System Verification and Validation Plan for Solar Cooker Energy Calculator

It isn't a high priority, but maybe you can come up with a better name for your software?

Deesha Patel

February 23, 2023

1 Revision History

Date	Version	Notes
February 14, 2023	0.1	Add General Information section
	0.2	Add further details in different sections
February 18, 2023	0.3	First Draft of VnV
February 22, 2023	1.0	Updates done according to issues

Contents

List of Tables

consistent capitalization

2 Symbols, Abbreviations and Acronyms

symbol	description
MG	Module Guide
MIS	Module Interface Specification
SRS	Software Requirement Specification
SCEC	Solar Cooker Energy Calculator
TC	Test Case
VnV	Verification and Validation

For complete symbols used within the system, please refer the section 1 in SRS document.

This document provides the road-map of the verification and validation plan for Solar Cooker Energy Calculator for ensuring the requirements and goals of the program (found in SRS document). The organization of this document starts with the General Information about the Solar Cooker Energy Calculator in section 3. A verification plan is provided in section 4 and section 5 describes the system tests, including tests for functional and non-functional requirements.

3 General Information

3.1 Summary

This document reviews the validation and verification plan for Solar Cooker Energy Calculator (SCEC), a program that calculate the balance temperature at recipient and cooking power in it using user inputs.

3.2 Objectives

The purpose of the validation plan is to define how system validation will perform at the end of the project. The strategy will use to assess whether the developed system accomplishes the designed goals. Also, the verification plan includes test strategies, definitions of what will be tested, and a test matrix with detailed mapping connecting the tests performed to the system requirements. This verification plan ensures that all requirements specified in the System Requirements Specification(SRS) document have been met and reviewed.

3.3 Relevant Documentation

The relevant documentation for the SCEC includes Problem Statement, System Requirements Specifications, VnV Report, MG and MIS (found in Github Repository).

4 Plan

This section describes the testing plan for the Solar Cooker Energy Calculator system. The planning starts with the Verification and Validation team,

followed by the SRS verification plan, design verification plan, implementation verification plan, Automated testing and verification tools, and Software validation plan.

what section?

4.1 Verification and Validation Team

This section describes the members of Verification and Validation plan.

Name	Document	Role	Description
Dr. Spencer Smith	All	Instructor/ Reviewer	Review the documents, design and documentation style.
Deesha Patel	All	Author	Create and manage all the documents, create the VnV plan, perform the VnV testing, verify the implementation.
Mina Mahdipour	All	Domain Expert Reviewer	Review all the documents.
Karen Wang	SRS	Secondary Reviewer	Review the SRS document
Lesley Wheat	VnV Plan	Secondary Reviewer	Review the VnV plan.
Sam Joseph Crawford	MG + MIS	Secondary Reviewer	Review the MG and MIS document.

Table 1: Verification and Validation team

4.2 SRS Verification Plan

The SCEC SRS document shall be verified in the following way:

1. Initial review from the assigned members (Dr. Spencer Smith, Mina Mahdipour, Karen Wang, and Deesha Patel) will be performed. For this, the manual review will perform using the given SRS Checklist, designed by Dr. Smith.

You should look to see whether there is anything that should be added to the checklist

2. Reviewer can give feedback to the author by creating an issue in Github.

3. Author (Deesha Patel) is responsible to address the issues created by the primary and secondary reviewers. Also, need to address the suggestions given by the instructor (Dr. Spencer Smith).

4.3 Design Verification Plan

The design documents, Module Guide (MG), and Module Interface Specification (MIS) will be verified through the static technic of document inspection by the domain/ primary expert (Mina Mahdipour) and secondary reviewer (Sam Joseph Crawford). Also, the class instructor (Dr. Spencer Smith) will review both documents. Reviewers can give feedback to the author by creating the issue in Github. The author is responsible to solve the issues and address the suggestions. The reviewer will assess this document with the help of MG Checklist and MIS Checklist designed by Dr. Smith.

4.4 Verification and Validation Plan Verification Plan

By following the table 1, the verification and validation plan will be written and validated by Author (Deesha Patel), then Domain expert (Mina Mahdipour) and Secondary reviewer (Lesley Wheat) will review it and give suggestion by creating an issue on GitHub. Once done, Instructor will do final review of the VnV plan. The reviewer will assess this document by VnV Checklist defined by Dr. Smith.

4.5 Implementation Verification Plan

The implementation of SCEC shall be verified in the following ways:

- Static testing for SCEC:
  - Code Walkthrough: This process will be performed by the author (Deesha Patel) and Domain expert (Mina Mahdipour). An author will share the copy of the original code with Domain expert and then Domain expert will manually test the code with different test cases. The Domain expert will raise the issue in GitHub if they finds any issue with the code.

Have you checked with Mina?

This isn't how a code walk through works. I think this is a good idea, but you need to do a bit of reading on code walkthroughs. Add a citation with information on code reviews.

- Dynamic testing for SCEC:
  - Test cases: Test cases for all the mentioned tests in section 5 will be carried out. These tests target functional and non-functional requirements listed in the SRS document (section 5.1 and 5.2). ~~All the test cases are manual or automatic~~.

4.6 Automated Testing and Verification Tools

System and Unit tests: Automated testing of SCEC is conducted using the Pytest library in Python. These tests are performed by predetermining user inputs and comparing them with expected values.

provide a link

If you aren't using Continuous Integration, you can make that explicit here.

4.7 Software Validation Plan

Software validation plan is beyond the scope for SCEC System as this is not the whole system ~~which~~ that can perform checking the temperature of whole solar box.

I agree you cannot do validation, but not with your reason. You aren't doing validation because you don't have experimental data.

5 System Test Description

5.1 Tests for Functional Requirements

Functional requirements for SCEC are given in SRS section 5.1. Some input values are taken from the paper (1). There are five functional requirements for SCEC, R1 and R2 are related to the inputs, while R3 to R5 are corresponding to outputs. section 5.1.1 describes the input tests related to R1 and R2; and section 5.1.2 describes the output tests for R3 to R5.

5.1.1 Input tests

Functional tests - Input tests - Area of object

1. test-id1: Valid Area inputs

   Control: Automatic

	Input			Output	
ID	A_t	A_{ref}	A_m	*valid?*	*ErrorMessage*
TC-SCEC-1-1	0.039	0.046	0.064	Y	NONE
TC-SCEC-1-2	0	0.037	0.059	N	Non-zero required
TC-SCEC-1-3	0.67	0.0942	0	N	Non-zero required
TC-SCEC-1-4	0.741	0	0.0424	N	Non-zero required
TC-SCEC-1-5	-0.063	0.728	0.572	N	Positive value required
TC-SCEC-1-6	0.025	-0.279	0.763	N	Positive value required
TC-SCEC-1-7	0.025	0.279	-0.763	N	Positive value required
TC-SCEC-1-8	1000	0.245	0.562	N	Too long area input
TC-SCEC-1-9	0.562	0.285	0.13f	N	Numeric values only
TC-SCEC-1-10		0.285	0.13f	N	Empty value not accepted

Table 2: TC-SCEC-1 - Area input constraints tests

Initial State: Pending input

Input: Set of input values for area of particular object given in the Table 2.

Output: Either give an appropriate error message for TC-SCEC-1-2 to TC-SCEC-1-10, or produces calculated temperature values as an output defined in the Table 2.

Test Case Derivation: This test case is to test the behaviour of the system when the system is supplied with inputs for area that are the physical constraints of Solar cooker box. In test cases TC-SCEC-1-2 to TC-SCEC-1-9, the system produces the error message, as those are invalid inputs.

How test will be performed: The automatic test is performed using PyTest.

Functional tests - Input tests - Temperature value

1. test-id1: Valid/Invalid Temperature value

	Input					Output	
ID	T_t	T_{g_2}	T_f	T_{init}	T_{ref}	valid?	Error Message
TC-SCEC-2-1	30	30.2	32.5	40.1	41.3	Y	NONE
TC-SCEC-2-2	0	24.5	41.3	24.1	51.3	N	Non-zero required
TC-SCEC-2-3	12.1	24.6	56.2	43.2	-13.4	N	Positive temperature required
TC-SCEC-2-4	23.5	26.4	26.6	36.2		N	Empty temperature value
TC-SCEC-2-5	24.5	26.8	210.3	25.7	29.4	N	Exceed temperature value

Table 3: TC-SCEC-2 - Temperature input constraints tests

Control: Automatic

Initial State: Pending input

Input: Pass the value of temperature specified input column in the Table 3.

Output: verify the output of the software matches the output column specified in Table 3.

Test Case Derivation: This test case is to test the behaviour of the system when the system is supplied with inputs for temperature. In test cases TC-SCEC-2-2 to TC-SCEC-2-5, the system produces the error message, as those are invalid inputs.

How test will be performed: The automatic test is performed using PyTest.

	Input		Output	
ID	ϵ_{ref}	ϵ_t	valid?	Error Message
TC-SCEC-3-1	1	0.95	Y	NONE
TC-SCEC-3-2	0.97	0.91	Y	NONE
TC-SCEC-3-3	0.93	1.3	N	Not in range
TC-SCEC-3-4	-0.75	0.86	N	Not in range

Table 4: TC-SCEC-3 - Other input constraints tests

Functional tests - Input tests - other parameters

1. test-id1: Valid/Invalid Emittance value of object

   Control: Automatic

   Initial State: Pending input

   Input: Pass the value of emittance specified input column in the Table 4.

   Output: verify the output of the software matches the output column specified in Table 4.

   Test Case Derivation: This test case is to test the behaviour of the system when the system is supplied with inputs for emittance. In test cases TC-SCEC-3-3 to TC-SCEC-3-4, the system produces the error message, as those are invalid inputs.

   How test will be performed: The automatic test is performed using PyTest.

5.1.2 Output tests

1. test-id1: Validate the output of the fluid temperature in recipient

   Control: Automatic

   Initial State: N/A

   Input: Pass the input values:
   $A_m = 1.5$
   $A_t = 0.0201$
   $A_{ref} = 0.0058$
   $T_{g2} = 30$
   $T_t = 30$
   $T_f = 30$
   $T_{ref} = 30$
   $\epsilon_r = 1$
   $\epsilon_t = 0.85$

   Output: Below output should be generated for each of the valid and real inputs.

- $T_r \approx 95.9$.
- For other inputs, it should calculate the temperature value of the fluid
- Graph should be generated with the temperature of the fluid and recipient.

Test Case Derivation: This test case is to test the output of the system when the system is supplied with all valid inputs. This test case is derived from 3rd and 4th requirement in SRS document.

How test will be performed: The automatic test is performed using PyTest.

2. test-id2: Validate the output of the energy temperature in fluid

   Control: Automatic

   Initial State: N/A

   Input: Pass the input value: $T_{init} = 30$

   Output: As an output, algorithm should calculate the non-negative and non-zero temperature energy value of the fluid.

   Test Case Derivation: This test case is to test the output of the system when the system is supplied with initial temperature of the fluid. This test case is derived from 5th requirement in SRS document.

   How test will be performed: The automatic test is performed using PyTest.

5.2 Tests for Nonfunctional Requirements

Non-Functional requirements for SCEC are given in SRS section 5.2. There are five non-functional requirements for SCEC.

5.2.1 Non-functional: Understandability

Understandability

1. test-id1: Understandability test

There are more potential code quality checks. There are coding standards that can be checked. You could google code review checklist. Since you are using Python, I highly recommend using flake8.

Type: Manual

Initial State: None

Input/Condition: None

Output/Result: By review over the code quality, more quality is achieved.

How test will be performed: Domain Expert (Mina Mahdipour) will review the shared code and complete the survey mentioned in the Table 5.

No.	Question	Score(0-10)
1.	The whole code indented properly to understand the code.	
2.	The names of the variables and methods is meaningful.	
3.	Different comments is useful to understand the importance of code.	
4.	Particular function perform single and understandable tasks.	
5.	Overall quality.	

ambiguous

ambiguous

Table 5: TC-SCEC-4 - Understandability test survey

5.2.2 Non-functional: Maintainability

Maintainability

1. test-id1: Maintainability

   Type: Code walkthrough

   Initial State: None

   Input: None

   Output: Walkthrough Meeting helps to improve the maintainability of the system and all documentations.

You need to be more specific on how you will run your code walkthrough

How test will be performed: During code walkthrough meeting all the details related to the software lifespan, coupling of the software architecture, documentation of the software will discussed among team members listed in Table 1.

5.2.3 Non-functional: Usability

Usability

1. test-id1: Usability

   Type: Manual with group of people

   Initial State: None

   Input: None

   Output: Survey can help to know about the user perspective towards the system.

   How test will be performed: The user group will be asked to install the software on their system and give input on their own. Then user need to fill out the short answer survey given in the Table 6.

No.	Question	Answer
1.	Which operating system are you using?	
2.	Is system running smoothly on your computer?	
3.	Is invalid input's message clear?	
4.	Is software easy to use?	
5.	Is text easy to read?	
6.	What, if anything, surprised you about the experience?	
7.	What did you like the least?	
8.	Do you have any suggestion?	

Table 6: TC-SCEC-5 - Usability test survey

Ask them to assess (measure) on a Likert scale.

5.2.4 Non-functional: Portability

Portability

1. test-id1: Portability

   Type: Manual

   Initial State: None

   Input: None

   Output: Successful running system over all platform give confidence about portability of the software.

   How test will be performed: Code developer (Deesha Patel) will try to install and run whole software in different operating systems. Also, need to ensure that all the given test cases pass in all different operating system.

regression testing

5.3 Traceability Between Test Cases and Requirements

A traceability between test cases and requirements is shown in Table 7

	R1	R2	R3	R4	R5	NFR1	NFR2	NFR3	NFR4
5.1.1	X	X							
5.1.2			X	X	X				
5.2.1						X			
5.2.2							X		
5.2.3								X	
5.2.4									X

Table 7: Tracebility between Test cases and Requirements

6 Unit Test Description

This section is intensionally blank until MIS complete.

References

[1] Hilario Terres, Arturo Lizardi, Raymundo Lpez, Mabel Vaca, and Sandra Ch˚vez. Mathematical model to study solar cookers box-type with internal reflectors. *Energy Procedia*, 57:1583–1592, 2014. 2013 ISES Solar World Congress.