Module Interface Specification for Bridge Corrosion

Cynthia Liu

March 21, 2024

1 Revision History

Date	**Version**	**Notes**
Mar. 8, 2024	1.0	Initial Release

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at SRS.

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

3 Introduction

The following document details the Module Interface Specifications for Bridge Corrosion which investigate how climate, traffic might impact corrosion-induced damage for reinforced concrete bridges by influencing the chloride exposure.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at here.

4 Notation

The structure of the MIS for modules comes from [HoffmanAndStrooper1995], with the addition that template modules have been adapted from [GhezziEtAl2003]. The mathematical notation comes from Chapter 3 of [HoffmanAndStrooper1995]. For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by BC.

Data Type	**Notation**	**Description**
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in $(-\infty, \infty)$
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$
empty	$\varnothing$	when the input is empty or the variable does not exist

The specification of BC uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, BC uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware Hiding	
Behaviour-Hiding Module	Input Parameter Module Input Verification Module Control Module Data Searching Module Output Visualization Module
Software Decision Module	Chloride Exposure Calculation Module Sequence Data Structure Module Plotting Module

Table 1: Module Hierarchy

6 MIS of Control Module

This module provides the main program and the GUI of the software. It is related to R1 and R2 in the SRS.

6.1 Module

main

6.2 Uses

- Input Parameter Module (Section **??**)
- Input Verification Module (Section **??**)
- Data Searching Module (Section **??**)
- Output Visualization Module (Section **??**)

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
main	-	-	-

6.4 Semantics

6.4.1 State Variables

- long, lat := long: $\mathbb{R}$, lat: $\mathbb{R}$ # coordinate get from Input Parameter Module.
- result = sequence of $\mathbb{R}$ # the predicted chloride exposure trend.

6.4.2 Environment Variables

None

6.4.3 Assumptions

None

6.4.4 Access Routine Semantics

main():

- transition: Control and execute the other modules as follow:
    - Get input from user by Input Parameter Module. (M2, Section **??**)
    - Pass the input to Input Verification Module. (M3, Section **??**)
    - Search the corresponding data in Data Searching Module if the input is valid. (M5, Section **??**)
    - Visualize the resulting data by using Output Visualization Module. (M6, Section **??**)
- output: out := None
- exception: exc := Exceptions are handled in other submodules.

6.4.5 Local Functions

None

7 MIS of Input Parameter Module

This module hides the format and structure of the input data, it is related to R1 in SRS.

7.1 Module

param

7.2 Uses

None

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
param	long: String, lat: String	long: $\mathbb{R}$, lat: $\mathbb{R}$	InputTypeMismatchError

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

Keyboard: this module takes input from the keyboard by typing.

7.4.3 Assumptions

None

7.4.4 Access Routine Semantics

This module load the input data and save it to the data type needed by the Input Verification Module.
read_input():

- transition: Get input from the user, pass the input to the Input Verification Module.
- output: out := None

- exception: exc:=

Expression	**Exception**
isNumeric(long) != float $\vee$ isNumeric(lat) != float	InputTypeMismatchError

7.4.5 Local Functions

isNumeric(e):

- output: out := Boolean # check if the input e is a string of float

8 MIS of Input Verification Module

This module verifies if the input is within the physical and software constraints, it is related to R1 in SRS.

8.1 Module

verify_param

8.2 Uses

Input Parameter Module (Section **??**)

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
read_geoinfo	filename: String	-	FileNotFoundError
verify_param	long: $\mathbb{R}$, lat: $\mathbb{R}$	Boolean	InputOutofOntarioError

8.4 Semantics

8.4.1 State Variables

- isValid: Boolean # if this input is valid

8.4.2 Environment Variables

- File: String # the geojson file that contain the shape of Ontario.

8.4.3 Assumptions

This module use the open source geojson file that contain the Ontario boundary, by drawing polygons with the vertex coordinate. The coordinates for those vertexes are assumed to be reliable.

8.4.4 Access Routine Semantics

read_geoinfo($filename$**)**:

- transition: access the data from geojson file
- exception: exc := FileNotFoundError: $filename == \varnothing$

verify_param($long, lat$**)**:

- output: out := isValid
- exception: exc := InputOutofOntarioError: $\neg((long, lat) \in Ontario)$

8.4.5 Local Functions

None

9 MIS of Data Searching Module

This module finds the data needed in the database, it is related to R2 in SRS.

9.1 Module

search

9.2 Uses

Input Parameter Module (Section **??**) , Chloride Exposure Calculation Module (Section **??**)

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
read_file	file: String	data: sequence of $\mathbb{R}$	FileNotFoundError
search	long: $\mathbb{R}$, lat: $\mathbb{R}$	result: sequence of $\mathbb{R}$	-

9.4 Semantics

9.4.1 State Variables

- data: sequence of $\mathbb{R}$ # the sequence of data read from database
- result: sequence of $\mathbb{R}$ # the sequence of predicted chloride exposure data

9.4.2 Environment Variables

None

9.4.3 Assumptions

All locations within Ontario must contain valid data. If a user inputs a location outside of Ontario, it will be handled by the Input Verification Module.

9.4.4 Access Routine Semantics

read_file($filename$):

- output: out := data # sequence of $\mathbb{R}$ read from the file
- exception: exc := FileNotFoundError = $filename == \varnothing$

search($long, lat, data$):

- output: out := result = search(find_grid_belonged($long, lat, data$))
- exception: exc := None

9.4.5 Local Functions

find_grid_belonged($long, lat, data$): If the input coordinate is not the exact one in data, find the grid that it belongs to and return the center coordinate.

- output: out := long, lat

10 MIS of Output Visualization Module

This module provides the cisualization of the resulting chloride exposure trend, it is related to R2 and R4 in SRS.

10.1 Module

draw

10.2 Uses

Data Searching Module (Section **??**)

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
draw	result: sequence of $\mathbb{R}$	graphs	-

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

Screen: The graphs are displayed on the screen.

10.4.3 Assumptions

None

10.4.4 Access Routine Semantics

draw($long, lat, result$):

- transition: display the graphs using the result from Data Searching Module.
- output: out := None
- exception: exc := None

10.4.5 Local Functions

None

11 MIS of Chloride Exposure Calculation Module

This module performs the calculation process to generate the database, it is related to the R2, R3, R4 in SRS.

11.1 Module

calculate

11.2 Uses

None

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
calculate	AADT: sequence of $\mathbb{R}$, AADTT: sequence of $\mathbb{R}$, t1: sequence of $\mathbb{N}$, h_{total}: sequence of $\mathbb{R}$, t2: sequence of $\mathbb{N}$	file: String	DataMissingError, DataInvalidError

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Environment Variables

File: Sequence of String # the result of calculation will be stored in an output csv file.

11.4.3 Assumptions

- The AADT and AADTT are assumed to have 2% increase rate every year.
- The map of Ontario is divided into multiple 25km * 25km grid (as mentioned in SRS) and the coordinates are the center of those grids. The locations inside each grid are consider to have same chloride exposure rate.

11.4.4 Access Routine Semantics

calculate($AADT, AADTT, t1, h_{total}, t2$):

- output: out := file = Cl_mass($spray\_density\_Cl(salt\_per\_day(h_{total}, t1),$ $water\_thickness(h_{total}, t2)), adjust(AADT, AADTT), t2$) # the result is saved to a csv file storing calculation result, which is the prediction of chloride exposure rate. The file has row lable as coordinate and column lable as year.

- exception: exc:=

Expression	**Exception**
$\exists e \in [AADT, AADTT, h_{total}, t1, t2], e = \varnothing$	DataMissingError
$(\exists i \in [0..\lvert AADT \rvert - 1], AADTT[i] > AADT[i]) \vee (\neg(t1, t2 \in (0, 365)))$	DataInvalidError

11.4.5 Local Functions

adjust($AADT, AADTT$): A function that convert AADT and AADTT from sequence to 2-D array, based on the assumption of annually 2% increase.

- out := AADT, AADTT

salt_per_day($h\_total, t1$): calcualte the quantity of deicing salts applied on a roadway per day during the winter season.

- out := M_app

water_thickness($h\_total, t2$): calculate the thickness of melted water per day with snow melting

- out := h_app

spray_density_Cl(M_app, h_app): calculate the mass of chloride ions by one truck

- out := SD_totalCl

Cl_mass($SD\_totalCl, AADT, AADTT, t2$): calculate the final chloride exposure

- out := Cl_result