Project Title: System Verification and Validation Plan for Sun Catcher

Sharon(Yu-Shiuan) Wu

November 22, 2019

1 Revision History

Date	Version	Notes
2019/11/04 1	1.0	First version of System VnV
Date 2	1.1	-

Contents

List of Tables

List of Figures

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
SC	Sun Catcher
Φ_P	the latitude of the solar panel
I_S	the intensity of the sun measured by the satellites
P_{A_h}	the height of the solar panel
P_{A_w}	the width of the solar panel
$\theta_{S_{\text{date}}}$	zenith angle of sun in the date of a sequence of day
$year_{\text{Start}}$	the year of the calcuation's starting date
$month_{\text{Start}}$	the month of the calcuation's starting date
day_{Start}	the day of the calcuation's starting date
$year_{\text{End}}$	the year of the calcuation's ending date
$month_{\text{End}}$	the month of the calcuation's ending date
day_{End}	the day of the calcuation's ending date

[symbols, abbreviations or acronyms – you can simply reference the SRS tables, if appropriate —SS]

This document provides an outline of the system verification and validation for the Sun Catcher. The general introduction section provides readers a summary of the functions in Sun Catcher and the related documents as the resources for testing. The plan section provides readers the plan for verifying and validating SC's software requirements specification(SRS) and introduces the method, tools, and external data to implement the testing. The system test description provides the readers with the test cases related to functional and nonfunctional requirements in Sun Catcher. The test builds for uncovering the errors and boosting the confidence of the software while ensuring an acceptable performance is provided.

[provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

3 General Information

3.1 Summary

The subsections below are the test cases of Sun Catcher. Sun Catcher is the software that calculates the optimum tilt angle of the days duration that decided by users. Then it takes the calculated angle to estimate the optimal solar energy output.

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

3.2 Objectives

The goal of the test is to build confidence in the software correctness and strengthen the robustness of the software. The functional and nonfunctional requirements of Sun Catcher ,and its related equations and constraints are found in the SRS [1]. The Verification and Validation Plan follows the requirements described in the SRS [1].

The object of the Verification and Validation Plan is:

To build the robustness of the system input.

To build the confidence of the correctness of the system output.

The functional requirements of Sun Catcher are:

R1: Sun Catcher should able to read the user's input value from the system's interface.

R2: Sun Catcher should able to verify the input by the input's constraint.

R3: Sun Catcher should able to output the result in the corrected format.

R4:Sun Catcher should able to verify the output by the output's constraint.

R5: Sun Catcher should able to determine the tilt angle.

R6: Sun Catcher should able to determine the solar energy output.

R7: Sun Catcher should able to verify the output by the output's constraint.

R8: Sun Catcher should able to show the comparison between the outputs corresponding to the different user's input.

R9: The system output should remain consistent with the corresponding of the input values.

The nonfunctional requirements of Sun Catcher are:

NFR1. Correct: The output should give correct results.

NFR2. Verifiable: The code is tested with a complete verification and validation plan.

NFR3. Understandable: The code is understandable for readers.

NFR4. Reusable: The code is modularized.

NFR5. Maintainable: The traceability between requirements, assumptions, theoretical models, general definitions, data definitions, instance models, likely changes, unlikely changes, and modules is completely recorded in traceability matrices in the SRS and module guide.

NFR6. Portable: The code can be run in different environments.

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: “build confidence in the software correctness,” “demonstrate adequate usability.” etc. You won’t list all of the qualities, just those that are most important. —SS]

3.3 Relevant Documentation

The SRS of the Sun Catchercan be found in [Wu [1]]

The external documents for verifying the equations used in Sun Catcher can be found in [Landau] [2] and [MarkandVijaysinh] [3]

[Reference relevant documentation. This will definitely include your SRS —SS]

4 Plan

4.1 Verification and Validation Team

The test team includes the following members:
Main reviewer: Sharon(Yu-Shiuan) Wu
Secondary reviewer: Deema Alomair, Bo Cao, Sasha Soraine, Zhi Zhang, and Doctor Smith
[Probably just you. :-) —SS]

4.2 SRS Verification Plan

- Get feedback from the reviewers: Sasha Soraine, Zhi Zhang, and Doctor Smith, after SRS is completed and put to the GitHub.
- Check the document by using SRS-Checklist and Writing-Checklist before publishing to GitHub.

[List any approaches you intend to use for SRS verification. This may just be ad hoc feedback from reviewers, like your classmates, or you may have something more rigorous/systematic in mind.. —SS]

4.3 Design Verification Plan

- The design should be verified by complete and success the test cases in the system VnV plan under section 5.
- The design should satisfy all the functional and nonfunctional requirements that stated in the SRS document [1].

[Plans for design verification —SS]

4.4 Implementation Verification Plan

The following tools will be used to facilitate testing:

> Rubber Duck Debugging: Performed by author, Sharon(Yu-Shiuan) Wu. The author should verbally explain the code line by line.
>
> Haskell Program Coverage: Dynamic Testing Tool, a tool-kit to record and display the code coverage of a Haskell Program. It aims to reinforce the correctness of the software and to eliminate the infeasibility problems.[Gill and Runciman] [4]
>
> QuickCheck: Automatic testing tool for Haskell programs, a library for random testing of program properties. It aims to boost the robustness of the software.[Claessen] [5]

[You should at least point to the tests listed in this document and the unit testing plan. —SS] [In this section you would also give any details of any plans for static verification of the implementation. Potential techniques including code walkthroughs, code inspection, static analyzers, etc. —SS]

4.5 Software Validation Plan

Sun Catchershould be valid by satisfied all the functional requirement in SRS plan.
Based on the physical concept of Sun Catcher, the author, Yu-Shiauan Wu, should record the actual solar energy by using the output from Sun Catcher. Then verify whether the calculated tilt angle can increase the energy gaining.

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

5 System Test Description

5.1 Tests for Functional Requirements

The subsection below is designed to cover the functional requirements of Sun Catcher, which also describes in section 3.2.
The test is divided into four subsections, which are input reading, input bounds, output calculation, and output verification. Input reading testing is designed for testing the ability to receive information from the software interface. Input bounds testing and output calculation testing are designed for testing the robustness of the software. Output verification testing is designed for the correctness of the implemented equation.

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS] [Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. —SS]

5.1.1 Input Reading

This test covers the requirements, R1, in section 3.2. Based on the SRS document[1], Sun Catcher has to identity users' inputs and then assign the values to designated equations or modules.

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

Identity users' input

1. InputReading-id1

   Control: Manual. Input the input value from the keyboard.

   Initial State: No input value

   Input: Input the value of requirements of Sun Catcher. It required to input the value of latitude, the area of the solar panel, the day started to estimate angle and day when the estimation end.

The given inputs are:
Φ_P: 43.250943 P_{A_h}:1455 P_{A_w}:665 $year_{\text{Start}}$:(2019) $month_{\text{Start}}$:(01) day_{Start}:(01) $year_{\text{End}}$:(2019) $month_{\text{End}}$:(12) day_{End}:(31)

Output: The expected result will for the given inputs is:
Φ_P: 43.250943 P_{A_h}:1455 P_{A_w}:665 $year_{\text{Start}}$:(2019) $month_{\text{Start}}$:(01) day_{Start}:(01) $year_{\text{End}}$:(2019) $month_{\text{End}}$:(12) day_{End}:(31)

[The expected result for the given inputs —SS]

Test Case Derivation: The output is justified if the output value is equal to the corresponding input value. [Justify the expected value given in the Output field —SS]

How the test will be performed:

- Input the value from the keyboard following the instruction of the software interface.
- Verified the output showing on the screen by the test case derivation instruction.

5.1.2 Input Bounds

This test covers the requirements, R2, in section 3.2. Based on the SRS document[1], the input data constraints can be found in the table, Specification Parameter Values, under the section, Input Data Constraints.

The Robustness of Input Bounds

1. InputBounds-id2

   Control: Manual.

   Input the extreme ends of the value of latitude.
   Initial State: No input value
   Input: Based on SRS[1] of the Sun Catcher[1], the boundaried of latitude is $-90 \leq \Phi_P \leq 90$.
   Therefore, the given inputs is:

input (1) Φ_P: 90 input (2) Φ_P: -90 input (3) Φ_P: 91 input(4) Φ_P: -91

Output: The expected result will for the given inputs is:
output (1) Φ_P: 90 output (2) Φ_P: -90 output (3) Φ_P: Latitude is not greater than 90 output (4) Φ_P: Latitude is not less than -90

[The expected result for the given inputs —SS]

Test Case Derivation: The output is justified if the output value is equal to the corresponding input value. When the input value is over the boundary of latitude, the system activates the error handler.
[Justify the expected value given in the Output field —SS]

How the test will be performed:

- Input the value from the keyboard following the instruction of the software interface.
- Verified the output showing on the screen by the test case derivation instruction.

2. InputBounds-id3
Control: Manual. Input the extreme ends, valid date and invalid date of the value according to the Gregorian calendar.
Initial State: No any given value.

Input: Based on calendar, the given inputs is:
Input(1): $year_{\text{Start}}$:(0) $month_{\text{Start}}$:(0) day_{Start}:(0)
Input(2): $year_{\text{Start}}$:(-1) $month_{\text{Start}}$:(-1) day_{Start}:(-1)

Input(3): $year_{\text{Start}}$:(2020) $month_{\text{Start}}$:(02) day_{Start}:(29)

Input(4): $year_{\text{Start}}$:(2020) $month_{\text{Start}}$:(02) day_{Start}:(28)

Output:The expected result will for the given inputs is:
Output (1) 0,0,0 doesnot exist.
Output (2) -1,-1,-1 does not exist.
Output(3) 2020.02.29 does not exist

Output(4) 2020.02.28 exist

[The expected result for the given inputs —SS]

Test Case Derivation: The output is justified if the output value is equal to the corresponding input value. When the input does not exist in the calendar, the system activates the error handler.

[Justify the expected value given in the Output field —SS]

How the test will be performed:

- Input the value from the keyboard following the instruction of the software interface.
- Verified the output showing on the screen by the test case derivation instruction.

3. InputBounds-id4
   Control: Manual. Input the test case where the start date is greater than the end date. Input the test case where the end date is greater than the start date.

   Initial State: No any given value.
   Input: Based on the calendar, the given inputs is:
   Input (1) $year_{\text{Start}}$:(2020) $month_{\text{Start}}$:(02) day_{Start}:(28) $year_{\text{End}}$:(2021) $month_{\text{End}}$:(02) day_{End}:(28)

   Input (2) $year_{\text{Start}}$:(2020) $month_{\text{Start}}$:(02) day_{Start}:(28) $year_{\text{End}}$:(2019) $month_{\text{End}}$:(02) day_{End}:(28)

   Input (3) $year_{\text{Start}}$:(2020) $month_{\text{Start}}$:(02) day_{Start}:(28) $year_{\text{End}}$:(2020) $month_{\text{End}}$:(02) day_{End}:(28)

   Output: The expected result will for the given inputs is:
   Output (1) Valid
   Output (2) Invalid

Output (3) Valid

[The expected result for the given inputs —SS]

Test Case Derivation: When the ending date is less than the start date, the system activates the error handler.

[Justify the expected value given in the Output field —SS]

How the test will be performed:

- Input the value from the keyboard following the instruction of the software interface.
- Verified the output showing on the screen by the test case derivation instruction.

5.1.3 Output Calculation

This test covers the requirements, R3 to R5, in section 3.2. This test relates to the previous test input reading testing. After the system reads the inputs from the software interface, the system starts calculating the outputs.

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

The Robustness of the Calculation

1. CalculateOutput-id5
   Control: Automatic. Input the extreme ends of the value of latitude.

   Initial State: Based on the assumption in SRS[1], $I_S = 1.35$, the input Φ_P is a valid value.

   Input: Input the value of requirements of Sun Catcherthat drive the calculation of the Solar intensity. To calculate the solar intensity, it needs the latitude. of the panel Φ_P

The given inputs are:
input (1) Φ_P: 90 input (2) Φ_P: -90 input (3) Φ_P: 0

Output: The expected result will for the given inputs is:
Output (1) Valid Output (2) Valid Output (3) Valid

[The expected result for the given inputs —SS]

Test Case Derivation: Based on the output constraint describes in SRS [1], the output tilt angle should be $-90 \leq Output \leq 90$ [Justify the expected value given in the Output field —SS]

How the test will be performed:

- Input the input values from a file, CalculateOutputid5.txt.
- Calculate the zenith angle corresponding to the inputs.
- Calculate the solar intensity corresponding to the zenith angle.
- Calculate the output corresponding to the solar intensity.
- Verify the output by using the Test Case Derivation.
- Test case pass if all the output is valid, otherwise the test case fail.

5.1.4 Output Verification

This test covers the requirements, R2, R7 and R8, in section 3.2. This test uses external data3.3 to verify the output. Based on the SRS document[1], the output data constraints can be found in the table, Output Variables, under the section, Properties of a Correct Solution.

The Correctness of the Calculation

1. VerifyOutput-id6

   This test case used the external data from [Jacobsonand Jadhav] [3] as the expected output and the expected input latitude.

Control: Automatic. The test cases contain cases that Φ_P = expected input latitude and $\Phi_P \neq$ expected input latitude.

Initial State: Based on the assumption in SRS[1], I_S:1.35, and based on the assumption in [Jacobson and Jadhav] [3], $year_{\text{Start}}$: 2018 $month_{\text{Start}}$: 01 day_{Start}: 01; $year_{\text{End}}$: 2018 $month_{\text{End}}$: 12 day_{End}: 31

Input: Input the value of requirements of Sun Catcherthat drive the calculation of the optimal tilt angle. To calculate the solar intensity, it needs $\theta_{S_{date}}$, which is driven by the input latitude.

The given inputs are:
Input (1)Φ_P: 64.13
Input(2)Φ_P:63.13

Output: The expected result wil for the given inputs is the optimal tilt angle:
Output (1) : 43
Output (2) : 43

[The expected result for the given inputs —SS]

Test Case Derivation: Based on the equation described in SRS[1], we get the **actual result**. Then we calculate the relative error using the data in the [JacobsonandJadhav][3] as our **expected result**. Therefore, relative error ≈ 0 where relative error = $|1 - \frac{\text{actual result}}{\text{expected result}}|$

[Justify the expected value given in the Output field —SS]

How the test will be performed:

- Build a linear graph using the expected input latitude as the x-axis and expected output as the y-axis.
- Input the input values from a file, VerifyOutputId6.txt.
- Calculate the **actual result** by the equation descibes in in SRS[1]
- Place the point($P_{\text{actual input}}$)(x-axis: input latitude, y-axis: actual result) in the linear graph
- Find the point($P_{\text{upper bound}}$),the lowest upper bound of $P_{\text{actual input}}$ and point($P_{\text{lower bound}}$), the greatest upper bound of $P_{\text{actual input}}$

- Calculate the area between $P_{\text{upper bound}}$ and $P_{\text{actual input}}$; and $P_{\text{lower bound}}$ and $P_{\text{actual input}}$ using the equation,
$Area = \frac{|(x_{\text{input latitude}} - x_{\text{expected latitude}})| \times |(y_{\text{actual result}} - y_{\text{expected result}})|}{2}$
- If $Area_{\text{actual input - \textbf{upper bound}}} < Area_{\text{actual input - \textbf{lower bound}}}$, then **expected result** = the y-axis of $P_{\text{upper bound}}$, otherwise **expected result** = the y-axis of $P_{\text{lower bound}}$
- Verified the output by the test case derivation instruction.
- If all the relative error of the test cases is approximately 0, then the test success, otherwise the test fails.

2. VerifyOutput-id7

This test case used the external data from [Landau][2] as the expected output, expected input latitude

Control: Automatic. The test cases contain cases that Φ_P = expected input latitude, $\Phi_P \neq$ expected input latitude.

Initial State: Basedon the assumption in SRS[1], I_S: 1.35, and based on the assumption in [Landau][2], the days duration of the winter in northern hemisphere is from
$year_{\text{Start}}$: 2018 $month_{\text{Start}}$: 10 day_{Start}: 05
to $year_{\text{End}}$: 2019 $month_{\text{End}}$: 03 day_{End}: 05

Input: Input the value of requirements of Sun Catcherthat drive the calculation of the Solar intensity. To calculate the solar intensity, it needs $\theta_{S_{date}}$, which is driven by the input latitude.

The given inputs are:
Input (1)Φ_P: 30
Input (2)Φ_P: 31

Output: The expected result wil for the given inputs is average the solar intensity during winter:
Output (1) : 5.6
Output (2) : 5.6

[The expected result for the given inputs —SS]

Test Case Derivation: Based on the equation described in SRS[1], we get the expected result. Therefore, relative error ≈ 0 where relative error $= |1 - \frac{\text{actual output}}{\text{expected output}}|$

[Justify the expected value given in the Output field —SS]

How the test will be performed:

- Input the input values from a file, VerifyOutputId7.txt.
- Calculate the daily solar intensity during winter
- Calculate the average solar intensity during winter, using the equation,
  the average solar intensity $= \frac{\text{the sum of the daily solar intensity}}{\text{days duration of winter}}$
- Output the average solar intensity as the actual output
- Verified the output by the test case derivation instruction.
- If all the relative error of the test cases is approximately 0, then the test success, otherwise the test fails.

5.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. —SS] [Tests related to usability could include conducting a usability test and survey. —SS]

5.2.1 Correctness

The correctness of the System

1. correctness-id1

   Type: Dynamic analysis The outputs of the system test under section 5.1.4. To get a reliable output, Sun Catcheris expected to pass all the test cases under section 5.1.4.

   How the test will be performed: Active the test cases under section 5.1.4, when the code is modified.

2. correctness-id2

   Type: Dynamic analysis - Code Coverage

   Use the code coverage tool, Haskell Program Coverage(HPC), to test the code coverage. The description of HPC can be found in section 4.3.

   Input/Condition: The main code of Sun Catcher

   Output: The percentage of the coverage

   How the test will be performed:

   - Use the compiler, ghc-6.8.1 or later version, to active Haskell Program Coverage.
   - Enable hpc with the command line, -fhpc.
   - The test case success, if the output gets the 100otherwise the test case fails.

5.2.2 Portability

The portability of the system

1. portability-id3

   Type: Manual

   Initial State: -

   Input/Condition: Implement Sun Catcherin diverse environments.

   Output/Result: If the Sun Catcherworks functionally.

   How the test will be performed:

   - Implement Sun Catcher on the virtual machines with the system environment of Windows 10, macOS.
   - Run every function of Sun Catcher.
   - If function works, then success, otherwise fail.

5.3 Traceability Between Test Cases and Requirements

	R1	R2	R3	R4	R5	R6	R7	R8	R9
id1	X								
id2		X							
id3		X							
id4		X							
id5				X					
id6			X		X			X	X
id 7			X			X		X	X

Table 1: Traceability Between Functional Requirements Test Cases and Requirements

	NFR1	NFR2	NFR3	NFR4	NFR5	NFR6
id1	X					
id2	X					
id3						X

Table 2: Traceability Between NonFunctional Requirements Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

References

[1] Yu-Shiuan Wu. Software requirements speci
cation for sun catcher, 2019. URL https://github.com/sharyuwu/optimum-tilt-of-solar-panels/blob/master/docs/SRS/SRS.pdf.

[2] Charles R. Landau. Optimum tilt of solar panels, 2001. URL https://www.solarpaneltilt.com/#other.

[3] Mark Z. Jacobson and Vijaysinh Jadhav. World estimates of pv optimal tilt angles and ratios of sunlight incident upon tilted and tracked pv panels relative to horizontal panels. Technical Report CAS-17-01-SS, Department of Civil and Environmental Engineering, Stanford University, Stanford,USA, 2018. URL https://web.stanford.edu/group/efmh/jacobson/Articles/I/TiltAngles.pdf.

[4] Andy Gill and Colin Runciman. Haskell program coverage, 2000. URL https://wiki.haskell.org/Haskell_program_coverage.

[5] Koen Claessen. Quickcheck: Automatic testing of haskell programs, 2000. URL http://hackage.haskell.org/package/QuickCheck.

6 Appendix

6.1 Symbolic Parameters

Symbol	Description	Value
I_S	Solar insensity	1.35

6.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]

Symbol	Answer
Do you think this software helps?	Yes/ No
Do you think this software is easy to use?	Yes/ No
Do you think this software is easy to use for elders?	Yes/ No
Do you think this software is easy to use for children(under 12)?	Yes/ No
Do you think this software works flawlessly?	Yes/ No
How many time you open this software in a week?	________times
How many stars would you like to give to this software?(1 - 10 starts)	________starts
Do you like to recommend this software to others?	Yes/ No

Table 3: Survey for home users

Symbol	Answer
The steps take to get the optimum tilt angle	_________steps
How long it need to get the optimum tilt angle	_________times
The steps take to get the expected solar energy gaining	_________steps
How long it need to get the expected solar energy gaining	_________times
The power consume after using the software for an hour	_________%
The memory consume after install the software	_________%
How many error alerts you take at the fist time using the software?	_________times
How many error alerts you take at the second time using the software?	_________times

Table 4: Survey for researching group