System Verification and Validation Plan for Bridge Corrosion: A Chloride Exposure Prediction Model

Cynthia Liu

March 16, 2024

Revision History

Date	Version	Notes
Feb 9, 2024	1.0	Initial release
Mar 7, 2024	2.0	Modify according to feedback from peer review

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

1.1 Table of Symbols

symbol	unit	description
C_s	kg/m^3/vehicle	chloride on the bridge substructure
$C_{s_{air}}$	kg/m^3/vehicle	chloride sprayed and splashed per unit air volume per vehicle
h_{app}	m	daily water film thickness on the road
h_{total}	m	the total snowfall during a winter season
M_{app}	kg/m^2	deicing salts quantity applied per day
M_{total}	kg/m^2	total amount of deicing salts quantity over winter
SD_{total}	kg/m^3/vehicle	spray density kicked up by each passing truck
$SD_{total\ cl}$	kg/m^3/vehicle	mass of chloride ions per unit air volume
t_1	days	number of days with snowfall
t_2	days	number of days with snow melting

1.2 Abbreviations and Acronyms

symbol	description
AADT	Annual Average Daily Traffic
AADTT	Annual Average Daily Truck Traffic
BC	Bridge Corrosion
CI	Continuous Integration
FR	Functional Requirement
MG	Module Guide
MIS	Module Interface Specification
NFR	Non-functional Requirement
R	Requirement
SRS	Software Requirements Specification
T	Test
VnV	Verification and Validation

2 General Information

This document provides the road-map of the verification and validation plan for Bridge Corrosion (BC), to ensure the requirements and goals of the program mentioned in the SRS. The organization of this document starts with the general information and verification plans, followed by the system test description for functional and non-functional requirements. [delete after mis —Author]Then, it includes the unit test which would not be filled in until after the MIS has been created.

2.1 Summary

BC provides predictive trends for chloride exposure based on user-input coordinates within the province of Ontario. The software utilizes climate and traffic models to build a chloride exposure database, and search for the trends when a user inputs location data.

2.2 Objectives

The objective of this document is to build confidence in the software's correctness and increase its realiability. All functional requirements and non-functional requirements mentioned in SRS are tested. We will try to cover a usability test if we have time and resource. We also assume the climate and traffic models we generate online have been verified by their implementation team so their accuracy is ensured. [TODO: modify later according to real situation —Author]

2.3 Relevant Documentation

The relevant documentation includes Problem Statement which is the proposed idea, Software Requirements Specifications which outlines the requirement of the software. It would also be related to VnV Report which is a conclusion for validation and verification after the software is developed.

3 Plan

The VnV plan begins by introducing the verification and validation team (section 3.1), followed by the verification plans for the SRS (section 3.2) and

design (section 3.3). Subsequently, the verification plans for the VnV Plan (section 3.4) and implementation (section 3.5) are presented. In the end, there are sections on automated testing and verification tools (Section 3.6) and the software validation plan (section 3.7).

3.1 Verification and Validation Team

Name	Document	Role	Description
Dr. Spencer Smith	All	Instructor/ Reviewer	Review all the documents.
Mingsai Xu	-	Domain Expert	Provide theory support for the software.
Cynthia Liu	All	Author	Create all the documents, implement the software, and verify the implementation according to the VnV plan.
Phil Du	All	Domain Expert Reviewer	Review all the documents.
Hunter Ceranic	SRS	Secondary Reviewer	Review the SRS document.
Fasil Cheema	VnV plan	Secondary Reviewer	Review the VnV plan.
Fatemeh Norouziani	MG + MIS	Secondary Reviewer	Review the MG and MIS document.

Table 1: Verification and Validation Team

3.2 SRS Verification Plan

The SRS verification is done by peer review with classmates (Phil Du and Hunter Ceranic), giving suggestions by creating issues in Github. The author (Cynthia Liu) is responsible for reviewing and addressing the issues.
The SRS verification is also reviewed by Dr. Spencer Smith to offer any suggestions and feedbacks.
There is a SRS checklist designed by Dr. Spencer Smith available to use.

3.3 Design Verification Plan

The design verification, including the module guide (MG) and module interface specification (MIS), will be verfied by Phil Du and Fatemeh Norouziani. Dr. Spencer Smith will also review both documents. Reviewers will give feedbacks through Github issues and the author need to address them. The MG checklist and MIS checklist designed by Dr. Spencer Smith could act as a help for reviewers.

3.4 Verification and Validation Plan Verification Plan

The VnV plan is first created and verfied by the author, then turns to the team members to give any suggests by Github issues. The reviewers will access the VnV checklist designed by Dr. Spencer Smith for reference.

3.5 Implementation Verification Plan

The implementation would be verified by testing the functional requirements and non-functional requirements in section 4. The unit tests will also be conducted as detailed in section 5. If possible, a code walkthrough with domain expert(Mingsai Xu) could happen.

3.6 Automated Testing and Verification Tools

The first part of the software, generating the database through climate and traffic model, will be done in MATLAB. The second part of finding the data for the input location, will be done in R. The automated testing will be done in MATLAB and R correspondingly. During the coding, mlint and lintr will be used as a static code analysis. The Github Actions is used for continuous integration, that the CI work flow will run when new code is pushed to the repository. It will also run the CI tests and provide the results of each test in the pull request, so we can see whether the change in the branch introduces an error.

3.7 Software Validation Plan

Software validation plan is beyond the scope for BC as it may require additional time, expertise, and resources that are not available within the scope

of the project. The validation process, ideally should be done by the domain expert, which include ensuring the models and formulas that this project are using are correct.

4 System Test Description

This section includes the system test that will be used for the functional requirements and non-functional requirements.

4.1 Tests for Functional Requirements

Functional requirements for BC are given in SRS section 5.1. There are five functional requirements for BC, R1 is related to the input, R3 is related to the midway calculation, and the other requirements correspond to outputs. Section 4.1.1 describes the input tests related to R1, and section 4.1.2 describes the test related to R3. The output test related to other requirements is discussed in section 4.1.3.

4.1.1 Input Test - test_invalid_input

This section covers R1 of the SRS which includes the input coordinates check for the software. This section also cover the dataset inputs that developers input from models to the software. In most of the cases, the dataset in the models are valid as they have been verified by their developers, but the test cases here still include the scenarios where data may be missing or invalid.

Functional tests - Input tests - Coordinate

1. test-input-coordinate
   Control: Automatic

   Initial State: Uninitialized

   Input: The input column in Table 2.

   Output: The output column in Table 2.

Input (°)		Output	
longitude	*latitude*	valid?	Error Message
278.9699	44.46539	Y	No error message, continue to calcualte output
278.9699	41.197	N	Input position not inside Ontario
-84.34	44.46539	N	Input position not inside Ontario
-81.0301	44.46539	Y	No error message, continue to calcualte output
-74.3	45.2	Y	No error message, continue to calcualte output
278.9699		N	Missing latitude
	44.46539	N	Missing longitude

Table 2: Coordinate input tests

Test Case Derivation: The input coordinate need to be a location inside Ontario. The software produces error message for the invalid inputs, for the valid inputs, it will proceed to next step. [Justify the expected value given in the Output field —SS]

How test will be performed: The author will create test cases and run automatic test by R.

Functional tests - Input tests - Models

Input (°)			Output	
h_{total}	t_1	t_2	valid?	Error Message
183.9371	109	76	Y	No error message, continue to import model data
194.6057	367	255	N	t_1 and t_2 must be less than 365
119.62	53	-30	N	t_1 and t_2 need to be positive
	109	76	N	Data missing for h_{total}
183.9371		76	N	Data missing for t_1
183.9371	109		N	Data missing for t_2

Table 3: Climate model input tests

1. test-input-climate

Control: Automatic

Initial State: Uninitialized

Input: The input column in Table 3.

Output: The output column in Table 3.

Test Case Derivation: This test case is used when inputing the climate models. There are three parameters that would be used from the model and this test is trying to detect if any of them is missing. If so, it will produce error message. It will also produce error message if there is such data but it is invalid.

How test will be performed: The author will pick the sample test case from the model (as in Table 3) and create automatic test by R.

Input (°)		Output	
AADT	*AADTT*	valid?	Error Message
20176	433	Y	No error message, continue to import model data
152	8095	N	*AADTT* should be less than *AADT*
	152	N	Data missing for *AADT*
8095		N	Data missing for *AADTT*

Table 4: Traffic model input tests

2. test-input-traffic

Control: Automatic

Initial State: Uninitialized

Input: The input column in Table 4.

Output: The output column in Table 4.

Test Case Derivation: This test case is applicable when entering traffic models into the software. There are two parameters that would be used from the model and this test is trying to detect if any of them is missing. If so, it will produce error message. Similarly, if the data exists but is invalid, the software will also generate an error message.

How test will be performed: The author will pick the sample test case from the model (as in Table 4) and create automatic test by R.

4.1.2 Intermediate Tests - test_model_calculation

[This section might be more suitable for the unit test as section 4 is for system level? Will come back to this after MG and MIS. —Author]
This section covers R3 of the SRS which includes the check for the middle calculation of the software to generate the database of the software. There are six steps over the calculation process following section 4.4 in SRS as follow. [TODO: how to determine expected output in automatic process? Look at Mingsai's verification method —Author]

- Step 1: Calculate the quantity of deicing salts applied per day with snowfall.
- Step 2: Calculate the thickness of melted water per day with snow melting.
- Step 3: Calculate the water sprayed and splashed by one truck.
- Step 4: Calculate the chloride ions sprayed and splashed by one truck.
- Step 5: Calculate the chloride ions sprayed and splashed by all vehicles in one winter season.
- Step 6: Calculate the deposition of deicing salts on the surface of the bridge substructure.

Test case	Input from Model	Input from Previous Steps	Expected Output
Step 1	$M_{total} = 7.42$, $t_1 = 106$	n/a	$M_{app} = 0.07$
Step 2	$h_{total} = 206.2145$, $t_1 = 106$	n/a	$h_{app} = 1.945$
Step 3	n/a	$h_{app} = 1.945$	$SD_{total} = 97.96$
Step 4	n/a	$SD_{total} = 97.96$, $M_{app} = 0.07$, $h_{app} = 1.945$	$SD_{total\ cl} = 0.00216$
Step 5	$AADT = 1730$, $AADTT = 103$, $t_2 = 76$	$SD_{total\ cl} = 0.00216$	$C_{s_{air}} = 21.7983$
Step 6	n/a	$C_{s_{air}} = 21.7983$	$C_s = 18.668$

Table 5: Test Case for Model Calculation

Functional tests - Intermediate tests - Model Calculations

1. test-calculation

   Control: Automatic

   Initial State: None

   Input: The input columns in Table 5. The input are selected from the models or the result value from previous steps.

   Output: The output columns in Table 5. The test case is considered to pass if the epsilon between the actual output and the expected output is within a specified threshold. The detail of threshold will be determined later in the development.

   Test Case Derivation: The table shows one sample calculation for the chloride exposure. The expected value in the output column is determined using the equation provided by Mingsai et al ([1]). At each step, the test case verifies if the calculation for the corresponding step is accurate. If it is, the test proceeds to the next step.

   How test will be performed: The automatic test will be performed by R. There will be assert function for each step, and there will be more test cases rather than the one in the table. [TODO: summary that into a table —Author]

4.1.3 Output Tests - test_output

This section covers R2, R4of the SRS which includes the check for the final outputs.

Functional tests - Output tests - Chloride exposure

1. test-output-exposure

   Control: Automatic

   Initial State: Uninitialized

   Input: Coordinate

   Output: If the input coordinate is within Ontario, it will return a series (length of at lease ten) of two decimal points data showing the chloride exposure at that location.

Test Case Derivation: This test case cover the R2 and R4 of the SRS. The test will be done by checking the length of the output and the precision after decimal points. The expected value should be within the data constraints in the 4.4.7 section of SRS.

How test will be performed: The automatic test will be performed by R.

2. test-output-verification

   Control: Manual

   Initial State: Uninitialized

   Input: Chloride exposure data

   Output: True or False.

   Test Case Derivation: This test case compares the chloride exposure data that the software generates with the real world data for verification. The test result will be true if the derivation is within a threshold. The detail value of the threshold will be determined during development.

   How test will be performed: Manually compare it with the real world data from previous year.

4.2 Tests for Nonfunctional Requirements

Non-functional requirements for BC are given in SRS section 5.2. There are five non-functional requirements for BC, which are described in the following sections correspondingly.

4.2.1 NFR: Reliability

Reliability The reliability of the software is tested through the test for functional requirements in section 4.1.2 and the unit testing in section 5.

4.2.2 NFR: Usability

Usability

1. test-usability

   Type: Manual with potential users

   Initial State: The software is set up and ready for testing.

   Input/Condition: None

   Output/Result: Survey result about the user experience with the software.

   How test will be performed: Observe real users as they perform tasks with the software. Ask them to fill out a survey on their experience, including ease of use, intuitiveness, and any challenges encountered. The survey is in Section 6.2

4.2.3 NFR: Maintainability

Maintainability

1. test-maintainability

   Type: Code walkthrough with potential developers

   Initial State: None

   Input/Condition: None

   Output/Result: Feedback from fellow classmates or team members in Table 1.

   How test will be performed: Show the code and the other documents to fellow classmates/potential developers, ask them to explain the code to the author and see how much they could understand. Gather any feedback or questions they have.

4.2.4 NFR: Portability

Portability

1. test-portability

   Type: Manual

   Initial State: None

   Input/Condition: None

Output/Result: Result of compilation.

How test will be performed: Use Github Actions to test if the software could be installed in multiple machines.

4.2.5 NFR: Scalability

Scalability

1. test-scalability

   Type: Automatic

   Initial State: The software is set up and ready for testing.

   Input/Condition: Increase the workload by adding more data from traffic and climate models, starting from individual location to larger scale.

   Output/Result: Different response time on how the software handles the increased workload.

   How test will be performed: Record the response time for the software when given increasing amount of dataset, evaluate the different response time.

4.3 Traceability Between Test Cases and Requirements

	R1	R2	R3	R4	NFR1	NFR2	NFR3	NFR4	NFR5
Test 4.1.1	X								
Test 4.1.2			X						
Test 4.1.3		X		X					
Test 4.2.1					X				
Test 4.2.2						X			
Test 4.2.3							X		
Test 4.2.4								X	
Test 4.2.5									X

Table 6: Traceability Between Test Cases and Requirements

5 Unit Test Description

This section will be filled in after the MIS (detailed design document) has been completed.

[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access program, including one test for normal behaviour and as many tests as needed for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, you code needs to be well-documented, with meaningful names for all of the tests. —SS]

5.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

5.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

5.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

2. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

5.3.1 Module ?

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   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:

5.3.2 Module ?

...

5.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

Reference

1. Mingsai Xu, Yuxin Zheng, Cancan Yang (2024). Assessing Highway Bridge Chloride Exposure at a Provincial Scale: Mapping and Projecting Impacts of Climate Change. [Manuscript in preparation].

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

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

6.2 Usability Survey Questions

- From one to five and five being the most satisfied, how would you rate your overall experience using the software?
- Were the instructions clear and easy to understand. If no, please explain why.
- Did the software include all the features you expected? If no, what additional features would you like to see?
- From one to five and five being the most satisfied, how would you rate the speed and responsiveness of the software?