[Since we always call it SSP, I think SSP should appear in the title —SS]
[Added SSP to the title —BM]

Software Stability Analysis: System Verification and Validation Plan for SSP

Brooks MacLachlan

December 9, 2018

1 Revision History

Date	Version	Notes
10/10/18	1.0	Initial template fill-ins
10/17/18	1.1	Added initial test cases
10/20/18	1.2	Added faulty input test cases
10/22/18	1.3	Added remaining test cases
11/01/18	1.4	Updates based on feedback
12/06/18	1.5	Updates based on Dr. Smith's feedback
12/09/18	1.6	Updates prompted by test implementation and for final submission

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**
j	index representing a single coordinate
MIS	Module Interface Specification
MG	Module Guide
TC	Test Case
VnV	Verification and Validation

Contents

List of Tables

[The title of the tables use different conventions for capitalization. The specific convention doesn't matter, but it should be applied consistently. (The same convention should be used for the figures and the table captions.) —SS]

[Made this consistent —BM]

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.

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.

[LaTeX has a rule that it inserts two spaces at the end of a sentence. It detects a sentence as a period followed by a capital letter. Since the period after Dr. isn't actually the end of a sentence, you need to tell LaTeX to insert one space. You do this either by Dr. Spencer Smith (if you don't mind a line-break between Dr. and Spencer), or Dr. Spencer Smith (to force LaTeX to not insert a line break). —SS]

[Fixed throughout the document —BM]

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.

[Use "uses" to get the correct opening and closing quotes. —SS]

[Thanks, done here and throughout documentation —BM]

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 a review by Dr. Spencer Smith. Any issues identified by this review will be recorded through the issue tracker on GitHub, and addressed 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 because there is no experimental data or other source of external validation.

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

These test cases are intended to cover different forms of valid user input. Valid user input includes: slopes with or without a water table, slopes that increase or decrease as x increases, and slopes described by the minimum number of points, 2, or more than the minimum number of points. These test cases can be found in the Unit VnV Plan, TC1-TC25.

[These tests look good, and they are system tests because they come from the SRS, but they are also unit tests because there will be one module with the responsibility to validate the input. We don't really need to run the tests to completion; we just need to unit test the input validation routines. I'm not suggesting that you change anything, but I am wondering if in the future I should encourage students to put these tests in the VnV plan for their units, instead of in the System VnV Plan. Some of your tests might fail because the input isn't adequate for the physics, even though the inputs satisfy all of the constraints. For instance, the minimal test (TC4) might fail for numerical reasons. —SS]

[I've removed some of these tests because I ended up writing more exhaustive tests that covered the same things in the Unit VnV plan. I also moved the test for setting f to a constant value to the calculation subsection because I think the Unit VnV plan test case for valid *const_f* input is sufficient in terms of testing the input module, but it is more interesting to

Input	Unit	Value
$\{(x_{\text{slope}}, y_{\text{slope}})\}$	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)}
$x_{\text{slip}}^{\text{minEtr}}$	m	10
$x_{\text{slip}}^{\text{maxEtr}}$	m	24
$x_{\text{slip}}^{\text{minExt}}$	m	34
$x_{\text{slip}}^{\text{maxExt}}$	m	53
$y_{\text{slip}}^{\text{min}}$	m	5
$y_{\text{slip}}^{\text{max}}$	m	26
c'	Pa	5000
φ'	°	20
γ	N m^{-3}	15000
γ_{Sat}	N m^{-3}	15000
γ_{w}	N m^{-3}	9800
const_f	N/A	0

Table 1: Input to be used for test cases

see what effect f has on the actual calculation (i.e. to verify that setting f actually does something). Of the tests that were left over, they made much more sense to me if they were in the Unit VnV plan. You are right that the minimal test may fail even though it meets input constraints, so it really doesn't make sense to run these as system tests, even if they do qualify as system tests. Since they will be run as unit tests, I moved them to the unit vnv plan. —BM]

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**
TC1	test-invalid_slope_nonMono	$\{(x_{\text{slope}}, y_{\text{slope}})\}$	{(0, 25), (20, 25), (30, 20), (20, 15), (70, 15)}
TC2	test-invalid_slope_onePt	$\{(x_{\text{slope}}, y_{\text{slope}})\}$	{(0, 15)}
TC3	test-invalid_slope_highStartWT	$\{(x_{\text{wt}}, y_{\text{wt}})\}$	{(10, 22), (10.87, 21.28), (21.14, 19.68), (31.21, 17.17), (38.69, 14.56), (40, 14), (70, 14)}
TC4	test-invalid_slope_highEndWT	$\{(x_{\text{wt}}, y_{\text{wt}})\}$	{(0, 22), (10.87, 21.28), (21.14, 19.68), (31.21, 17.17), (38.69, 14.56), (40, 14), (80, 14)}
TC5	test-invalid_slope_lowStartWT	$\{(x_{\text{wt}}, y_{\text{wt}})\}$	{(-10, 22), (10.87, 21.28), (21.14, 19.68), (31.21, 17.17), (38.69, 14.56), (40, 14), (70, 14)}
TC6	test-invalid_slope_lowEndWT	$\{(x_{\text{wt}}, y_{\text{wt}})\}$	{(0, 22), (10.87, 21.28), (21.14, 19.68), (31.21, 17.17), (38.69, 14.56), (40, 14), (60, 14)}
TC7	test-invalid_slope_nonMonoWT	$\{(x_{\text{wt}}, y_{\text{wt}})\}$	{(0, 22), (10.87, 21.28), (21.14, 19.68), (31.21, 17.17), (28.69, 14.56), (40, 14), (60, 14)}
TC8	test-invalid_slope_onePtWT	$\{(x_{\text{wt}}, y_{\text{wt}})\}$	{(0, 22)}
TC9	test-invalid_slip_xMinEtr	$x_{\text{slip}}^{\text{minEtr}}$	-5
TC10	test-invalid_slip_xMaxEtr	$x_{\text{slip}}^{\text{maxEtr}}$	-5

TC11	test-invalid_slip_xMinExt	$x_{\text{slip}}^{\text{minExt}}$	-5
TC12	test-invalid_slip_xMaxExt	$x_{\text{slip}}^{\text{maxExt}}$	-5
TC13	test-invalid_slip_xMinEtrOut	$x_{\text{slip}}^{\text{minEtr}}$	75
TC14	test-invalid_slip_xMaxEtrOut	$x_{\text{slip}}^{\text{maxEtr}}$	75
TC15	test-invalid_slip_xMinExtOut	$x_{\text{slip}}^{\text{minExt}}$	75
TC16	test-invalid_slip_xMaxExtOut	$x_{\text{slip}}^{\text{maxExt}}$	75
TC17	test-invalid_slip_yMin	$y_{\text{slip}}^{\text{min}}$	30
TC18	test-invalid_slip_yMax	$y_{\text{slip}}^{\text{max}}$	0
TC19	test-invalid_cohesion_0	c'	0
TC20	test-invalid_cohesion_negative	c'	-5
TC21	test-invalid_angFric_0	φ'	0
TC22	test-invalid_angFric_negative	φ'	-5
TC23	test-invalid_angFric_90	φ'	90
TC24	test-invalid_angFric_obtuse	φ'	100
TC25	test-invalid_unitWt_0	γ	0
TC26	test-invalid_unitWt_negative	γ	-5
TC27	test-invalid_unitWtSat_0	γ_{Sat}	0
TC28	test-invalid_unitWtSat_negative	γ_{Sat}	-5
TC29	test-invalid_unitWtWater_0	γ_{w}	0
TC30	test-invalid_unitWtWater_negative	γ_{w}	-5

Table 2: Faulty input test cases

[Very thorough. Nice summary of the tests. Test cases are specific. —SS]

5.1.2 Calculation Tests

The tests in this section verify that SSP correctly determines critical slip surfaces and calculates factors of safety by comparing results obtained from SSP to results reported in literature or results obtained by the original version of SSP for some specific example problems. In each test case, the sources to compare to are specified.

Input	**Unit**	**Value**
$\{(x_{\text{slope}}, y_{\text{slope}})\}$	m	{(0, 5), (5, 5), (15, 10), (25, 10)}
$\{(x_{\text{wt}}, y_{\text{wt}})\}$	m	N/A
$x_{\text{slip}}^{\text{minEtr}}$	m	0
$x_{\text{slip}}^{\text{maxEtr}}$	m	10
$x_{\text{slip}}^{\text{minExt}}$	m	10
$x_{\text{slip}}^{\text{maxExt}}$	m	20
$y_{\text{slip}}^{\text{min}}$	m	0
$y_{\text{slip}}^{\text{max}}$	m	12
c'	Pa	9800
φ'	$^\circ$	10
γ	$\text{N}\,\text{m}^{-3}$	17640
γ_{Sat}	$\text{N}\,\text{m}^{-3}$	17640
γ_{w}	$\text{N}\,\text{m}^{-3}$	N/A
const_f	N/A	0

Table 3: Input to be used for literature comparison test cases

TC31: test-ex1_FS

Control: Automatic

Initial State: New session

Input: As described in Table 3.

Output: Relative error between factor of safety calculated by SSP and factor of safety from literature, as described below. The relative error will be averaged across 5 runs since there is some expected variability.

Source	**Factor of Safety**
Greco (1 July 1996)	1.3270
Malkawi et al. (1 August 2001)	1.2380
Cheng et al. (27 Decemeber 2006)	1.3250
Li et al. (25 June 2010)	1.3270

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

TC32: test-ex1_slip

Control: Automatic

Initial State: New session

Input: As described in Table 3.

Output: Relative error between critical slip surface determined by SSP and critical slip surfaces from literature. The relative error will be averaged across 5 runs since there is some expected variability. Relative error is calculated by dividing the critical slip surfaces into 10 evenly spaced slices, calculating the distance between respective vertices from SSP and from literature, computing the Euclidean norm of the resulting vector, and dividing it by the Euclidean norm for the vector describing the original slip surface from literature. For brevity, the table below gives only the starting and ending x values of the critical slip surfaces from literature. More complete descriptions of the slip surface can be found by referring to the source.

[This looks like a good test, but rather than using the sum of the discrepancies, I suggest that you use a relative value of the Euclidean norm. You could calculate the discrepancy vector, as you described, and then calculate the norm of this vector. You would then divide this norm by the norm of the original y values for the slip surface. This same comment applies elsewhere in this document. —SS]

[Updated the way the relative error is calculated based on your suggestion. Also updated it in the other similar tests. —BM]

Source	Starting x value	Ending x value
Greco (1 July 1996)	4.8077	18.2911
Malkawi et al. (1 August 2001)	3.5400	20.1419
Cheng et al. (27 Decemeber 2006)	4.5258	18.3943
Li et al. (25 June 2010)	4.6000	18.5300

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

TC33: test-origProg_FS

Control: Automatic

Initial State: New session

Input: As described in Table 1.

Output: Relative error between the factor of safety calculated by SSP and the factor of safety calculated by the original version of SSP, 0.9835, which is assumed to be correct and can be found at https://github.com/smiths/caseStudies/tree/master/CaseStudies/ssp/misc. The relative error will be averaged across 5 runs since there is some expected variability.

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

TC34: test-origProg_slip

Control: Automatic

Initial State: New session

Input: As described in Table 1.

Output: Relative error between critical slip surface determined by SSP and critical slip surfaces determined by the original version of SSP, which is assumed to be correct and can be found at https://github.com/smiths/caseStudies/tree/master/CaseStudies/ssp/misc. The relative error will be averaged across 5 runs since there is some expected variability. Relative error is calculated by dividing the critical slip surfaces into 10 evenly spaced slices, calculating the distance between respective vertices from SSP and from the original version,

computing the Euclidean norm of the resulting vector, and dividing it by the Euclidean norm for the vector describing the slip surface from the original version of SSP. The critical slip surface calculated by the original version of SSP is shown in Figure 1

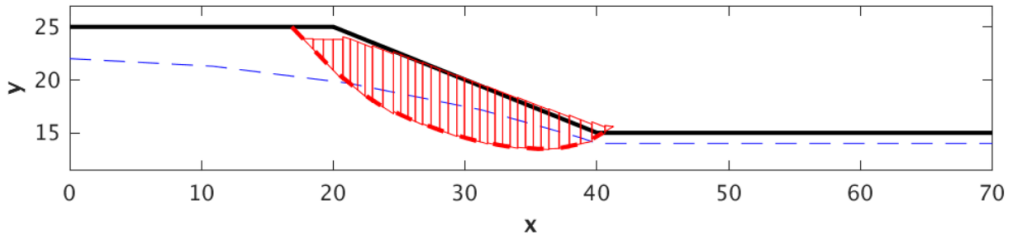

Figure 1: The critical slip surface for TC34 calculated by the original version of SSP

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

TC35: test-origProg_normal

Control: Automatic

Initial State: New session

Input: As described in Table 1.

Output: Relative error between the interslice normal force calculated by SSP and the interslice normal force calculated by the original version of SSP, which is assumed to be correct and can be found at https://github.com/smiths/caseStudies/tree/master/CaseStudies/ssp/misc. The relative error will be averaged across 5 runs since there is some expected variability. Relative error is calculated by dividing the interslice normal force curve into 10 evenly spaced slices, calculating the distance between respective vertices from SSP and from the original version, computing the Euclidean norm of the resulting vector, and dividing it by the Euclidean norm for the vector describing the interslice normal force from the original version of SSP. The interslice normal forces along the critical slip surface calculated by the original SSP are shown in Figure 2.

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

TC36: test-origProg_shear

Control: Automatic

Initial State: New session

Input: As described in Table 1.

Output: Relative error between the interslice shear force calculated by SSP and the interslice shear force calculated by the original version of SSP, which is assumed to be correct and can be found at https://github.com/smiths/caseStudies/tree/master/CaseStudies/ssp/misc. The relative error will be averaged across 5 runs since there is some expected variability. Relative error is calculated by dividing the interslice shear force curve into 10 evenly spaced slices, calculating the distance between respective vertices from SSP and from the original version, computing the Euclidean norm of the resulting vector, and dividing it by the Euclidean norm for the vector describing the interslice shear force from the original version of SSP. The interslice shear forces along the critical slip surface calculated by the original SSP are shown in Figure 2.

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

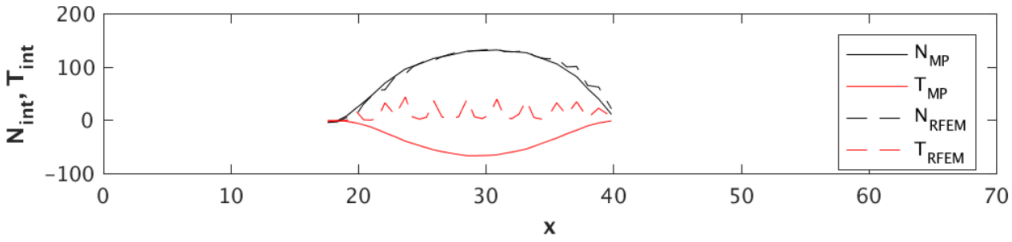

Figure 2: The interslice normal and shear forces for TC35 and TC36 calculated by the original version of SSP, where N_{MP} is the interslice normal force and T_{MP} is the interslice shear force

TC37: test-calc_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 *const_f* input to 1.

Output: SSP runs to completion with no errors and the output is similar to Figures 1 and 2, though not identical. The relative difference in the factor of safety output compared to 0.9835 (the known factor of safety output by the original program) will also be reported.

How test will be performed: Visual inspection of output.

[I wanted to add tests for the correctness NFR in Section 5.2 but felt that it was already covered by these functional requirement test cases. Is that true? —BM]

[Yes, this is true. You can still include a heading for Correctness, but then point out which, already specified, tests cover this NFR. You don't repeat the tests, just list their identifiers. —SS]

[This is done —BM]

5.1.3 Output Tests

Output Delivery Tests

The following set of test cases are intended to verify that all expected outputs are delivered by SSP. The test cases in Table 4 cover the output of certain input values. For each test the table lists the input value that is expected to be output. They are automatic tests with a new session as initial state and will be performed on a unit testing framework, with inputs described in Table 1.

Test Case	Test Name	Input	Expected Output
TC38	test-output_input_XEtrMin	$x_{\text{slip}}^{\text{minEtr}}$	10
TC39	test-output_input_XEtrMax	$x_{\text{slip}}^{\text{maxEtr}}$	24
TC40	test-output_input_XExtMin	$x_{\text{slip}}^{\text{minExt}}$	34
TC41	test-output_input_XExtMax	$x_{\text{slip}}^{\text{maxExt}}$	53
TC42	test-output_input_YLimMin	$y_{\text{slip}}^{\text{min}}$	5
TC43	test-output_input_YLimMax	$y_{\text{slip}}^{\text{max}}$	26
TC44	test-output_input_Const_f	*const_f*	0

Table 4: Test cases for output of input values

TC45: test-output_FS

Control: Automatic

Initial State: New session

Input: As described in Table 1.

Output: SSP reports a value for the factor of safety of the critical slip surface

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

TC46: test-output_slip

Control: Automatic

Initial State: New session

Input: As described in Table 1.

Output: SSP reports the coordinates of the critical slip surface for the slope.

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

TC47: test-output_normal

Control: Automatic

Initial State: New session

Input: As described in Table 1.

Output: SSP reports the interslice normal force along the critical slip surface for the slope.

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

TC48: test-output_shear

Control: Automatic

Initial State: New session

Input: As described in Table 1.

Output: SSP reports the interslice shear force along the critical slip surface for the slope.

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

Output Verification Tests

The following set of test cases are intended to verify that that the output of SSP obeys the physical constraints described in Section 5.2.6 of the SRS.

TC49: test-valid_output_FS

Control: Automatic

Initial State: New session

Input: As described in Table 1.

Output: The factor of safety reported by SSP is greater than 0.

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

5.2 Tests for Nonfunctional Requirements

5.2.1 Maintainability Test

TC50: test-maintainability

Type: Manual

Initial State: Test subject has a completed MG for SSP

Input/Condition: MG for SSP and the following question "If you were tasked with modifying SSP to accept an imposed surface load as an input and incorporate this additional force into the computation, which module(s) would you change?"

Output/Result: Input Module, Morgenstern-Price Calculation Module

How test will be performed: Test subject will be asked the question described in "Input/Condition" and must answer only by consulting the MG. Their answer will be recorded and compared with the correct answer outlined in "Output/Result". If the subject answers correctly, the test passes and confidence in the maintainability of SSP is increased. If the subject answers incorrectly, the test fails and the documentation or implementation of SSP must be updated to be more maintainable.

[I like these tests! —SS]

5.2.2 Reusability Test

TC51: test-reusability

Type: Manual

Initial State: There is a completed implementation of SSP

Input/Condition: Code that prompts the user for command-line inputs related to slope stability analysis.

Output/Result: An alternative version of SSP that uses the input code. All modules in the alternative version should be identical to the existing SSP modules, with the exception of the Input Module.

How test will be performed: The alternative version of SSP will be developed by Brooks MacLachlan. If only the Input Module is changed from the existing version, then the test passes and confidence in the reusability of SSP's modules is increased. If other modules need to be changed, the test fails and the other modules must be modified to be completely reusable.

5.2.3 Correctness Tests

The correctness NFR is covered by the functional requirement test cases for calculations, found in Section 5.1.2.

5.2.4 Understandability Tests

Understandability is a contributing factor to maintainability and reusability. Therefore, the understandability NFR is covered by the test cases for maintainability and reusability, found in Section 5.2.1 and Section 5.2.2, respectively.

5.3 Traceability Between Test Cases and Requirements

The purpose of the traceability matrix shown in Table 5 and Table 6 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	R11
UnitVnVPlan TC1 - TC25	X										
TC1 - TC30		X									
TC31 - TC34			X	X	X						
TC35									X		
TC36										X	
TC38 - TC44							X				
TC45								X			
TC46									X		
TC47										X	
TC48											X
TC49						X					

Table 5: Traceability matrix showing the connections between functional requirements and test cases

	NFR1	NFR2	NFR3	NFR4
TC31 - TC36	X			
TC50		X		X
TC51		X	X	

Table 6: Traceability matrix showing the connections between non-functional requirements and test cases

[I had to recompile the SRS to get the table references to work. Could you add a makefile to the SSP example, like the Blank Project example, so that compiling the example would work via make? —SS]

[Makefiles added for all documents, sorry about that —BM]

[I should modify the template to remove this section, since you are correct that you already discussed these topics. You can remove this section in your version. You don't actually propose code walkthroughs in Section 4.4, so the text here isn't quite correct. —SS]

[The above comment originally referred to the Static Verification Techniques section, which I have now removed —BM]

6 References

Y. M. Cheng, Liang Li, Shi chun Chi, and W. B. Wei. Particle swarm optimization algorithm for the location of the critical non-circular failure surface in two-dimensional slope stability analysis. *Computers and Geotechnics*, (24):92–103, 27 Decemeber 2006.

Venanzio R. Greco. Efficient monte carlo technique for locating critical slip surface. *J. Geotech. Engrg.*, (122):517–525, 1 July 1996.

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

Yu-Chao Li, Yun-Min Chen, Tony L.T Zhan, Dao-Sheng Ling, and Peter John Cleall. An efficient approach for locating the critical slip surface in slope stability analyses using a real-coded genetic algorithm. *Can. Geotech. J.*, (47):806–820, 25 June 2010.

Abdallah I. Husein Malkawi, Waleed F. Hassan, and Sarada K. Sarma. Global search method for locating general slip surface using monte carlo techniques. *J. Geotech. Geoenviron. Eng.*, (127):688–698, 1 August 2001.