Software Stability Analysis: System Verification and Validation Plan

Brooks MacLachlan

October 17, 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

5.1 Tests for Functional Requirements

5.1.1 User Input Tests

Valid User Input

TC1: test-valid_input_decreasing

Control: Automatic

Initial State: New session

Input: A valid input file with 2 slope layers, no water table, and y-values decreasing as x-values increase

Output: No errors

How test will be performed: Automated integration test

TC2: test-valid_input_increasing

Control: Automatic

Initial State: New session

Input: A valid input file with 2 slope layers, no water table, and y-values increasing as x-values increase

Output: No errors

How test will be performed: Automated integration test

TC3: test-valid_input_single

Control: Automatic

Initial State: New session

Input: A valid input file with 1 slope layer and no water table Output: No errors

How test will be performed: Automated integration test

TC4: test-valid_input_watertable

Control: Automatic

Initial State: New session

Input: A valid input file with 2 slope layers, and a water table

Output: No errors

How test will be performed: Automated integration test

Invalid User Input

TC5: test-valid_input_decreasing_increasing

Control: Automatic

Initial State: New session

Input: Input file where a slope layer's y-values start by decreasing with increasing x but then start to increase

Output: Error

How test will be performed: Automated integration test

TC6: test-valid_input_increasing_decreasing

Control: Automatic

Initial State: New session

Input: Input file where a slope layer's y-values start by increasing with increasing x but then start to decrease

Output: Error

How test will be performed: Automated integration test

TC7: test-valid_input_diff_start

Control: Automatic

Initial State: New session

Input: Input file with 2 slope layers with different initial x-values

Output: Error

How test will be performed: Automated integration test

TC8: test-valid_input_diff_end

Control: Automatic

Initial State: New session

Input: Input file with 2 slope layers with different final x-values

Output: Error

How test will be performed: Automated integration test

TC9: test-valid_input_diff_start_wt

Control: Automatic

Initial State: New session

Input: Input with a water table with a different initial x-value than the slope layers

Output: Error

How test will be performed: Automated integration test

TC10: test-valid_input_diff_end_wt

Control: Automatic

Initial State: New session

Input: Input with a water table with a different final x-value than the slope layers

Output: Error

How test will be performed: Automated integration test

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 matrices 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 1: 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.

References

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

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

7.2 Usability Survey Questions?

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