Software Stability Analysis: System Verification and Validation Plan

Brooks MacLachlan

October 20, 2018

1 Revision History

Date	Version	Notes
10/10/18	1.0	Initial template fill-ins

2 Symbols, Abbreviations and Acronyms

The symbols, abbreviations, and acronyms used in this document include those defined in the table below, as well as any defined in the tables found in Section 2 of the Software Requirements Specification (SRS) document.

symbol	**description**
MIS	Module Interface Specification
MG	Module Guide
TC	Test Case
VnV	Verification and Validation

Contents

List of Tables

List of Figures

This document outlines the system verification and validation plan for the software. General information regarding the system under test and the objectives of the verification and validation activities is provided in Section 3. Overviews of verification plans for the SRS, design, and implementation are given in Section 4, along with a summary of the validation plan for the software. Section 5 details specific system test cases for verifying the requirements outlined in Section 6 of the SRS. A summary of planned static verification activities can be found in Section 6 of this document.

3 General Information

3.1 Summary

The software being tested is the Slope Stability Analysis Program (SSP). Based on user-defined slope geometry and material properties, SSP determines the critical slip surface of the given slope, the corresponding factor of safety, and interslice normal and shear forces along the critical slip surface.

3.2 Objectives

The purpose of the verification and validation activities is to confirm that SSP exhibits desired software qualities. The primary objective is to build confidence in the correctness of the software. The tests described in this document cannot definitively prove correctness, but they can build confidence by verifying that the software is correct for the cases covered by tests. Other important qualities to be verified are the understandability, maintainability, and reusability of the software.

3.3 References

Extensive information about the purpose and requirements of SSP can be found in the SRS document. This System VnV Plan is complemented by the System VnV Report, where the results of the tests planned in this document are discussed. For more details on test cases specific to the implementation of SSP, consult the Unit VnV Plan document. The latest documentation for SSP can be found on GitHub, at
https://github.com/smiths/caseStudies/tree/master/CaseStudies/ssp.

4 Plan

4.1 Verification and Validation Team

Brooks MacLachlan is responsible for the verification and validation of SSP, though input from various students and the professor, Dr. Spencer Smith, of CAS 741 will also contribute.

4.2 SRS Verification Plan

SRS verification will be carried out by reviews. Brooks MacLachlan will extensively review the SRS, including reviewing the sources to confirm that the theories and models described in the SRS are correct. Any issues identified during this review will be fixed immediately by Brooks MacLachlan. This review will be followed up by additional reviews by Dr. Spencer Smith and Vajiheh Motamer, a student in CAS 741. Any issues identified by these reviewers will be recorded through the issue tracker on GitHub, and addressed by Brooks MacLachlan. Brooks MacLachlan will also informally present the SRS to the entire class of CAS 741. Any issues discovered during this presentation will be subsequently fixed by Brooks MacLachlan. A focus of these reviews will be to verify the non-functional requirements of correctness and understandability by identifying information in the SRS that is incorrect or ambiguous.

4.3 Design Verification Plan

The design of SSP is outlined in the Module Guide (MG) and Module Interface Specification (MIS) documents. The design will be verified by review of these documents. Brooks MacLachlan will extensively review both documents. To verify correctness, part of this review will be to ensure that every module traces to a requirement and that every requirement is traced to by a module. To evaluate the modularity of the design, ”uses” relationships between modules will be examined to ensure modules do not mutually use each other. The review will also check that each module hides exactly one secret. Ensuring the program is well-modularized contributes to verification of the non-functional requirements of maintainability and reusability. Any issues identified during these reviews will be fixed immediately by Brooks MacLachlan. Dr. Spencer Smith will also review both documents. An additional review will be conducted by students in CAS 741: Karol Serkis for the MG and Malavika Srinivasan for the MIS. Any issues identified by these reviews will be recorded through the issue tracker on GitHub, and addressed by Brooks MacLachlan. Brooks MacLachlan will also informally present the MG to the entire class of CAS 741. Any design issues discovered during this presentation will be subsequently fixed by Brooks MacLachlan. Reviews of these design documents will also focus on ensuring understandability by identifying descriptions and specifications that are ambiguous.

4.4 Implementation Verification Plan

The implementation of SSP will be verified by review and by testing. Brooks MacLachlan will extensively review the implementation. Any issues identified during these reviews will be fixed immediately by Brooks MacLachlan. This review will be followed up by reviews by Dr. Spencer Smith and Robert White, a student in CAS 741. Any issues identified by these reviews will be recorded through the issue tracker on GitHub, and addressed by Brooks MacLachlan. Brooks MacLachlan will also informally present the implementation to the entire class of CAS 741. Any implementation issues discovered during this presentation will be subsequently fixed by Brooks MacLachlan. These reviews will contribute to verifying correctness and understandability of the software by identifying code that is not traceable to any specifications described in the SRS, MG, or MIS.

The implementation will also be verified by testing. Specific test cases are outlined in Section 5 of this document. Test cases that are directly dependent on implementation details are outlined in an accompanying document, the Unit VnV Plan. All tests will be written (where applicable), reviewed, executed, and reported on by Brooks MacLachlan.

4.5 Software Validation Plan

There is no validation plan for SSP.

5 System Test Description

The values in Table 1 will be used as input for many of the test cases described throughout this section. These values were taken from the User's Guide for this project by Karchewski (8 January 2012). Individual test cases will reference the table as input but specify new values for any input parameter that should have a different value than specified by the table.

5.1 Tests for Functional Requirements

5.1.1 User Input Tests

Valid User Input

The following set of test cases is intended to cover different forms of valid user input. Valid user input includes: slopes with or without a water table, slopes with one layer or multiple layers, slopes that increase or decrease as x increases, slopes described by the minimum number of points, 2, or more than the minimum number of points, and slopes solved with f as a constant or as a half-sine.

TC1: test-valid_input_decreasing

Control: Automatic

Initial State: New session

Input: As described in Table 1.

Output: SSP runs to completion with no errors

How test will be performed: Automated test on unit testing framework

TC2: test-valid_input_increasing

Control: Automatic

Initial State: New session

Input: As described in Table 1, except with slope coordinates $\{(x_{\text{us}}, y_{\text{us}})\}$ increasing as x increases, as follows: {(0, 15), (30, 15), (40, 20), (50, 25), (70, 25)}.

Output: SSP runs to completion with no errors

How test will be performed: Automated test on unit testing framework

Input	**Unit**	**Value**
$\{(x_{\text{us}}, y_{\text{us}})\}$	m	{(0, 25), (20, 25), (30, 20), (40, 15), (70, 15)}
$\{(x_{\text{wt}}, y_{\text{wt}})\}$	m	{(0, 22), (10.87, 21.28), (21.14, 19.68), (31.21, 17.17), (38.69, 14.56), (40, 14), (70, 14)}
$x_{\text{slip}}^{\text{minStart}}$	m	10
$x_{\text{slip}}^{\text{maxStart}}$	m	24
$x_{\text{slip}}^{\text{minEnd}}$	m	34
$x_{\text{slip}}^{\text{maxEnd}}$	m	53
$y_{\text{slip}}^{\text{min}}$	m	5
$y_{\text{slip}}^{\text{max}}$	m	26
c'	Pa	5000
φ'	$^\circ$	20
γ	$\text{N}\,\text{m}^{-3}$	15000
γ_{Sat}	$\text{N}\,\text{m}^{-3}$	15000
γ_{w}	$\text{N}\,\text{m}^{-3}$	9800
Boolean representing the form of the interslice variation function f. 1 means f is constant, 0 means f is a half-sine	N/A	0

Table 1: Input to be used for test cases

TC3: test-valid_input_multiple

Control: Automatic

Initial State: New session

Input: As described in Table 1, except with an additional slope layer described by the following coordinates: {(0, 20), (36, 17), (40, 15), (70, 15)}.

Output: SSP runs to completion with no errors

How test will be performed: Automated test on unit testing framework

TC4: test-valid_input_noWT

Control: Automatic

Initial State: New session

Input: As described in Table 1, except with no water table vertices $\{(x_{\text{wt}}, y_{\text{wt}})\}$.

Output: SSP runs to completion with no errors

How test will be performed: Automated test on unit testing framework

TC5: test-valid_input_minimal

Control: Automatic

Initial State: New session

Input: As described in Table 1, except with except with only 2 slope coordinates $\{(x_{\text{us}}, y_{\text{us}})\}$, as follows: {(0, 25), (20, 25). Also, with no water table vertices $\{(x_{\text{wt}}, y_{\text{wt}})\}$.

Output: SSP runs to completion with no errors

How test will be performed: Automated test on unit testing framework

TC6: test-valid_input_fConstant

Control: Automatic

Initial State: New session

Input: As described in Table 1, except with f as a constant. This is accomplished by setting the Boolean-type input to 1.

Output: SSP runs to completion with no errors

How test will be performed: Automated test on unit testing framework

TC7: test-valid_input_xMinStartEqual

Control: Automatic

Initial State: New session

Input: As described in Table 1, except with $x_{\text{slip}}^{\text{minStart}}$ set to 0.

Output: SSP runs to completion with no errors

How test will be performed: Automated test on unit testing framework

TC8: test-valid_input_xMaxEndEqual

Control: Automatic

Initial State: New session

Input: As described in Table 1, except with $x_{\text{slip}}^{\text{maxEnd}}$ set to 70.

Output: SSP runs to completion with no errors

How test will be performed: Automated test on unit testing framework

Invalid User Input

The test cases described in Table 2 are intended to cover all invalid input possibilities. Invalid input is input that defies the data constraints described in Section 5.2.6 of the SRS. These test cases are identical to each other with the exception of their input. The input for each is specified in Table 2. For each test case, the inputs not specified in this table are specified in Table 1. The control method for these test cases is automatic. The initial state for each is a new session. The expected output is the generation of an exception. The tests will be performed as automated tests on a unit testing framework.

Test Case	**Test Name**	**Input**	**Value**
TC9	test-invalid_slope_decToInc	$\{(x_{\text{us}}, y_{\text{us}})\}$	{(0, 25), (20, 25), (30, 20), (40, 25), (70, 25)}
TC10	test-invalid_slope_incToDec	$\{(x_{\text{us}}, y_{\text{us}})\}$	{(0, 15), (20, 15), (30, 20), (40, 15), (70, 15)}
TC11	test-invalid_slope_diffStart	$\{(x_{\text{us}}, y_{\text{us}})\}$	{(0, 15), (20, 15), (30, 20), (40, 25), (70, 25)}, {(10, 15), (20, 15), (30, 20), (40, 25), (70, 25)}
TC12	test-invalid_slope_diffEnd	$\{(x_{\text{us}}, y_{\text{us}})\}$	{(0, 15), (20, 15), (30, 20), (40, 25), (70, 25)}, {(10, 15), (20, 15), (30, 20), (40, 25), (60, 25)}

TC13	test-invalid_slope_onePt	$\{(x_{\text{us}}, y_{\text{us}})\}$	{(0, 15)}
TC14	test-invalid_slope_diffStartWT	$\{(x_{\text{wt}}, y_{\text{wt}})\}$	{(10, 22), (10.87, 21.28), (21.14, 19.68), (31.21, 17.17), (38.69, 14.56), (40, 14), (70, 14)}
TC15	test-invalid_slope_diffEndWT	$\{(x_{\text{wt}}, y_{\text{wt}})\}$	{(0, 22), (10.87, 21.28), (21.14, 19.68), (31.21, 17.17), (38.69, 14.56), (40, 14), (60, 14)}
TC16	test-invalid_slope_onePtWT	$\{(x_{\text{wt}}, y_{\text{wt}})\}$	{(0, 22)}
TC17	test-invalid_slip_xMinStart	$x_{\text{slip}}^{\text{minStart}}$	-5
TC18	test-invalid_slip_xMaxStart	$x_{\text{slip}}^{\text{maxStart}}$	5
TC19	test-invalid_slip_xMinEnd	$x_{\text{slip}}^{\text{minStart}}$	20
TC20	test-invalid_slip_xMaxEnd	$x_{\text{slip}}^{\text{minStart}}$	30
TC21	test-invalid_slip_xMaxEndOut	$x_{\text{slip}}^{\text{minStart}}$	75
TC22	test-invalid_slip_yMin	$y_{\text{slip}}^{\text{min}}$	30
TC23	test-invalid_slip_yMax	$y_{\text{slip}}^{\text{min}}$	0
TC24	test-invalid_slip_yEqual	$y_{\text{slip}}^{\text{min}}$	5
TC25	test-invalid_cohesion_0	c'	0
TC26	test-invalid_cohesion_negative	c'	-5
TC27	test-invalid_angFric_0	φ'	0
TC28	test-invalid_angFric_negative	φ'	-5
TC29	test-invalid_angFric_90	φ'	90
TC30	test-invalid_angFric_obtuse	φ'	100
TC31	test-invalid_unitWt_0	γ	0
TC32	test-invalid_unitWt_negative	γ	-5
TC33	test-invalid_unitWtSat_0	γ_{Sat}	0

TC34	test-invalid_unitWtSat_negative	γ_{Sat}	-5	
TC35	test-invalid_unitWtWater_0	γ_{w}	0	
TC36	test-invalid_unitWtWater_negative	γ_{w}	-5	

Table 2: Faulty Input Test Cases

5.1.2 Calculation Tests

5.1.3 Output Tests

5.2 Tests for Nonfunctional Requirements

5.2.1 Correctness tests

Title for Test

1. test-id1

   Type:

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.2.2 Area of Testing2

...

5.3 Traceability Between Test Cases and Requirements

The purpose of the traceability matrix shown in Table 3 is to provide easy references on which requirements are verified by which test cases, and which test cases need to be updated if a requirement changes. If a requirement is changed, the items in the column of that requirement that are marked with an "X" may have to be modified as well.

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
TC2										

Table 3: Traceability Matrix Showing the Connections Between Requirements and Test Cases

6 Static Verification Techniques

The reviews described in Section 4.4 will employ the static verification techniques of code walkthroughs and code inspection to attempt to uncover issues with the implementation.

7 References

Brandon Karchewski. Slope stability program (ssp) user's guide 1.0, 8 January 2012.

8 Appendix

This is where you can place additional information.

8.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

8.2 Usability Survey Questions?

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