

# Verification and Validation Report: Bridge Corrosion

Cynthia Liu

April 15, 2024

# 1 Revision History

Date	Version	Notes
Apr. 14 2024	1.0	Initial release

## 2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [SRS](#). The symbols, abbreviations and acronyms used in [MG](#), [MIS](#), [VnVPlan](#) also applied here.

# Contents

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>ii</b>
<b>3</b>	<b>Functional Requirements Evaluation</b>	<b>1</b>
3.1	Input tests - Coordinate . . . . .	1
3.2	Input tests - Models . . . . .	1
3.3	Output tests - Chloride exposure . . . . .	1
<b>4</b>	<b>Nonfunctional Requirements Evaluation</b>	<b>1</b>
4.1	Reliability . . . . .	1
4.2	Usability . . . . .	1
4.3	Maintainability . . . . .	2
4.4	Portability . . . . .	2
<b>5</b>	<b>Unit Testing</b>	<b>3</b>
5.1	test_calculation . . . . .	3
5.2	test_model_input . . . . .	3
5.3	test_input_check . . . . .	3
5.4	test_search . . . . .	3
5.5	test_visualization . . . . .	3
<b>6</b>	<b>Changes Due to Testing</b>	<b>3</b>
<b>7</b>	<b>Automated Testing</b>	<b>4</b>
<b>8</b>	<b>Trace to Requirements</b>	<b>4</b>
<b>9</b>	<b>Trace to Modules</b>	<b>4</b>

## List of Tables

1	Traceability Between Test Cases and Requirements . . . . .	4
2	Traceability Between Test Cases and Modules . . . . .	5

This document includes the results of [VnVPlan](#).

## 3 Functional Requirements Evaluation

This section covers the tests of the functional requirements.

### 3.1 Input tests - Coordinate

This test classifies the valid and invalid input from user. It is covered by unit testing in [test\\_input.check.py](#). All the test cases passed successfully.

### 3.2 Input tests - Models

This test check if there are any missing values from the input climate and traffic model. It is achieved by unit testing in [test\\_model.input.py](#). All the test cases passed successfully.

### 3.3 Output tests - Chloride exposure

This test checks if the output is reliable by comparing the actual result with the result from Matlab, which is also implemented by the author as a second way to ensure the accuracy. It is achieved by unit testing in [test.calculation.py](#). All the test cases passed successfully.

## 4 Nonfunctional Requirements Evaluation

This section covers the tests of the non-functional requirements.

### 4.1 Reliability

The reliability is tested through section [3](#) and section [5](#).

### 4.2 Usability

The author did a demonstration during the final presentation and shared it with some friends. Feedback from participants included:

- The interactive buttons for output graphs were positioned too closely to the map.
- After obtaining results for one location, the map reverted to its initial state. This resulted in delays for users trying to compare two nearby locations.
- The Key and Value for the result list is confusing, participants suggested using more descriptive text to improve clarity.
- There could be the  $25 \times 25$  grid shown on the map to visualize the distance.

Due to time constraints, these comments were not addressed immediately. However, plans are in place to address them in the future to enhance the usability and clarity of the demonstration.

### 4.3 Maintainability

The author did a code review with friends from different background in McMaster and the most common comments are as follows:

- The relationship between modules are not clear without looking at module interface specification
- Functions lack comments or explanations regarding their purpose, which can make understanding the code more challenging.

The primary approach I used to enhance maintainability was to add comments and explanations in the code, ensuring easier comprehension for future developers.

### 4.4 Portability

The author used Github Actions to test if the software could be installed in multiple machines and environments. Some manual testing is done by asking fellows to run the project following the [Readme.md](#), most of the process went smoothly, thought there are some problems:

- When installing make, user need to restart the Terminal or PowerShell to make it work. This is added to the readme.

- To run the python files, “*python filename.py*” works on most computers while some need to be “*python3 filename.py*”. In the makefile it is set to python by default, but the alternative way to run section is added to readme for cases with python3.

## 5 Unit Testing

This section shows the result of the other unit testing for this software. To regenerate the result, you can run “*make test*” or “*pytest src/database/testfile.py*” in the root folder of this repo, replacing *testfile* with the actual file names.

### 5.1 test\_calculation

All the test cases pass successfully though there are some warnings. Check the [log](#) for detail.

### 5.2 test\_model\_input

All the test cases pass successfully. Check the [log](#) for detail.

### 5.3 test\_input\_check

All the test cases pass successfully. Check the [log](#) for detail.

### 5.4 test\_search

All the test cases pass successfully. Check the [log](#) for detail.

### 5.5 test\_visualization

All the test cases pass successfully. Check the [log](#) for detail.

## 6 Changes Due to Testing

Many minor bugs are fixed and more explanation are added for maintainability and portability. One important change is regarding precision: in the beginning the output results were rounded to two decimal places. However,

after careful consideration, it seemed like that this level of precision might not be sufficient. This is to be discussed with Dr. Yang and Mingsai, but I decide not to round it for now. Meanwhile, in the visualization, I still keep it as two decimal for readability when users putting mouses on the points on graph.

## 7 Automated Testing

Continuous Integration (CI) is used for automated testing. The test cases using pytest are set up in [GitHub Actions](#). Whenever there is a new push, CI will run through all test cases to check if everything's working as expected.

## 8 Trace to Requirements

	R1	R2	R3	NFR1	NFR2	NFR3	NFR4
<a href="#">t_calculation</a>		X	X	X			
<a href="#">t_model_input.py</a>	X						
<a href="#">t_input_check.py</a>	X						
<a href="#">t_search.py</a>		X					
<a href="#">t_visualization</a>					X		
Test Reliability				X			
Test Usability					X		
Test Maintainability						X	
Test Portability							X

Table 1: Traceability Between Test Cases and Requirements

## 9 Trace to Modules



	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14
<a href="#">t_calculation</a>					X	X	X	X	X	X	X	X	X
<a href="#">t_model_input</a>					X								
<a href="#">t_input_check</a>	X	X											
<a href="#">t_search.py</a>		X	X										
<a href="#">t_visualization</a>				X									

Table 2: Traceability Between Test Cases and Modules