Module Guide for Bridge Corrosion

Cynthia Liu

April 7, 2024

1 Revision History

Date	Version	Notes
March 6, 2024	0.0	Initial release
April 5, 2024	0.1	Edit according to feedback from peer review and Dr. Smith

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	**description**
AC	Anticipated Change
BC	Bridge Corrosion
DAG	Directed Acyclic Graph
GUI	Graphical User Interface
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SAS	Sprayed and Splashed
SC	Scientific Computing
SRS	Software Requirements Specification
UC	Unlikely Change

Contents

List of Tables

List of Figures

3 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team. We advocate a decomposition based on the principle of information hiding. This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by, as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented in only one module.
- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed. The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility, and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 9 describes the use relation between modules.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: Coordinate outside Ontario might be considered if the project becomes a larger scale. This is relevant to R1 in SRS.

AC2: The algorithm for calculating chloride exposure might change if the theory proceeds.

AC3: The output could have another format which is saving the search result to file.

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: The traffic data model and climate data model are inputed from developer and will be hidden to user.

UC2: The user input is only the coordinate.

UC3: There will be visualization for the output.

UC4: The database storing chloride exposure data will always be external.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.
The whole project could be divided into two parts. M1 to M5 are the software that user will actually interact with, and M6 to M14 are the modules that generate the chloride exposure database that the software depends on. The purpose for divding the process of generating

the database is because each module hides a secret of the calculation formula, so if one of the formula changes only the corresponding module need to be changed. M15 and M16 are the built in modules by Python that is used during the implementation.

M1: Hardware Hiding Module

M2: Input Module

M3: Control Module

M4: Data Searching Module

M5: Output Visualization Module

M6: Data Model Reading Module

M7: Constant Module

M8: Deicing Salts Calculation Module

M9: Melted Water Thickness Calculation Module

M10: Single Water SAS Calculation Module

M11: Single Chloride Ions SAS Calculation Module

M12: All Chloride Ions SAS Calculation Module

M13: Deicing Salts Deposition Calculation Module

M14: Chloride Exposure Database Generation Module

M15: File I/O Module

M16: Plotting Module

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 2.

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Module Control Module Data Searching Module Output Visualization Module Data Model Reading Module Constant Module Deicing Salts Calculation Module Melted Water Thickness Calculation Module Single Water SAS Calculation Module Single Chloride Ions SAS Calculation Module All Chloride Ions SAS Calculation Module Deicing Salts Deposition Calculation Module Chloride Exposure Database Generation Module
Software Decision Module	File I/O Module Plotting Module

Table 1: Module Hierarchy

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding". The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *ProgName* means the module will be implemented by the ProgName software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented.

7.1 Hardware Hiding Modules (M1)

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

7.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

7.2.1 Input Module (M2)

Secrets: The kind of valid and invalid input coordinate data.

Services: Get input from the user and verify if the input is within the physical and software constraint, throw warning message if it is not.

Implemented By: BC

Type of Module: Abstract Data Type

7.2.2 Control Module (M3)

Secrets: The algorithm for running the program.

Services: Provide the main program and the GUI.

Implemented By: BC

Type of Module: Library

7.2.3 Data Searching Module (M4)

Secrets: The irrelevant values in database.

Services: Find the data needed in the database by searching algorithm.

Implemented By: BC

Type of Module: Abstract Data Type

7.2.4 Output Visualization Module (M5)

Secrets: The format and structure of the output chloride exposure data.

Services: Provide the visualization of the resulting chloride exposure trend, using line graph or histogram.

Implemented By: BC

Type of Module: Abstract Data Type

7.2.5 Data Model Reading Module (M6)

Secrets: The origin climate model and traffic model that the calculation is based on.

Services: Read data from the climate model and traffic model and store them in the data structure needed for further calculation.

Implemented By: BC

Type of Module: Abstract Data Type

7.2.6 Constant Module (M7)

Secrets: The constant used during calculation.

Services: Store the constant for mantainability purpose.

Implemented By: BC

Type of Module: Abstract Data Type

7.2.7 Deicing Salts Calculation Module (M8)

Secrets: The formula to determine the quantity of deicing salts applied per day with snowfall.

Services: Calculate and store the quantity of deicing salts applied per day with snowfall.

Implemented By: BC

Type of Module: Abstract Object

7.2.8 Melted Water Thickness Calculation Module (M9)

Secrets: The formula to determine the thickness of melted water per day with snow melting.

Services: Calculate and store the thickness of melted water per day with snow melting.

Implemented By: BC

Type of Module: Abstract Object

7.2.9 Single Water SAS Calculation Module (M10)

Secrets: The formula to determine water sprayed and splashed by one truck using a Computational Fluid Dynamics (CFD)-based analytical model.

Services: Calculate and store the water sprayed and splashed by one truck.

Implemented By: BC

Type of Module: Abstract Object

7.2.10 Single Chloride Ions SAS Calculation Module (M11)

Secrets: The formula to determine chloride ions sprayed and splashed by one truck.

Services: Calculate and store the chloride ions sprayed and splashed by one truck.

Implemented By: BC

Type of Module: Abstract Object

7.2.11 All Chloride Ions SAS Calculation Module (M12)

Secrets: The formula to determine chloride ions sprayed and splashed by all vehicles in one winter season.

Services: Calculate and store the chloride ions sprayed and splashed by all vehicles in one winter season.

Implemented By: BC

Type of Module: Abstract Object

7.2.12 Deicing Salts Deposition Calculation Module (M13)

Secrets: The formula to determine the deposition of deicing salts on the surface of the bridge substructure.

Services: Calculate and store the deposition of deicing salts on the surface of the bridge substructure.

Implemented By: BC

Type of Module: Abstract Object

7.2.13 Chloride Exposure Database Generation Module (M14)

Secrets: The detail formula for calculating the chloride exposure database.

Services: Save the chloride exposure data to the database to be used by the user end.

Implemented By: BC

Type of Module: Abstract Object

7.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

7.3.1 File I/O (M15)

Secrets: The data structure and algorithms for File I/O

Services: Provides functions that could read from file and write to new file.

Implemented By: Pandas

Type of Module: Library

7.3.2 Plotting Result Module (M16)

Secrets: The data structure and algorithms for plotting data graphically.

Services: Provides plot function that can plot the results from Data Searching Module, used by Output Visualization module.

Implemented By: Plotly

Type of Module: Library

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
R1	M1, M2, M3
R2	M3, M4, M5, M16
R3	M5, M14
NFR1	M4, M14
NFR2	M3, M4, M16
NFR3	all modules
NFR4	M3
NFR5	all modules

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M2
AC2	one of M8, M9, M10, M11, M12, M13
AC3	M3

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

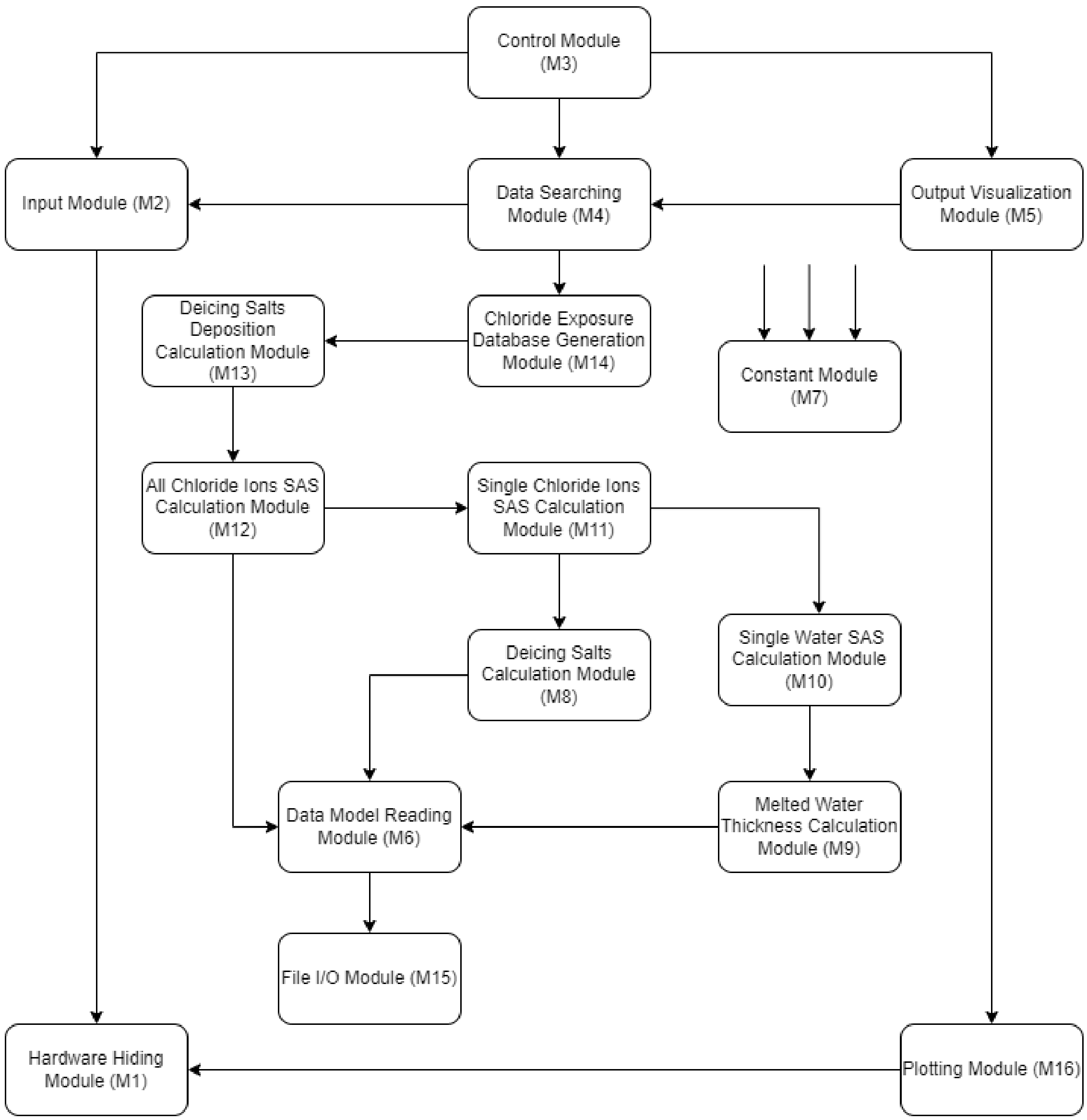

Figure 1: Use hierarchy among modules

10 User Interfaces

Figure 2: GUI illustration before inputing coordinate

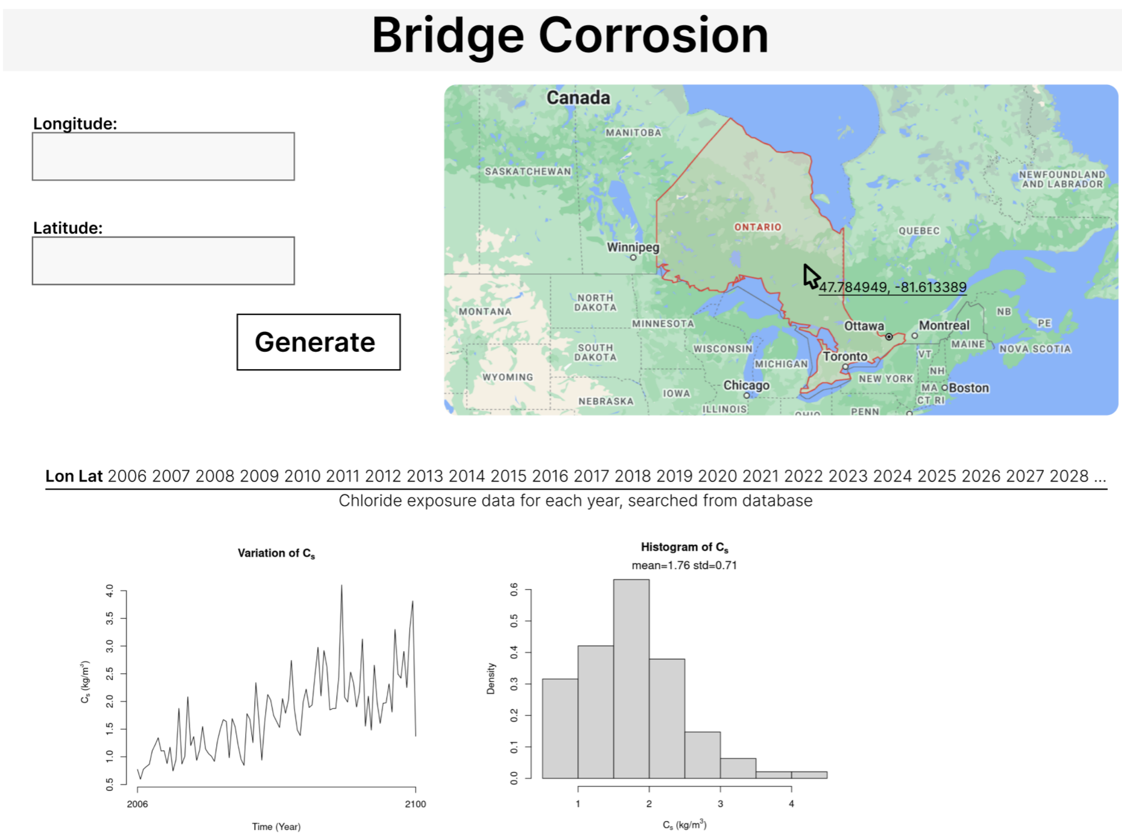

Figure 3: GUI illustration after inputing coordinate

11 Timeline

The specific timeline for module implementation is shown in below table:

Modules	Finish by	Responsible
M6, M7	March 2, 2024	Cynthia Liu
M8, M9, M10, M11, M12, M13	March 9, 2024	Cynthia Liu
M14	March 11, 2024	Cynthia Liu
M2	March 13, 2024	Cynthia Liu
M4	March 20, 2024	Cynthia Liu
M5	March 27, 2024	Cynthia Liu
M3	March 30, 2024	Cynthia Liu

Table 4: Timeline

12 Appendix

This section includes a link to SRS for requirements.