Module Interface Specification for Bridge Corrosion

Cynthia Liu

March 11, 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

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 Introduction **1**

4 Notation **1**

5 Module Decomposition **1**

6 MIS of Control Model **3**
6.1 Module . . . 3
6.2 Uses . . . 3
6.3 Syntax . . . 3
6.3.1 Exported Constants . . . 3
6.3.2 Exported Access Programs . . . 3
6.4 Semantics . . . 3
6.4.1 State Variables . . . 3
6.4.2 Environment Variables . . . 4
6.4.3 Assumptions . . . 4
6.4.4 Access Routine Semantics . . . 4
6.4.5 Local Functions . . . 4

7 MIS of Input Parameter Model **5**
7.1 Module . . . 5
7.2 Uses . . . 5
7.3 Syntax . . . 5
7.3.1 Exported Constants . . . 5
7.3.2 Exported Access Programs . . . 5
7.4 Semantics . . . 5
7.4.1 State Variables . . . 5
7.4.2 Environment Variables . . . 5
7.4.3 Assumptions . . . 5
7.4.4 Access Routine Semantics . . . 6
7.4.5 Local Functions . . . 6

8 MIS of Input Verification Model **7**
8.1 Module . . . 7
8.2 Uses . . . 7
8.3 Syntax . . . 7
8.3.1 Exported Constants . . . 7
8.3.2 Exported Access Programs . . . 7

8.4 Semantics . . . 7
8.4.1 State Variables . . . 7
8.4.2 Environment Variables . . . 7
8.4.3 Assumptions . . . 8
8.4.4 Access Routine Semantics . . . 8
8.4.5 Local Functions . . . 8

9 MIS of Data Searching Model **9**
9.1 Module . . . 9
9.2 Uses . . . 9
9.3 Syntax . . . 9
9.3.1 Exported Constants . . . 9
9.3.2 Exported Access Programs . . . 9
9.4 Semantics . . . 9
9.4.1 State Variables . . . 9
9.4.2 Environment Variables . . . 9
9.4.3 Assumptions . . . 10
9.4.4 Access Routine Semantics . . . 10
9.4.5 Local Functions . . . 10

10 MIS of Output Visualization Model **11**
10.1 Module . . . 11
10.2 Uses . . . 11
10.3 Syntax . . . 11
10.3.1 Exported Constants . . . 11
10.3.2 Exported Access Programs . . . 11
10.4 Semantics . . . 11
10.4.1 State Variables . . . 11
10.4.2 Environment Variables . . . 11
10.4.3 Assumptions . . . 11
10.4.4 Access Routine Semantics . . . 12
10.4.5 Local Functions . . . 12

11 MIS of Chloride Exposure Calculation Model **13**
11.1 Module . . . 13
11.2 Uses . . . 13
11.3 Syntax . . . 13
11.3.1 Exported Constants . . . 13
11.3.2 Exported Access Programs . . . 13
11.4 Semantics . . . 13
11.4.1 State Variables . . . 13
11.4.2 Environment Variables . . . 13
11.4.3 Assumptions . . . 13

11.4.4 Access Routine Semantics 14
11.4.5 Local Functions 14

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. [Fill in your project name and description —SS]

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

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

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 (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

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 Model Sequence Data Structure Module Plotting Module

Table 1: Module Hierarchy

6 MIS of Control Model

[Use labels for cross-referencing —SS]
[You can reference SRS labels, such as R1. —SS]
[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

6.1 Module

main

6.2 Uses

- Hardware Hiding Module
- Input Parameter Module
- Input Verification Module
- Data Searching Module
- Output Visualization Module
- Chloride Exposure Calculation Model
- Sequence Data Structure Module
- Plotting Module

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: a pair of input of [long: $\mathbb{R}$, lat: $\mathbb{R}$] #input coordinate get from Input Parameter Module. [Not all modules will have state variables. State variables give the module a memory. —SS]

6.4.2 Environment Variables

None

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

6.4.3 Assumptions

None [Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

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 7)
  - Pass the input to Input Verification Module. (M3, Section 8)
  - Search the corresponding data in Data Searching Module if the input is valid. (M5, Section 9)
  - Visualize the resulting data by using Output Visualization Module. (M6, Section 10)
- output: None
- exception: Exception is handled in Section 8

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

6.4.5 Local Functions

None [As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

7 MIS of Input Parameter Model

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R1. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

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	pair of [long: $\mathbb{R}$, lat: $\mathbb{R}$]	-	InputMissingError, InputTypeMismatchError

7.4 Semantics

7.4.1 State Variables

long, lat := a pair of [long: $\mathbb{R}$, lat: $\mathbb{R}$] #input coordinate [Not all modules will have state variables. State variables give the module a memory. —SS]

7.4.2 Environment Variables

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

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

7.4.3 Assumptions

None [Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

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: None
- exception: None

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

7.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

8 MIS of Input Verification Model

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R1. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

8.1 Module

verify_param

8.2 Uses

Input Parameter Module

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
verify_param	[long: $\mathbb{R}$, lat: $\mathbb{R}$]	Boolean	InputOutofOntarioError

8.4 Semantics

8.4.1 State Variables

- long, lat: a pair of [long: $\mathbb{R}$, lat: $\mathbb{R}$] # input coordinate
- isValid: Boolean # if this input is valid
- geojson: TODO

[Not all modules will have state variables. State variables give the module a memory. —SS]

8.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

8.4.3 Assumptions

This module use the open source geojson file TODO and the vertex coordinate there is assumed to be true. [Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

8.4.4 Access Routine Semantics

verify_param(long, lat):

- output: Boolean
- exception: InputOutofOntarioError

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

8.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

9 MIS of Data Searching Model

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R1. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

9.1 Module

search

9.2 Uses

Input Parameter Module, Chloride Exposure Calculation Module

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
read_file	filename: String	data: sequence of $\mathbb{R}$	FileNotFoundError
search	[long: $\mathbb{R}$, lat: $\mathbb{R}$]	result: [cl: $\mathbb{R}$]	-

9.4 Semantics

9.4.1 State Variables

- long, lat: a pair of [long: $\mathbb{R}$, lat: $\mathbb{R}$] # input coordinate
- filename: String # the database generated by Calculation Module
- data: sequence of $\mathbb{R}$ # the sequence of data read from database
- cl: $\mathbb{R}$ # predicted chloride exposure data at one time, one location
- result: [cl: $\mathbb{R}$] # the list of predicted chloride exposure data

[Not all modules will have state variables. State variables give the module a memory. —SS]

9.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

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. [Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

9.4.4 Access Routine Semantics

read_file(filename):

- transition: read the data in the database file
- output: data := sequence of [cl: $\mathbb{R}$]
- exception: FileNotFoundError := Could not find such file

search(long, lat, data):

- transition: if the input coordinate do not have a corresponding value in the data, (long, lat) := find_grid_belonged(long, lat, data). Use the new (long, lat) to do the searching.
- output: result := [cl: $\mathbb{R}$]
- exception: none

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

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: (long, lat) := $\mathbb{R}$

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

10 MIS of Output Visualization Model

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R1. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

10.1 Module

draw

10.2 Uses

Data Searching Module

10.3 Syntax

10.3.1 Exported Constants

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
draw	result: [cl: $\mathbb{R}$]	graphs	-

10.4 Semantics

10.4.1 State Variables

- long, lat: a pair of [long: $\mathbb{R}$, lat: $\mathbb{R}$] # input coordinate
- cl: $\mathbb{R}$ # predicted chloride exposure data at one time, one location
- result: [cl: $\mathbb{R}$] # the list of predicted chloride exposure data

[Not all modules will have state variables. State variables give the module a memory. —SS]

10.4.2 Environment Variables

Screen: The graphs are displayed on the screen. [This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

10.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

10.4.4 Access Routine Semantics

draw(long, lat, result):

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

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

10.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

11 MIS of Chloride Exposure Calculation Model

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R1. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

11.1 Module

calculate

11.2 Uses

none

11.3 Syntax

11.3.1 Exported Constants

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: an output file	DataMissingError, DataInvalidError

11.4 Semantics

11.4.1 State Variables

none [Not all modules will have state variables. State variables give the module a memory. —SS]

11.4.2 Environment Variables

The result of calculation will be stored in an output csv file. [This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

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.

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

11.4.4 Access Routine Semantics

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

- transition: result:= Cl_mass(spray_density_Cl(salt_per_day(h_{total}, t1), water_thickness(h_{total}, t2)), adjust(AADT, AADTT), t2). See Local Functions for step by step calculation.
- output: 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

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

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

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]