Module Guide for Pot-Pulator

Team 24
Gillian Ford
Juan Moncada
Aaron Billones
Steven Ramundi

April 5, 2023

1 Revision History

Date	Version	Notes
2023-01-07	1.0	Initial Release
2023-04-05	2.0	Update for final documentation

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

Symbol	Description
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
UC	Unlikely Change

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 anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 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 9 describes the use relation between modules.

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

4.1 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 physical input devices that the user will have available to them.

AC2: Calulations for timing conveyor stoppage and movement.

AC3: Sensors used for verification and validation (increased accuracy and lower latency).

AC4: Communication protocol used between the individual modules amd main board.

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: Use of distance sensing for validation and verification.

UC2: Use of Arduino microcontrollers to drive smaller modules.

UC3: Tray dispensing delivery method.

UC4: Pot dispensing mechanics.

5 Module Hierarchy

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 are implemented.

M1: Front End Module

M2: Communication Module

M3: Pot Dropping Input Module

M4: Pot Dropping Position Module

M5: Pot Dropping Stepper Module

M6: Pot Dropping Output Module

M7: Conveyor Input Module

M8: Conveyor Movement Module

M9: Tray Dispenser Input Module

M10: Tray Dispenser Output Module

M11: Verification Analysis Module

M12: Verification Output Module

6 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 2.

7 Module Decomposition

7.1 Behaviour-Hiding Module

7.1.1 Pot Dropping Input Module (M3)

Secrets: Subsystem state and main board command.

Services: Take in state and command and allow subsystem action.

Implemented By: [Pot-pulator]

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	Pot Dropping Input Module Pot Dropping Stepper Module Pot Dropping Output Module Conveyor Input Module Conveyor Movement Module Tray Dispenser Input Module Tray Dispenser Gantry Module Tray Dispenser Raising Module Tray Dispenser Output Module Verification Output Module
Software Decision Module	Pot dropping Position Module Verifications Analysis Module Communication Module Front End Module

Table 1: Module Hierarchy

7.1.2 Pot Dropping Stepper Module (M5)

Secrets: Position of dispensing thread.

Services: Drives stepper motors and dispenses pots.

Implemented By: [Pot-pulator]

7.1.3 Pot Dropping Output Module (M6)

Secrets: State of pot dropping subsystem

Services: Serves to maintian the status, errror and readiness of the pot dispensing subsystem as well as communicate this status with main control board.

Implemented By: [Pot-pulator]

7.1.4 Conveyor Input Module (M7)

Secrets: Subsystem state and main board command

Services: Takes in subsystem state and command and allows for system functionaility.

Implemented By: [Pot-pulator]

7.1.5 Conveyor Movement Module (M8)

Secrets: Subsystem state and main board command.

Services: Serves to drive conveyor motor and create movement.

Implemented By: [Pot-pulator]

7.1.6 Tray Dispenser Input Module (M9)

Secrets: Subsystem state and main board command.

Services: Takes in subsystem state and command and allows for system functionaility.

Implemented By: [Pot-pulator]

7.1.7 Tray Dispenser Output Module (M10)

Secrets: Subsystem state and main board command.

Services: Serves to maintian the status, errror and readyness of the tray dispensing subsystem as well as communicate this status with main control board.

Implemented By: [Pot-pulator]

7.1.8 Verification Output Module (M12)

Secrets: Verification analysis.

Services: Serves to communicate output failure with user interface.

Implemented By: [Pot-pulator]

7.2 Software Decision Module

7.2.1 Front End Module (M1)

Secrets: User inputs and individual subsystem status.

Services: Take in user input and converts into functional command for individual modules.

Implemented By: [Pot-pulator]

7.2.2 Pot Dropping Position Module (M4)

Secrets: Position of tray within dispensing area.

Services: Confirms tray position within dispensing area, triggers pot dispensing.

Implemented By: [Pot-pulator]

7.2.3 Verification Analysis Module (M11)

Secrets: System functional tolerances, serveral distance readings.

Services: Serves to check system output, and to alert if there is an output that does not conform to requirements.

Implemented By: [Pot-pulator]

7.2.4 Communication Module (M2)

Secrets: All subsystem states, user inputs.

Services: Serves be the communications between user and subsystems. Serves to coordinate action between subsystems.

Implemented By: [Pot-pulator]

8 Traceability Matrix

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

Req.	Modules
PDR1	M3
PDR2	M4
PDR3	M5
PDR7	M3
CR1	M7
CR2	M8
CR5	M7
TDR1	M9
TDR2	M9
TDR6	M9
VR1	M11
VR2	M12

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M8
AC3	M11
AC4	M2

Table 3: Trace Between Anticipated Changes and Modules

9 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.

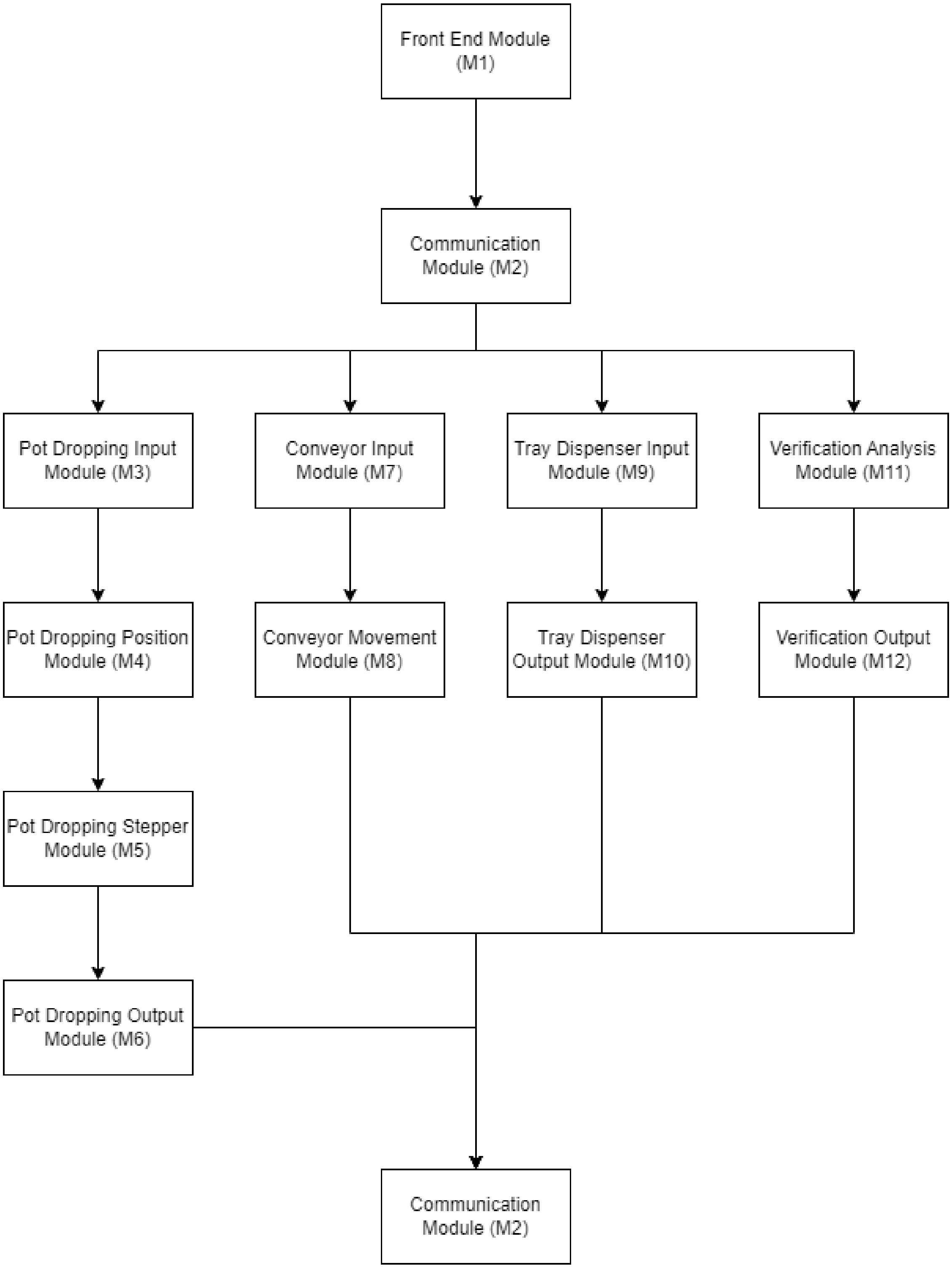

Figure 1: Use hierarchy among modules

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.