System Verification and Validation Plan for Truss Tool

Maryam Valian

April 17, 2023

1 Revision History

Date	Version	Notes
13-02-2023	1.0	First Draft of VnV plan
14-02-2023	1.1	Updating Test cases
16-02-2023	1.2	Updating Tests for NFRs
25-02-2023	1.3	Updating According to first reviewer feedback
26-02-2023	1.4	Updating According to second reviewer feedback

Contents

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
SRS	Software Requirements Specification
R	Functional Requirements
NFR	Nonfunctional Requirements
N/A	Not Applicable
VnV	Verification and Validation

This document provides an overview of the Verification and Validation plan for Truss Tool. The general information is introduced in section 3. The verification plans and the test descriptions are explained in sections 4,5, and 6, respectively.

3 General Information

3.1 Summary

In this document, we explain how Truss Tool is being tested. Truss Tool is software for truss analysis. Users can define a specific truss, by giving the exact location of the joints, members, supports and external forces. Truss Tool calculates the reaction of the supports and internal forces.

3.2 Objectives

Our objective is to build confidence in the software's correctness and increase its reliability. The functional requirements will be tested according to section 5.1, and Non-functional requirements will be tested according to section 5.2. We do not measure usability for this project. In this phase, we did not develop a graphical visualization for the truss. The usability will be considered when a graphical module is added to the project.

3.3 Relevant Documentation

Relevant documentation can be found by providing direct links. The VnV plan is highly related to the VnV report and we will report the results of the designed test in that document. Please see the Problem statement for goal definition and the SRS document for requirement analysis.

- Problem Statement
- SRS
- VnV report

4 Plan

In this section, we explain the VnV plan of Truss Tool. The first section will introduce the VnV team members and their roles in the project's verification. Next, the verification plans of the SRS, design, the VnV verification plan, implementation, automated testing, and software validation plan are given. For static analysis, we use Pylint library of Python as a code inspection.

4.1 Verification and Validation Team

The VnV team members and their roles are as follows:

- Maryam Valian reviews the whole project as the author.
- Dr. Smith reviews the whole project as supervisor.
- Lesley Wheat reviews the whole project as the domain expert.
- Joachim de Fourestier reviews the VnV plan as the secondary reviewer.
- Jason Balaci reviews the MIS as the secondary reviewer.
- Zehong Wong failed to review the SRS as the secondary reviewer.

4.2 SRS Verification Plan

The SRS will be reviewed by Smith and Lesely Wheat. Reviewers can give feedback and revision suggestions to the author by creating issues on GitHub. It is the author's responsibility to check the submitted issues regularly and make necessary modifications.

4.3 Design Verification Plan

The design document MIS will be reviewed by Dr. Smith, Lesley Wheat and Jason Balaci. Reviewers can give feedback and revision suggestions to the author by creating issues on GitHub. It is the author's responsibility to check the submitted issues regularly and make necessary modifications.

4.4 Verification and Validation Plan Verification Plan

The VnV plan document will be reviewed by Dr. Smith, Lesley Wheat and Joachim de Fourestier. Reviewers can give feedback and revision suggestions to the author by creating issues on GitHub. It is the author's responsibility to check the submitted issues regularly and make necessary modifications.

4.5 Implementation Verification Plan

The implementation of the software will be verified using several techniques involving manual or automated interactions. The software will be developed in Python programming language. The code evaluation will involve unit and system testing. The tools for the evaluations will be mentioned in section 4.6. Furthermore, the test cases used in system and unit testing will be stated in sections 5 and 6, respectively.

4.6 Automated Testing and Verification Tools

Truss Tool will be tested and verified by Pytest as an automated tool for unit testing and system testing. Automated testing will be implemented for individual units as well as the integrated system.

4.7 Software Validation Plan

Validation is the process of comparing the outputs of models to experimental values. The validation of the Truss Tool cannot be measured because we don't have real experimental data.

5 System Test Description

5.1 Tests for Functional Requirements

The functional requirements are described in the SRS. Truss Tool will detect invalid inputs and assure that the calculated outputs are correct. Testing R1 and R2 will be explained in the input verification section. R3 and R4 will be explained in the output verification test.

5.1.1 Input Verification

The inputs will be tested to satisfy R1 and R2 from Truss Tool SRS. Specifically, this test will ensure values of the inputs align with the input constraints. Table 1 displays the inputs and outputs of test cases for the input constraints tests.

Input Verification test

1. T-1: Expected Input

   Control: Automatic

   Initial State: N/A

   Input: Valid

   Output: Support Reactions (25,25), Internal forces (-35,-35,25)

   Test Case Derivation: Derived from existing software 2D-Truss Analysis.

   How the test will be performed: It will be performed by test classes built with the help of Pytest

2. T-2: Invalid Number of joints

   Control: Automatic

   Initial State: N/A

   Input: Invalid Joint

   Output: Exception: Invalid Joints

   Test Case Derivation: N input is out of bound for 20 and 2.

   How the test will be performed: It will be performed by test classes built with the help of Pytest.

3. T-3: Invalid amount of external force

   Control: Automatic

   Initial State: N/A

TC	N	J,M,F, S_r, S_p	**Output**
Valid	3	$J = \{(0,0),(1,1),(2,0)\}$ $M = \{(1,2),(2,3),(1,3)\}$ $F = \{(0,0),(0,-50),(0,0)\}$ $S_p = 1, S_r = 3$	(0,25,25) (-35,-35,25)
Invalid Force	3	$J = \{(0,0),(1,1),(2,0)\}$ $M = \{(1,2),(2,3),(1,3)\}$ $F = \{(0,0),(0,100001),(0,0)\}$ $S_p = 1, S_r = 3$	Invalid force Exception
Invalid Joint	2	$J = \{(0,0),(1,1)\}$ $M = \{(1,2)\}$ $F = \{(0,0),(0,-50),(0,0)\}$ $S_p = 1, S_r = 0$	Invalid Joint Exception
Invalid Location	3	$J = \{(0,0),(1,201),(1,0)\}$ $M = \{(1,2),(1,3),(2,3)\}$ $F = \{(0,0),(0,-50),(0,0)\}$ $S_p = 1, S_r = 3$	Invalid Location Exception
Invalid Connection	3	$J = \{(0,0),(1,1),(2,0)\}$ $M = \{(1,1),(3,3),(2,2)\}$ $F = \{(0,0),(0,-50),(0,0)\}$ $S_p = 1, S_r = 3$	Invalid Connection Exception
Invalid Members	4	$J = \{(0,0),(1,1),(2,0),(3,1)\}$ $M = \{(1,2),(1,3),(2,3)\}$ $F = \{(0,0),(0,-50),(0,0)\}$ $S_p = 1, S_r = 3$	Invalid Members Exception
Invalid Index	3	$J = \{(0,0),(1,1),(2,0)\}$ $M = \{(1,2),(1,3),(2,3)\}$ $F = \{(0,0),(0,-50),(0,0)\}$ $S_p = 4, S_r = 5$	Invalid Index Exception
Invalid Supports	3	$J = \{(0,0),(1,201),(1,0)\}$ $M = \{(1,2),(1,3),(2,3)\}$ $F = \{(0,0),(0,-50),(0,0)\}$ $S_p = 0, S_r = 0$	No Support Exception

Table 1: Test cases For input Verification.

Input: Invalid Force

Output: Exception: Invalid Force

Test Case Derivation: Amount of the external force/forces out of bound.

How the test will be performed: It will be performed by test classes built with the help of Pytest.

4. T-4: Invalid index of joints

   Control: Automatic

   Initial State: N/A

   Input: Invalid Index

   Output: Exception: Invalid Index

   Test Case Derivation: Any reference to joints must be in the range of 1 to N. The support Index can be zero in case of no existing support, but both indices of support cannot be zero at the same time.

   How the test will be performed: It will be performed by test classes built with the help of Pytest.

5. T-5: No support defined

   Control: Automatic

   Initial State: N/A

   Input: Invalid Supports

   Output: Exception: Invalid Supports

   Test Case Derivation: There must be at least one support, otherwise the truss moves.

   How the test will be performed: It will be performed by test classes built with the help of Pytest.

6. T-6: Invalid Connection

   Control: Automatic

   Initial State: N/A

Input: Invalid Connection

Output: Exception: Invalid Connection

Test Case Derivation: A valid member connects to different joints.

How the test will be performed: It will be performed by test classes built with the help of Pytest.

7. T-7: Invalid Members

   Control: Automatic

   Initial State: N/A

   Input: Invalid Members

   Output: Exception: Invalid Members

   Test Case Derivation: Member numbers cannot be less than the Joints number.

   How the test will be performed: It will be performed by test classes built with the help of Pytest.

5.1.2 Output verification

The output will be tested as follows:

Output Verification test

1. T-8: Zero summation for the last joint

   Control: Automatic

   Initial State: All internal forces are already calculated.

   Input: Internal forces array I

   Output: Exception: Output Error

   How the test will be performed: It will be performed by Truss Tool. The summation of internal forces in the last joint will be less than a small number e. Otherwise, it will generate an exception message.

Test cases	N	**input**	Results
Valid N3m3	3	$J = \{(0,0),((1,1),(2,0)\}$ $M = \{(1,2),((1,3),(2,3)\}$ $F = \{(0,0),(0,-50),(0,0)\}$ $P = 2, R = 3$	$Px = 0, Py = 25$ $Ry = 25$ $I = -35, -35, 25$
Valid N4m5	4	$J = \{(0,0),((1,1),(1,0),(2,0)\}$ $M = \{(1,2),((1,3),(2,3)(2,4),(3,4)\}$ $F = \{(0,0),(0,0)(0,10)(0,0)\}$ $P = 1, R = 2$	$Px = 0, Py = 0$ $Ry = -10$ $I = 0, 0, -10, 0, 0$
Valid N5m7	5	$J = \{(0,0),((1,1),(2,0),(3,1),(4,0)\}$ $M = \{(1,2),((1,3),(2,3)$ $,(2,4),(3,4),(3,5)\}$ $F = \{(0,0),(0,-50),$ $(0,0),(0,0)(0,0)\}$ $P = 1, R = 5$	$Px = 0, Py = 37.5$ $Ry = 12.5$ $I = -53.0, 37.5,$ $-17.6, -25, 17.6,$ $12.5, -17.6$
Valid Crane	10	$J = \{(0,0),((2,0),(0,2),(2,2),$ $(0,4),(2,4)(0,6),(2,6),(2,8),(6,6)\}$ $M = \{(1,2),(1,3),(2,3),(2,4),$ $(3,4),(3,5),(4,5),(4,6),(5,6),$ $(5,7),(6,7),(6,8),(7,8),(8,10),$ $(7,9),(8,9),(9,10)\}$ $F = \{(0,0),(0,0),(0,0),(0,0),(0,0),$ $(0,0),(0,0)(0,0),(0,0),(0,-100)\}$ $P = 2, R = 1$	$Px = 0, Py = 300$ $Ry = -200$ $I = 0, 200, 0, -300, 0, 200,$ $0, -300, 0, 200, 0,$ $-300, -200, -200, 282.8,$ $-300, 223.6$

Table 2: Test cases for output verification Pseudo-oracle test

2. T-9: Pseudo-oracle test
   Control: Automatic

   Initial State: All internal forces are already calculated.

   Input: Valid n4m5

Output: The mean error between results.

How the test will be performed: It will be performed by the help of Pytest.

5.2 Tests for Nonfunctional Requirements

This section will define the tests to ensure Truss Tool satisfies the nonfunctional requirements seen in the SRS document of Truss Tool.

5.2.1 Reliability

Performance measurement

1. T-9: Reliability

   Type: Automatic

   Initial State: Results of Truss tool on Test cases are ready. Input/Condition: The user has entered the inputs correctly.

   Output/Result: The mean error between the results of truss tool and pseudo-oracle.

   How the test will be performed: The results from table 2 will be compared to the results of Truss Tool and automatically checked that the difference is less than a small number.

5.3 Traceability Between Test Cases and Requirements

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that is marked with an X may have to be modified as well. Table 4 shows which test cases are supporting which requirements.

6 Unit Test Description

In this section, we explain our overall philosophy of test case selection. The main idea is to divide test cases to two categories: the first one to check

Test Cases	R1	R2	R3	R4	NFR1
Valid	X	X	X	X	X
Invalid force	X	X			
Invalid Joints	X	X			
Invalid Members	X	X			
Invalid supports	X	X			
Invalid index	X	X			
Invalid Connection	X	X			
Invalid Location	X	X			
Valid n4m5 X	X	X	X	X	X
Valid n5n7 X	X	X	X	X	X
Valid Crane X	X	X	X	X	X

Table 3: Traceability Matrix between tests and requirements.

input verification module and the second to verify the correctness of the output validation module. for more detailed information about design and modules please check MIS Document.

6.1 Unit Testing Scope

In this project, we use Scipy library of Python to solve our linear equations. Testing this library is not in the scope of this document and we trust its results. For unit testing we use Pytest to automatically check modules.

6.2 Tests for Functional Requirements

6.2.1 Input parameters Module

1. Input parameter

   Type: Automatic

   Initial State: Truss Tool is running.

   Input: import input.txt

Output: assert = True

Test Case Derivation: The parameters inputted by the user should match the state variable from Inputs Module. An assert statement will return True if these are equal.

How test will be performed: with help of Pytest Library.

6.2.2 Calculation Module

1. Caclulate react module

   Type: Automatic

   Initial State: Truss Tool is running.

   Input: import input.txt

   Output: assert = True

   Test Case Derivation: Test Case Derivation: Outputs of the system should match the values calculated by an online software Valdivia. An assert statement will return True if the mean error between calculated outputs and expected outputs is less than 0.1

   How test will be performed: with help of Pytest Library.

2. Caclulate Internals module

   Type: Automatic

   Initial State: Truss Tool is running.

   Input: import input.txt

   Output: assert = True

   Test Case Derivation: Test Case Derivation: Outputs of the system should match the values calculated by an online software Valdivia. An assert statement will return True if the mean error between calculated outputs and expected outputs is less than 0.1

   How test will be performed: with help of Pytest Library.

6.2.3 Output Module

1. Outputs

   Type: Automatic

   Initial State: Truss Tool is running.

   Input: import input.txt

   Output: Generates output.txt

   Test Case Derivation: Outputs should include all the calculated support reactions and internal forces.

   How test will be performed: Manually checking the file output.txt in the Src folder.

6.3 Tests for Nonfunctional Requirements

Testing nonfunctional requirements of units is not required for Truss. Tests for nonfunctional requirements of system are introduced in section 5.2.

6.4 Traceability Between Test Cases and Modules

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Table shows the dependencies between the test cases and the all of designed modules except the modules that are implemented by the operating system or Python.

Test Cases	Modules
6.2.1	Inputs Module
6.2.2	React Module
6.2.2	Internal Module
6.2.3	Outputs Module

References

Anton Valdivia. 2d-truss analysis. URL https://valdivia.staff.jade-hs.de/fachwerk_en.html.

7 Appendix

For easy understanding, we provide a symbolic parameters table briefly here. For a more detailed table please check SRS Document.

7.1 Symbolic Parameters

Symbol	Meaning
J	Tuples of the joint's coordination. Each tuple is a pair of two-dimensional geometric coordination.
M	Tuples of members. Each tuple is pair of joint index
F	External forces. Tuples of the vertical and horizontal amount of force for each joint
$S = (S_p, S_R)$	Pair of two indices. Index of pinned support and index of roller support

Table 4: Symbol Table for VnV document.

7.2 Usability Survey Questions?

Not Applicable.