Module Guide for Sun Catcher

Sharon (Yu-Shiuan) Wu

December 20, 2019

1 Revision History

Date	Version	Notes
2019/11/25	1.0	First version update to Git
Date 2	1.1	Notes

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
Sun Catcher	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 (**?**). We advocate a decomposition based on the principle of information hiding (**?**). 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 **?**, 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 (**?**). 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. Section9 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 adopted here is called design for change.

AC1: The specific hardware on which the software is running.

AC2: The type of the input parameter. [The initial what of the input parameter? —SS][Changed —Author]

AC3: The constraint of the input data.

AC4: The name of input file.

AC5: The algorithm of the solar energy absorption.

AC6: The algorithm of the optimum tilt angle.

AC7: The algorithm of the calculating multiple optimum tilt angles with in a day period.

AC8: The equation of the sun intensity.

AC9: The structure of the data type day ADT.

AC10: The structure of the table-layout.

AC11: The name of the output file.

AC12: The structure of the data type sun catcher.

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 decisions 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: The goal of Sun Catcheris always predict the optimum title angle and the solar energy absorption and show the difference of the solar energy absorptions.

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 will actually be implemented.

M1: Hardware Hiding Modules

M2: Control Module

M3: Input/Read Parameters Module

M4: Input Verification Module

M5: Solar Energy Absorption Module

M6: Optimum Tilt Angle Module

M7: Calculation Module

M8: Day ADT Module

M9: Table-layout Module

M10: Sun Catcher Type Module

Level 1	**Level 2**
Hardware-Hiding Modules	
Behaviour-Hiding Module	Control Module
	Input/Read Parameters Module
	Input Verification Module
	Solar Energy Absorption Module
	Optimum Tilt Angle Module
	Calculation Module
Software Decision Module	Day ADT Module
	Table-layout Module
	Sun Catcher Type Module

Table 1: Module Hierarchy

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.

[The intention of this section is to document decisions that are made "between" the requirements and the design. To satisfy some requirements, design decisions need to be made. Rather than make these decisions implicit, they are explicitly recorded here. For instance, if a program has security requirements, a specific design decision may be made to satisfy those requirements with a password. In scientific examples, the choice of algorithm could potentially go here, if that is a decision that is exposed by the interface. —SS]

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by **?**. 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* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *Sun Catcher* means the module will be implemented by the Sun Catcher 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.

7.1 Hardware Hiding Modules (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

7.2 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: –

7.2.1 Control Module (M2) [Nothing is currently labelled as mC, so this reference does not work. —SS]

[Fixed —Author]

Secrets: The structure that modules connect to each other.

Services: Called the designed modules.

Implemented By: Sun Catcher

7.2.2 Input Parameters Module (M3)

Secrets: The type of the parameters [spell check —SS][Fixed —Author] of inputs.

Services: Assigns the input values to the designed parameters [spell check —SS][Fixed —Author].

Implemented By: Sun Catcher

7.2.3 Input Verification Module (M4)

Secrets: The constraint of the input values.

Services: Verifies [spell check! —SS][Sorry.. —Author] the input values.

Implemented By: Sun Catcher

7.2.4 Solar Energy Absorption Module (M5)

Secrets: The algorithm of calculating solar energy absorption.

Services: Calculate solar energy absorption.

Implemented By: Sun Catcher

7.2.5 Optimum Tilt Angle Module (M6)

Secrets: The algorithm of calculating optimum tilt angle and sun intensity.

Services: Calculates the optimum tilt angle and sun intensity.

Implemented By: Sun Catcher

7.2.6 Calculate Module (M7)

Secrets: The algorithm of calculating the sun intensity and zenith angle.

Services: Calculate sun intensity and zenith angle.

Implemented By: Sun Catcher

7.3 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 does not provide direct interaction with the user.

Implemented By: –

7.3.1 Day ADT Module (M8)

Secrets: The structure of defining day's type and the algorithm of calculating the day.

Services: Creates variables using day's type and calculates data in day's type.

Implemented By: Data.Time

7.3.2 Table-layout Module (M9)

Secrets: The structure and the content [spell check —SS] of the table.

Services: Display the data with the designed format. [spell check - there are many spelling mistakes in this document —SS][I am sorry about the spelling mistakes —Author]

Implemented By: Text.Layout.Table

7.3.3 Etc.

7.3.4 Sun Catcher Type Module (M10)

Secrets: The structure of defining the sun catcher's type.

Services: Creates variables using sun catcher's type.

Implemented By: Sun Catcher

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
R1	M1, M3, M2
R2	M4, M2
R3	M9, M2
R4	M6, M5, M2
R5	M6, M7, M2
R6	M5, M7, M2
R7	M9, M2
R8	M2
R9	M2

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M3
AC4	M3
AC3	M4
AC5	M5
AC6	M6
AC7	M2
AC8	M7
AC9	M8
AC10	M9
AC11	M9
AC12	M10

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. **?** 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.

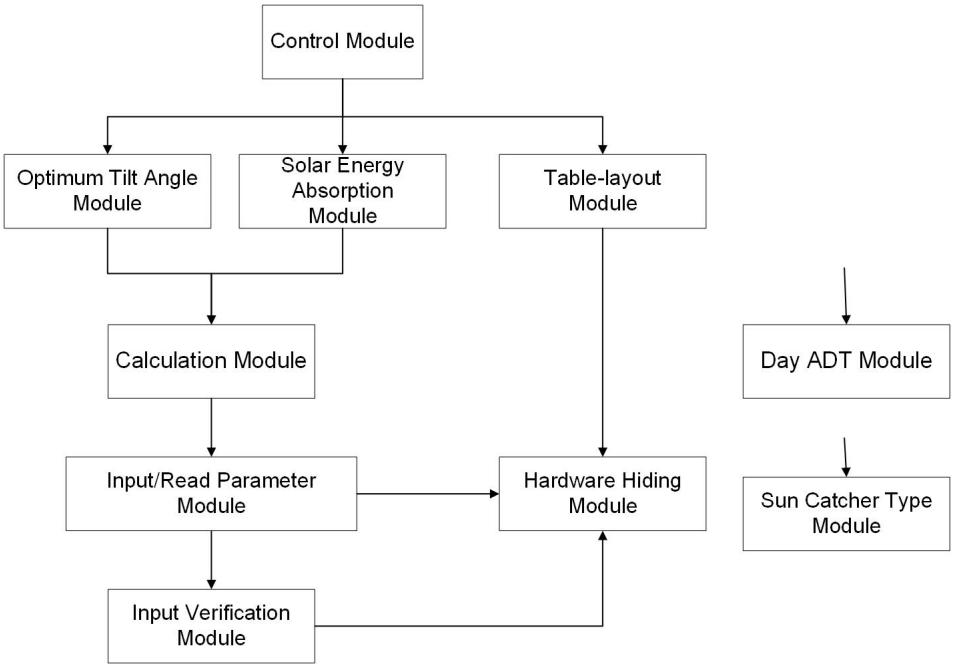

Figure 1: Use hierarchy among modules

[I suggest combining Read File with Input Parameters. Also, as it is, the Read File does not use Input Parameters, so there is not way for the Read File module to change the state of the inputs. Zenith Angle shouldn't really need to use Read File. If some of the Input Parameters are dates, then why doesn't the Input Parameter module use the Day Duration ADT? I also wonder if the Sun Intensity and Zenith Angle modules should be combined? —SS]

References

[Your references are not working. —SS]