Module Guide for Mechtronics Enigeering

Team 32, Wingman, SmartVault
Edward He
Erping Zhang
Guangwei Tang
Peng Cui
Peihua Jin

January 18, 2023

1 Revision History

Date	**Version**	**Notes**
1.10	1.0	Revise the template
1.14	1.1	Complete the detailed discussion on Module
1.16	1.1.1	Work on Likely Change and Unlikely Change
1.17	1.1.2	Finish the section 6 and section 7
1.18	1.2	Add the traceability matrices in the last section

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
DAG	Directed Acyclic Graph
LC	Likely Change
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
Mechtronics Enigeering	Explanation of program name
UC	Unlikely Change
[etc. —SS]	[... —SS]

Contents

List of Tables

List of Figures

3 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 implemented 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 4 lists the likely and unlikely changes of the software requirements. Section 5 gives a detailed description of the modules. Section 6 describes the use relation between modules. Section 7 specifies the connections between the software requirements and the modules and 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.

4 Likely 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. Likely changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Likely Changes

Likely changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the likely changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

LC1: The camera may get upgraded and replaced with the one with higher resolution to achieve more accurate images.
Rationale: To ensure that Image Processing Module could collect clear information with high resolution.

LC2: The security of user's private account will be strengthened.
Rationale: It is necessary to safely protect the private information of the users'.

LC3: The specific hardware on which the software is running.

LC4: The algorithm of Image Processing Module and Information Storage Module might be optimized.
Rationale: To accurately execute the task and efficiently track the missing item.

LC5: Any one except for the users is not allowed to access the any file that stores the information about the objects in the room.

LC6: The device is supposed to work in only indoor space.
Rationale: To ensure that the system is able to adapt multiple environment.

4.2 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: The maintenance for the device should be done by the developers.

UC2: The performance of the product should not violate the laws that protect the privacy of the user.
Rationale: To ensure that the private information of the users'is safely protected.

UC3: The device is supported by any computers supports both C and python programming languages.

UC4: The device should be easy to installed in the room.
Rationale: To satisfy the convenience of the installation of the device.

UC5: The device should not have exposed internal electronic wiring.

UC6: The base of the device should be strong enough without falling from the floor.
Rationale: To ensure that the base of the system is stable enough to avoid any accidents

5 Secret of Each Module

5.1 Software Modules

5.1.1 Login Module

Secrets: The data structure and unit used to login in to the personal account.

Services: This module provides the interface between the user and the application. So, the system can use it to display personal account.

Implemented By: Python

5.1.2 Information Storage Module

Secrets: The unit used to storage data and information hardware.

Services: Provides with a storage platform used by the rest of the system. This module protects the data and store the information and picture of the recorded items. Once the object detection is captured by the Image Processing Module, the camera will take a photo and store in the local file. Therefore, the system can use it to display and fetch out the desired information.

Implemented By: Python

5.1.3 Image Processing Module

5.1.3.1 Object Detection Module

Secrets: The data structure and algorithm used to implement the image processing and compare the image differences.

Services: This module serves as a tool to capture the object movement. In addition, it provides with the function for the comparison of different positions. So, the system can use it to process the required basic operation.

Implemented By: Python

5.1.3.2 Human Detection Module

Secrets: The data structure and algorithm used to implement the image processing and capture the movement of human.

Services: This module serves as a tool to capture the human movement. So, the system can use it to process the required basic operation.

Implemented By: Python

5.1.4 Information Extraction Module

Secrets: The module unit implemented to display specific information and attribute of the desired object.

Services: Serves as an information extraction system. This module cooperates with the Information Storage Module to output the required information which depends on the attributes chosen by the user. Consequently, the system can use it to display outputs according to various variables determined by the user.

Implemented By: Python

5.1.5 Communication Port 1 Module

Secrets: This module unit is used to connect the software data with the hardware module through sending the USB signal, which is considered as a serial communication system.

Services: Serves as a connection bridge between the hardware and software module in the system. This module works along with the Communication Port 2 Module to deal with the data transmission through sending command from the software end to the hardware end. Thus, the system can use it to transmit and receive data and signals from the hardware module unit.

Implemented By: Python

5.2 Hardware Module

5.2.1 Communication Port 2 Module

Secrets: This module unit is used to communicate the data and signal with the software module through reading the USB signal, which is defined as a serial communication system.

Services: Behaves as a link between the hardware and software module in the system. This module works along with the Communication Port 1 Module to receive the data transmitted from the software end. Therefore, the system can use it to receive data and signals from the software module unit and also send the data to the software system.

Implemented By: Python

5.2.2 Motor Control Module

Secrets: The hardware module designed to implement the control of step motor, including the two-dimension speed control, human movement capture.

Services: The module serves as a unit to control the rotatory angular velocity of the motor, which is implemented with a closed-loop control as the controller of the angular speed of step motor. So, the system can use it to proceed the desired basic operation.

Implemented By: Python

5.3 Module Summary

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

symbol	**description**
M1	Login Module
M2	Information Storage Module
M3	Image Processing Module
M4	Information Extraction Module
M5	Communication Port 1 Module
M6	Communication Port 2 Module
M7	Motor Control Module

Table 1: Module Summary

6 Relationship 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.

In summary, the user has to login into the account through Login Module M1. Then, he or she is expected to look for the desired item using Information Extraction Module M4 with multiple searching keys. The information and data are stored in Information Storage Module M2 and progressed by Image Processing Module M3. In addition, the data transmission between the hardware-end and the software-end is connected with Communication Port 1 Module M5 and Communication Port 2 Module M6. The motor is controlled and operated by Motor Control Module M7.

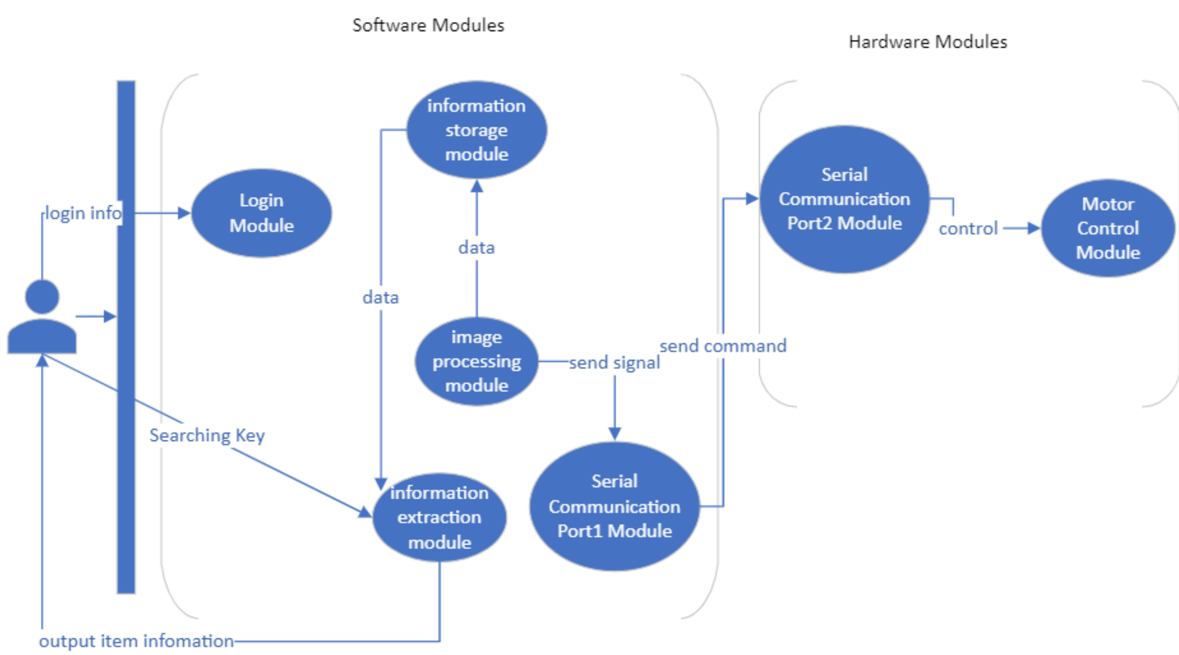

Figure 1: Use hierarchy among modules

7 Relationship Between Modules and SRS

This part specifies the connections between the software requirements and the modules.

IPR1: Image Processing Module M3 is able to identify human's body.

IPR2: Image Processing Module M3 could identify human's hand.

IPR3: Information Storage Module M2 and Image Processing Module M3 should be able to identify all the small items being exposed to the camera.

IPR4: Information Storage Module M2, Image Processing Module M3, Information Extraction Module M4 should take a photo once the change of the location of an item is captured.

IPR5: Information Storage Module M2, Image Processing Module M3, Information Extraction Module M4 should be able to differentiate one item from another items which are identified by the system through 3 main parameters, item_shape, item_color and item_size.

IPR6: Information Storage Module M2, Information Extraction Module M4 must be able to store all the photos into a file, and indicate the time when it was taken.

IPR7: Information Storage Module M2, Information Extraction Module M4 must be able to name each item with a unique ID.

IPR8: Information Storage Module M2 should be able to arrange the photos stored in the file in ascending or descending order according to the time it was taken.

IPR9: Information Storage Module M2 should be able to arrange the photos stored in the file in ascending or descending order according to their IDs.

UIR1: Information Storage Module M2 should be able to let user to choose whether to highlight a certain item or not.

UIR2: Information Storage Module M2, Information Extraction Module M4 should be able to let user to switch the ordering method.

UIR3: Login Module M1, Information Storage Module M2, Image Processing Module M3, Information Extraction Module M4 must be able to notify the user when the WI-FI signal is weak or unstable.

UIR4: Login Module M1, and Information Storage Module M2 must be able to allow the user to view the system's status at any given point in time.

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

Req.	**Modules**
IPR1	M3
IPR2	M3
IPR3	M2, M3
IPR4	M2, M3, M4
IPR5	M2, M3, M4
IPR6	M2, M4
IPR7	M2, M4
IPR8	M2
IPR9	M2
UIR1	M2
UIR2	M2, M4
UIR3	M1, M2, M3, M4
UIR4	M1, M2

Table 2: Trace Between Requirements and Modules

LC	**Modules**
LC1	M6, M7
LC2	M1 ,M2
LC3	M6
LC4	M2, M3
LC5	M1
LC6	-

Table 3: Trace Between Likely Changes and 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.