Module Guide for Sun Catcher

Author Name

November 7, 2019

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
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

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 memberto 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. Theother 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 approachadapted here is called design for change.

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

AC2: The initial of the input parameter.

AC3: The constraint of theinput data.

AC4: The format of the input parameter.

AC5: The constraint of the output data.

AC6: The format of the output data.

AC7: How to calculate the output solar energy.

AC8: How to calculate the date duration.

AC9: How to calculate the optimum tilt angle.

AC10: The algorithm of sun intensity.

AC11: The algorithm of zenith angle.

AC12: The algorithm of sun declination angle.

AC13: The implementation for plotting the data.

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system ismore 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: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

UC2: The goal of Sun Catcheris always predict the title angle and the solar energy absorption.

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 Module

M2: Input Parameters Module

M3: Input Format Module

M4: Output Parameters Module

M5: Solar Energy Output Module

M6: Optimum Tilt Angle Module

M7: Sun Intensity Equation Module

M8: Zenith Angle Equation Module

M9: Sun Declination Equation Module

M10: Date Duration Module

M11: Plotting 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

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

Level 1	**Level 2**
Hardware-Hiding Module	
	Input Parameters Module
	Output Parameters Module
	Solar Energy Absorption Module
Behaviour-Hiding Module	Optimum Tilt Angle Module
	Sun Intensity Equation Module
Software Decision Module	Optimum Sun Intensity Equation Module
	Zenith Angle Equation Module
	Maximum Sun Intensity's Angle Module
	Sun Declination Module
	Date Duration Module
	Plotting Module
	Sequence Data Structure Module

Table 1: Module Hierarchy

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 haveto 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 Input Parameters Module (M2)

Secrets: The initialization of the input parameter.

Services: Verifies the user's input with the constraints of the input data and then converts the users' input into the initial input parameter.

Implemented By: Sun Catcher

7.2.2 Input Format Module (M3)

Secrets: The format and structure of the input data.

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: Sun Catcher

7.2.3 Output Parameters Module (M4)

Secrets: The initialization of the output parameter.

Services: Verifies the output from the related equation with the constraints of the output data and then converts the data into the initial output parameter.

Implemented By: Sun Catcher

7.2.4 Solar Energy Output Module (M5)

Secrets: The equation of calculate the solar energy output, using the data from input parameter module.

Services: Define the solar energy output equation, using the data from input parameter module.

Implemented By: Sun Catcher

7.2.5 Optimum Tilt Angle Module (M6)

Secrets: The equation of calculate the optimal tilt angle, using the datafrom input parameter module.

Services: Define the optimal tilt angle equation, using the data from input parameter module.

Implemented By: Sun Catcher

7.2.6 Etc.

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 do not provide direct interaction with the user.

Implemented By: –

7.3.1 Sun Intensity Equation Module (M7)

Secrets: The equation of calculate the sun intensity corresponding to thezenith angle of sun every day.

Services: Proves the algorithm calculate the initial sun intensity. Calculates the initial sun intensity by using govering equation,initial parameter, and the data provide from Zenith Angle Equation Module.

Implemented By: Sun.Intensity

7.3.2 Optimum Sun Intensity Equation Module (M??)

Secrets: The equation of calculate the sun intensity corresponding to thezenith angle of sun every day.

Services: Proves the algorithm calculate the optimal sun intensity. Calculates optimal sun intensity by using govering equation, the greatest point of sun intensity, and the data provide from Zenith Angle Equation Module.

Implemented By: Sun.Intensity

7.3.3 Maximum Sun Intensity's Angle Module (M??)

Secrets: The equation of calculate the sun intensity corresponding to thezenith angle of sun every day.

Services: Proves the algorithm calculate the angle correcponding sun intensity. Calculates angle corresponding the maximum sun intensity within the date duration.

Implemented By: Sun.Intensity

7.3.4 Zenith Angle Equation Module (M8)

Secrets: The equation of calculate the zenith angle corresponding to sun declination.

Services: Calculate the zenith angle using govering equation, and the data provide from Sun Declination Equation Module

Implemented By: Sun.ZenithAngle

7.3.5 Sun Declination Module (M9)

Secrets: The equation of calculate the sun declination corresponding to date within the date duration.

Services: Calculate the zenith angle using govering equation, and the dataprovide from Date Duration Equation Module

Implemented By: Sun.Declination

7.3.6 Date Duration Module (M10)

Secrets: The algorithm of calculate the date duration, using the data from input parameter module.

Services: Provides the date duration calculating function. Calculate the date duration by using the data from input parameter module.

Implemented By: Data.Time

7.3.7 Plotting Module (M11)

Secrets: The data structure and algorithm for plotting module.

Services: Provide the plot function.

Implemented By: Graphics.EasyPlot

7.3.8 Etc.

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, M??, M??, M??
R2	M??, M??
R3	M??
R4	M??, M??
R5	M??, M??, M??, M??, M??, M??
R6	M??, M??, M??, M??, M??, M??
R7	M??, M??, M??, M??, M??
R8	M??, M??, M??, M??, M??
R9	M??
R10	M??, M??, M??
R11	M??, M??, M??, M??

Table 2: Trace Between Requirements and Modules

AC	**Modules**
AC1	M1
AC4	M??
AC??	M??
AC??	M??
AC6	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC13	M??

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 ifcorrect 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 simplerbecause they use modules from the lower levels.

Figure 1: Use hierarchy among modules

References