System Verification and Validation Plan for SCEC (Solar Cooker Energy Calculator)

Deesha Patel

April 19, 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

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 (SCEC) for ensuring the requirements and goals of the program (found in the 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 nonfunctional requirements.

3 General Information

3.1 Summary

This document reviews the validation and verification plan for Solar Cooker Energy Calculator (SCEC). SCEC is a program that calculate the balance temperature of recipient and fluid. The cooking power is also calculated 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 design 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 which identifies the proposed idea, System Requirements Specifications which provides information about the requirement of the proposed system, VnV Report for validation and verification, MG and MIS design documents (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 in section 4.1, followed by the SRS verification plan (section 4.2), design verification plan (section 4.3), implementation verification plan (section 4.5), Automated testing and verification tools (section 4.6), and Software validation plan (section 4.7).

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.

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 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 final copy of the original code with Domain expert before the meeting and then Domain expert will manually go through important test cases by hand in the meeting. The Domain expert will share and discuss her own findings. Also, she can ask any questions in meeting.
- 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 nonfunctional requirements listed in the SRS document (section 5.1 and 5.2).

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. Continuous Integration will not be used.

4.7 Software Validation Plan

Software validation plan is beyond the scope for SCEC system as we do not have enough experimental data for the validation of the system behavior.

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 5.1.4 describes the output tests for R3 to R5.

5.1.1 Input Tests - test_invalid_input

Functional tests - Input tests - Area of object

	Input (m^2)			Output	
ID	A_t	A_{ref}	A_{m}	valid?	Error Message
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 large 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

1. test-input-area-id1: Valid Area inputs

   Control: Automatic

   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

	Input (°C)				Output	
ID	T_t	T_{g2}	T_f	T_{ref}	valid?	Error Message
TC-SCEC-2-1	30	30.2	32.5	41.3	Y	NONE
TC-SCEC-2-2	0	24.5	41.3	51.3	N	Non-zero required
TC-SCEC-2-3	12.1	24.6	56.2	-13.4	N	Positive temperature required
TC-SCEC-2-4	23.5	26.4	26.6		N	Empty temperature value
TC-SCEC-2-5	24.5	26.8	210.3	29.4	N	Excessive temperature value

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

1. test-input-temperature-id2: Valid/Invalid Temperature value

   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.

Functional tests - Input tests - other parameters

1. test-input-emittance-id3: Valid/Invalid Emittance value of object

   Control: Automatic

   Initial State: Pending input

	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

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 Loading Tests - test_load_params

1. test-load-id4: Validate the input are read correctly for further process.

   Control: Automatic

   Initial State: N/A

   Input: Name of the file. Sample input file is shown in Table 8 .

   Output: verify all the inputs are there. If not it will give an appropriate missing value message.

   Test Case Derivation: This test case is to test the behaviour of the system to fetch the inputs from the input file.

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

5.1.3 Intermediate Tests - test_temperature_calculation and test_energy

1. test-fluid-temp-id5: Validate the output of the fluid temperature in recipient

   Control: Automatic

   Initial State: N/A

Input: Pass the input values:
$A_m = 1.5$ m^2
$A_t = 0.0201$ m^2
$A_{\text{ref}} = 0.0058$ m^2
$T_{g2} = 30$ °C
$T_t = 30$ °C
$T_f = 30$ °C
$T_{\text{ref}} = 30$ °C
$\epsilon_r = 1$
$\epsilon_t = 0.85$

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

- $T_r \approx 100.7$ °C.
- For other inputs, it should calculate the temperature value of the fluid.
- Graph should be generated with the temperature of the fluid and recipient.
- The relative error will be calculate to match the expected solution of the system.

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-fluid-energy-id6: Validate the output of the energy temperature in fluid

   Control: Automatic

   Initial State: N/A

   Input: Pass the input value: $T_{\text{init}} = 30$ °C

   Output: As an output, algorithm should calculate the non-negative and non-zero temperature energy value of the fluid. Tentative output of the fluid energy is 1000 watt.

   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.

5.1.4 Output Tests

1. test-output-temp-id7: Validate the final result is not negative value of temperature

   Control: Automatic

Initial State: N/A

Input: Calculated temperate

Output: If the input temperature is negative, it will generate an error message about negative temperature value.

Test Case Derivation: This test case is to test the final output of the system when the system have calculated temperature values of the fluid.

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

2. test-output-file-id8: Validate the final result is stored in the outputfile.

   Control: Manual

   Initial State: N/A

   Input: None

   Output: Test case will be pass if the output file is generated, otherwise it will be failed

   Test Case Derivation: This test case is to test the final output of the system is stored in the output file or not.

   How test will be performed: Manually check the file exist or not.

5.2 Tests for Nonfunctional Requirements

NonFunctional requirements for SCEC are given in SRS section 5.2. There are five nonfunctional requirements for SCEC.

5.2.1 Nonfunctional: Understandability

Understandability

1. test-id9: Understandability test

   Type: Automatic

   Initial State: None

   Input/Condition: None

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

   How test will be performed: Flake8 will be used to check the code quality. This library can check the quality of code against different coding styles such as unused libraries and undefined variable.

5.2.2 Nonfunctional: Maintainability

Maintainability

1. test-id10: Maintainability

   Type: Code walkthrough

   Initial State: None

   Input: None

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

   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 Nonfunctional: Usability

Usability

1. test-id11: 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 scale of the question given in the survey Table 5.

5.2.4 Nonfunctional: Portability

Portability

1. test-id12: Portability

   Type: Manual

   Initial State: None

   Input: None

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

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 5: TC-SCEC-5 - Usability test survey

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 regression testing that means all the given test cases pass in all different operating system.

5.3 Traceability Between Test Cases and Requirements

A traceability between test cases and requirements is shown in Table 6

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

Table 6: Tracebility between test cases and requirements

6 Unit Test Description

The source code for the SCEC software has following modules,

- Calculation.py - Provides function to calculate temperature of fluid, reflector, glass 1, glass 2, and lid.
- Constant.py - Provides all the constant values to the other modules.
- Energy_Calculation.py - Calculate the energy of fluid.
- load_params.py - Module for load the input parameters from the input file.
- main.py - Main module to control all other modules.
- parameters.py - A data structure for input parameters.
- plot.py - A module for plotting the result in a graph and export result in file.
- verify_params.py - A module to verify input values.

6.1 Unit Testing Scope

Unit testing is performed for the following modules:

- Energy
- Inputs
- Load Parameters
- Temperature Calculation

These modules are high priority modules that can affect the whole system if not work properly. Other modules like Constant and Parameters modules are tested in the system as those have low priority static modules. Also, the plotting library plot the result with the given inputs. So, our system do not perform unit test on that module.
Intermediate results for calculating energy and temperature can be tested. However, ODE is calculated using the Scipy library. So, our system is not validating the result of the calculated ODEs.
The whole system follows the paper (1). This paper do not contains all the information about inputs and outputs. So, our system is not able to compare the final result of the system with original outputs and can not calculate relative error. However, unit test is able to check that the output module is working properly and provide appropriate output.

6.2 Tests for Functional Requirements

These section follows the functional tests those are not performed using the PyTest automated testing. The main target is to address the modules given in the MG and MIS document.

6.2.1 Constant Value Module (M2)

A Constant Value Module is assigned to a object that stores the constant values used in the system, manual test is performed by checking all the values are present in this module.

1. test-constant-id13

   Type: Automatic

   Initial State: N/A

   Input/Condition: Constant value file called constant.py

   Output/Result: Pass, if all the values are present in the module and Fail, otherwise.

   How test will be performed: Run the constant test file to check the all the missing parameter.

6.2.2 Temperature ODE Module (M8)

A Temperature ODE Module is use to calculate the temperature of fluid, reflector, lid, glass 1 and glass 2 over time.

1. test-temp-ode-id14

   Type: Manual/Automatic

   Initial State: N/A

   Input/Condition: Calculation file called calculation.py

   Output/Result: Pass, if this module is able to calculate different temperature values and Fail, otherwise.

   How test will be performed: Write print statement to print the result of different temperature values. To run this print statement, run main module to get the valid inputs and main call the calculation module which will print the temperature values.

6.3 Tests for Nonfunctional Requirements

Unit testing the non-functional requirements is beyond the scope.

6.4 Traceability Between Test Cases and Modules

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

	M2	M3	M4	M5	M6	M7	M8
test-input-area-id1						X	
test-input-temperature-id2						X	
test-input-emittance-id3						X	
test-load-id4			X	X			
test-fluid-temp-id5							X
test-fluid-energy-id6		X					
test-output-temp-id7					X		
test-output-file-id8					X		
test-constant-id13	X						
test-temp-ode-id14							X

Table 7: Tracebility between test cases and module

7 Appendix

This section provides the sample input file for the test.

Variable	Description	Value
A_t	AREA OF LID	0.0201
T_t	TEMPERATURE OF LID	30
T_f	TEMPERATURE OF FLUID	30
e_t	EMISSIVITY OF LID	0.85
A_ref	AREA OF REFLECTOR	0.0804
T_ref	TEMPERATURE OF REFLECTOR	30
e_ref	EMISSIVITY OF REFLECTOR	0.84
T_g2	TEMPERATURE OF GLASS2	30
A_m	AREA OF MASS	0.064
A_g1	AREA OF GLASS1	0.4761
A_g2	AREA OF GLASS2	0.4761

Table 8: Sample Input

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.