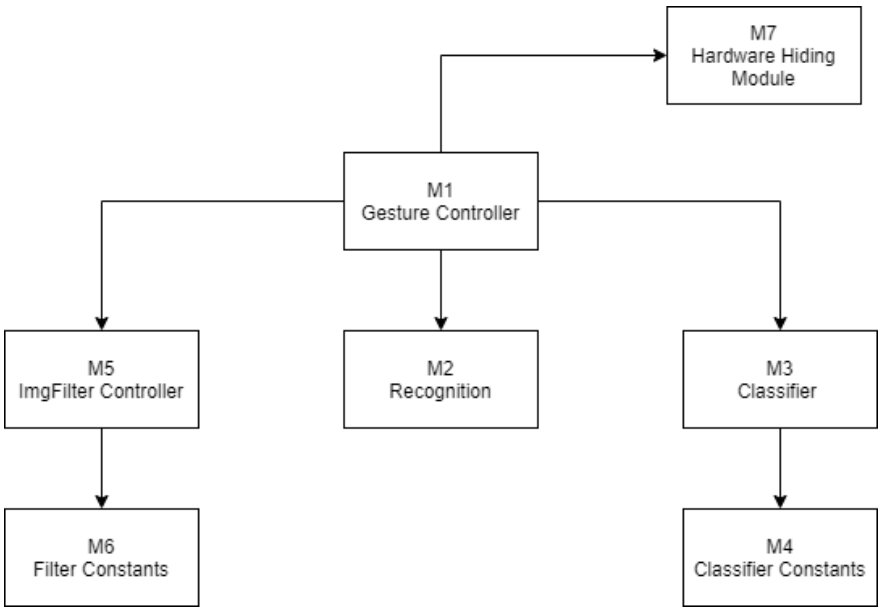

Figure 1: Use hierarchy among modules

References

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.