Module Guide for MTOBridge

Team 15, Alpha Software Solutions
Badawy, Adham
Yazdinia, Pedram
Jandric, David
Vakili, Farzad
Vezina, Victor
Chiu, Darren

April 4, 2023

1 Revision History

Date	Developer(s)	Notes
2023-01-17	Adham	Added Sections 4,5,7, and 8
2023-01-17	Farzad	Added Introduction
2023-01-17	Victor	Added Use Hierarchy
2023-04-4	Pedram	Rev1 Changes

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
CHBDC	Canadian Highway Bridge Design Code
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
MTOBridge	A portable Software program capable of helping bridge engineers
OS	Operating System
R	Requirement
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 (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). 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 laid out by Parnas et al. (1984), 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 (Parnas et al., 1984). 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, the feasibility of the decomposition, and flexibility of the design.

3.1 System Purpose

This project aims to develop an application MTOBridge that leverages novel bridge analysis methods and better visualizes their outputs while enhancing the user experience. This enables the refined methods of analysis to become routine for bridge design and evaluation. Moreover, A unique feature of MTOBridge which is otherwise unavailable in other bridge design/evaluation software, is that a direct comparison can be obtained between the results

obtained from the CHBDC approximate/simplified approach and those from the refined analysis. This endeavor will facilitate confidence in using refined analysis for bridge evaluation and design.

3.2 Document Purpose

The Purpose of this document is to demonstrate how the system is decomposed into modules and communicate the overall design of the system which satisfies the requirements mentioned in the SRS. As a result helping future project members, maintainers, and designers to easily identify the parts of the software.

3.3 System Scope

The system is meant for use in offices (including home offices) by civil engineers designing or retrofitting bridges.

3.4 Document Scope

The scope of this document includes modules created by the development team and not components provided by the stakeholder (such as the MATLAB engines) or third-party libraries.

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.

1. How the set of backend MATLAB functions will be called

2. How the calculation call will be visualized
3. The format of the truck platoon input data
4. How the truck platoon input data is stored
5. The method of visualizing the truck platoon configuration
6. The method of visualizing the truck platoon data saving
7. The method of visualizing the truck platoon data loading
8. The format of the bridge input data
9. How the bridge input data is stored
10. The method of visualizing the bridge configuration
11. The method of visualizing the bridge data saving
12. The method of visualizing the bridge data loading
13. How the solver configuration data is stored
14. The method of visualizing the solver configuration
15. The format of the solver configuration data
16. How the concerned section specification data is stored
17. How the critical section specification data is stored
18. The method of visualizing the solver configuration data saving
19. The method of visualizing the solver configuration data loading
20. How data is saved to the file system
21. How data is loaded from the file system
22. The format of the output report created by the program
23. The method by which the Output Report is visualized for the user
24. How the output report data is stored
25. The method of visualizing the output report saving
26. The set of backend MATLAB calculations called by the program

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 this 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.

- The language the backend calculations are written in
- The operating system the program will run on
- The input/output devices available to the user
- The baseline level of technical knowledge of the users of the software
- The primary functions of the software (i.e, visually and textually representing the results of forces imparted on some bridge by some truck platoon)
- The publicity of the software (expected to be a private in-house software used only by the MTO)
- The number of solvers available to choose between

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented. For clarity, The three top level categories as well and all leaf nodes are bolded.

— **Hardware Hiding**

— Data Storage Hiding

— Input Data Storage Hiding

— **Truck Platoon Configuration Data Storage Hiding**

— **Bridge Configuration Data Storage Hiding**

— Calculation Data Storage Hiding

— Calculation Variable Data Storage Hiding

— **Solver Choice Data Storage Hiding**

— **Concerned Section Specification Data Storage Hiding**

— **Critical Section Specification Data Storage Hiding**

— **Output Report Data Storage Hiding**

- File System Hiding
  - **Save To File System Hiding**
  - **Load From File System Hiding**

- **Software Decision Hiding**
  - **Calculation Call Hiding**
  - **MATLAB Code Hiding**
  - Data Format Hiding
    - Configuration Data Format Hiding
      - **Truck Platoon Configuration Data Format Hiding**
      - **Bridge Configuration Data Format Hiding**
      - **Solver Configuration Data Format Hiding**
    - **Report Data Format Hiding**

- **Behavior Hiding**
  - Visualization Hiding
    - Configuration Visualization Hiding
      - **Truck Platoon Configuration Visualization Hiding**
      - **Bridge Configuration Visualization Hiding**
    - Calculation Visualization Hiding
      - **Calculation Call Visualization Hiding**
      - **Solver Configuration Visualization Hiding**
    - **Output Report Visualization Hiding**
    - File System Visualization Hiding
      - Save To File System Visualization Hiding
        - **Truck Platoon Configuration Save To File System Visualization Hiding**
        - **Bridge Configuration Save To File System Visualization Hiding**
        - **Solver Configuration Save To File System Visualization Hiding**
        - **Output Report Save To File System Visualization Hiding**
      - Load From File System Visualization Hiding
        - **Truck Platoon Configuration Load From File System Visualization Hiding**
        - **Bridge Configuration Load From File System Visualization Hiding**
        - **Solver Configuration Load From File System Visualization Hiding**

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 1.

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). 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. *MTOBridge* means the module will be implemented by the MTOBridge 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

Data Storage Hiding

Secrets: How to store all program data on the machine

Services: Takes data configurations or calculation results and stores them for later use by the program.

Implemented By: –

Input Data Storage Hiding

Secrets: How to store all input data on the machine

Services: Takes data configurations and stores them for later use by the program

Implemented By: –

Truck Platoon Configuration Data Storage Hiding

Secrets: How to store truck platoon configuration data on the machine

Services: Takes a truck platoon configuration and stores it for later use by the program

Implemented By: OS

Bridge Configuration Data Storage Hiding

Secrets: How to store bridge configuration data on the machine

Services: Takes a bridge configuration and stores it for later use by the program

Implemented By: OS

Calculation Data Storage Hiding

Secrets: How to store all calculation related data on the machine

Services: Takes a user calculation input data or calculation results and stores them

Implemented By: –

Calculation Variable Data Storage Hiding

Secrets: How to store all calculation variable related data on the machine

Services: Takes a calculation variable and stores it for future use

Implemented By: –

Solver Choice Data Storage Hiding

Secrets: How to store solver choice data on the machine

Services: Takes a solver choice and stores it for future use

Implemented By: OS

Concerned Section Specification Data Storage Hiding

Secrets: How to store Concerned Section Specification data on the machine

Services: Takes a Concerned Section Specification and stores it for future use

Implemented By: OS

Critical Section Specification Data Storage Hiding

Secrets: How to store Critical Section Specification data on the machine

Services: Takes a Critical Section Specification and stores it for future use

Implemented By: OS

Output Report Data Storage Hiding

Secrets: How to store output report data on the machine

Services: Takes an output report data format and stores it for later use

Implemented By: OS

File System Hiding

Secrets: How to save/load data to/from the file system from/to the program runtime

Services: Provides a durable external storage system that outlasts program runtime

Implemented By: –

Save To File System Hiding

Secrets: How to save data to the file system from the program runtime

Services: Allows the program to save data in a durable manner that outlasts the program runtime.

Implemented By: MTOBridge

Load From File System Hiding

Secrets: How to load data from the file system to the program runtime

Services: Allows the program to load data from a durable source that outlasts the program runtime.

Implemented By: MTOBridge

7.2 Software Decision Modules

Calculation Call Hiding

Secrets: How to call the calculations to have them provide the necessary data for the software to visualize

Services: Allows the user to call the backend calculations to provide simulation results

Implemented By: MTOBridge

MATLAB Code Hiding

Secrets: How to calculate the demands imposed on a given bridge by a given truck platoon

Services: Provides the outputs of the relevant calculations to the software using the backend MATLAB functions for later use

Implemented By: Dr Yang's Lab

Data Format Hiding

Secrets: How to manage all input/output data in the program

Services: Takes input from User or saved files and converts them into appropriate input/output data.

Implemented By: –

Data Format Hiding

Secrets: How to manage all user input data regarding Truck Platoon/Bridge/Solver configuration

Services: Takes User input data and converts into a Truck Platoon/Bridge/Solver configuration for use by the program

Implemented By: –

Truck Platoon Configuration Data Format Hiding

Secrets: How to manage all user input data regarding Truck Platoon configuration

Services: Takes User input data and converts it into a Truck Platoon configuration for use by the program

Implemented By: MTOBridge

Bridge Configuration Data Format Hiding

Secrets: How to manage all user input data regarding Bridge configuration

Services: Takes User input data and converts it into a Bridge configuration for use by the program

Implemented By: MTOBridge

Solver Configuration Data Format Hiding

Secrets: The format of the User's Solver Configuration data

Services: Provides a data structure to hold the relevant data

Implemented By: MTOBridge

Report Configuration Data Format Hiding

Secrets: The format of the output report

Services: Takes visualization and calculation results from the appropriate modules and converts them to an output file format.

Implemented By: MTOBridge

7.3 Behavior-Hiding Modules

Visualization Hiding

Secrets: How to visualize all information in the software

Services: Allows the user to graphically interact with the program for various I/O purposes

Implemented By: –

Configuration Visualization Hiding

Secrets: How to visualize the truck platoon or bridge configuration process

Services: Allows the user to graphically interact with the program to choose a configuration and see a preview

Implemented By: –

Truck Platoon Configuration Visualization Hiding

Secrets: How to visualize the truck platoon configuration process

Services: Allows the user to graphically interact with the program to choose a truck platoon configuration and see a preview

Implemented By: MTOBridge

Bridge Configuration Visualization Hiding

Secrets: How to visualize the bridge configuration process

Services: Allows the user to graphically interact with the program to choose a bridge configuration and see a preview

Implemented By: MTOBridge

Calculation Visualization Hiding

Secrets: How to visualize everything to do with the backend MATLAB Calculations

Services: Allows the user to graphically interact with the program for various calculation I/O purposes.

Implemented By: –

Calculation Call Visualization Hiding

Secrets: How to visualize the calling of the backend MATLAB calculations

Services: Allows the user to graphically interact with the program to call the MATLAB calculations.

Implemented By: MTOBridge

Solver Configuration Visualization Hiding

Secrets: How to visualize the solver configuration choices

Services: Allows the user to graphically interact with the program to configure the solver

Implemented By: MTOBridge

Output Report Visualization Hiding

Secrets: How to visualize the output report

Services: Allows the user to graphically interact with the program to see the output report

Implemented By: MTOBridge

File System Visualization Hiding

Secrets: How to visualize everything to do with saving/loading data to/from the file system from/to the program runtime

Services: Allows the user to graphically interact with the program for various file system related I/O purposes

Implemented By: –

Save To File System Visualization Hiding

Secrets: How to visualize everything to do with saving data to the file system from the program runtime

Services: Allows the user to graphically interact with the program to save their program data to the file system for later use.

Implemented By: –

Truck Platoon Configuration Save To File System Visualization Hiding

Secrets: How to visualize the saving of truck platoon configuration data

Services: Allows the user to graphically interact with the program to save their truck platoon configuration data to the file system for later use.

Implemented By: MTOBridge

Bridge Configuration Save To File System Visualization Hiding

Secrets: How to visualize the saving of bridge configuration data

Services: Allows the user to graphically interact with the program to save their bridge configuration data to the file system for later use.

Implemented By: MTOBridge

Solver Configuration Save To File System Visualization Hiding

Secrets: How to visualize the saving of solver configuration data

Services: Allows the user to graphically interact with the program to save their solver configuration data to the file system for later use.

Implemented By: MTOBridge

Output Report Save To File System Visualization Hiding

Secrets: How to visualize the saving of output report data

Services: Allows the user to graphically interact with the program to save their output report data to the file system for later use.

Implemented By: MTOBridge

Load From File System Visualization Hiding

Secrets: How to visualize everything to do with loading data from the file system to the program runtime

Services: Allows the user to graphically interact with the program to load their file system data to the program runtime for use.

Implemented By: –

Truck Platoon Configuration Load From File System Visualization Hiding

Secrets: How to visualize the loading of truck platoon configuration data

Services: Allows the user to graphically interact with the program to load their truck platoon configuration data to the program runtime for use.

Implemented By: MTOBridge

Bridge Configuration Load From File System Visualization Hiding

Secrets: How to visualize the loading of bridge configuration data

Services: Allows the user to graphically interact with the program to load their Bridge configuration data to the program runtime for use.

Implemented By: MTOBridge

Solver Configuration Load From File System Visualization Hiding

Secrets: How to visualize the loading of solver configuration data

Services: Allows the user to graphically interact with the program to load their solver configuration data to the program runtime for use.

Implemented By: MTOBridge

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**
FR.1	Calculation Call Hiding, Calculation Call Visualization Hiding, and MATLAB Code Hiding
FR.2	Truck Platoon Configuration Data Storage/Format Hiding and Truck Platoon Configuration Visualization Hiding
FR.3	Truck Platoon Configuration Visualization Hiding
FR.4	Save to File System Hiding and Truck Platoon Config. Save to File System Visualization Hiding
FR.5	Load from File System Hiding and Truck Platoon Config. Load from File System Visualization Hiding
FR.6	Bridge Configuration Data Storage/Format Hiding and Bridge Configuration Visualization Hiding
FR.7	Bridge Configuration Visualization Hiding
FR.8	Save to File System Hiding and Bridge Config. Save to File System Visualization Hiding
FR.9	Load from File System Hiding and Bridge Config. Load from File System Visualization Hiding
FR.10/11/12/13	Solver Configuration Data Format/Storage Hiding and Visualization
FR.14/15/16	Output Report Data Format/Storage Hiding and Visualization, as well as Report Save/Load Visualization

Table 1: Trace Between Requirements and Modules

AC	Modules
AC.1	Calculation Call Hiding
AC.2	Calculation Call Visualization Hiding
AC.3	Truck Platoon Configuration Data Format Hiding
AC.4	Truck Platoon Configuration Data Storage Hiding
AC.5	Truck Platoon Configuration Visualization Hiding
AC.6	Truck Platoon Configuration Save to File System Visualization Hiding
AC.7	Truck Platoon Configuration Load From File System Visualization Hiding
AC.8	Bridge Configuration Data Format Hiding
AC.9	Bridge Configuration Data Storage Hiding
AC.10	Bridge Configuration Visualization Hiding
AC.11	Bridge Configuration Save to File System Visualization Hiding
AC.12	Bridge Configuration Load From File System Visualization Hiding
AC.13	Solver Configuration Data Storage Hiding
AC.14	Solver Choice Visualization Hiding
AC.15	Solver Choice Data Format Hiding
AC.16	Concerned Section Specification Data Storage Hiding
AC.17	Critical Section Specification Data Storage Hiding
AC.18	Solver Configuration Save to File System Visualization Hiding
AC.19	Solver Configuration Load from File System Visualization Hiding
AC.20	Save to File System
AC.21	Load From File System
AC.22	Output Report Data Format Hiding
AC.23	Output Report Visualization Hiding
AC.24	Output Report Data Storage Hiding
AC.25	Output Report Save to File System Visualization Hiding
AC.26	MATLAB Code Hiding

Table 2: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas (1978) said of two programs A and B that 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.

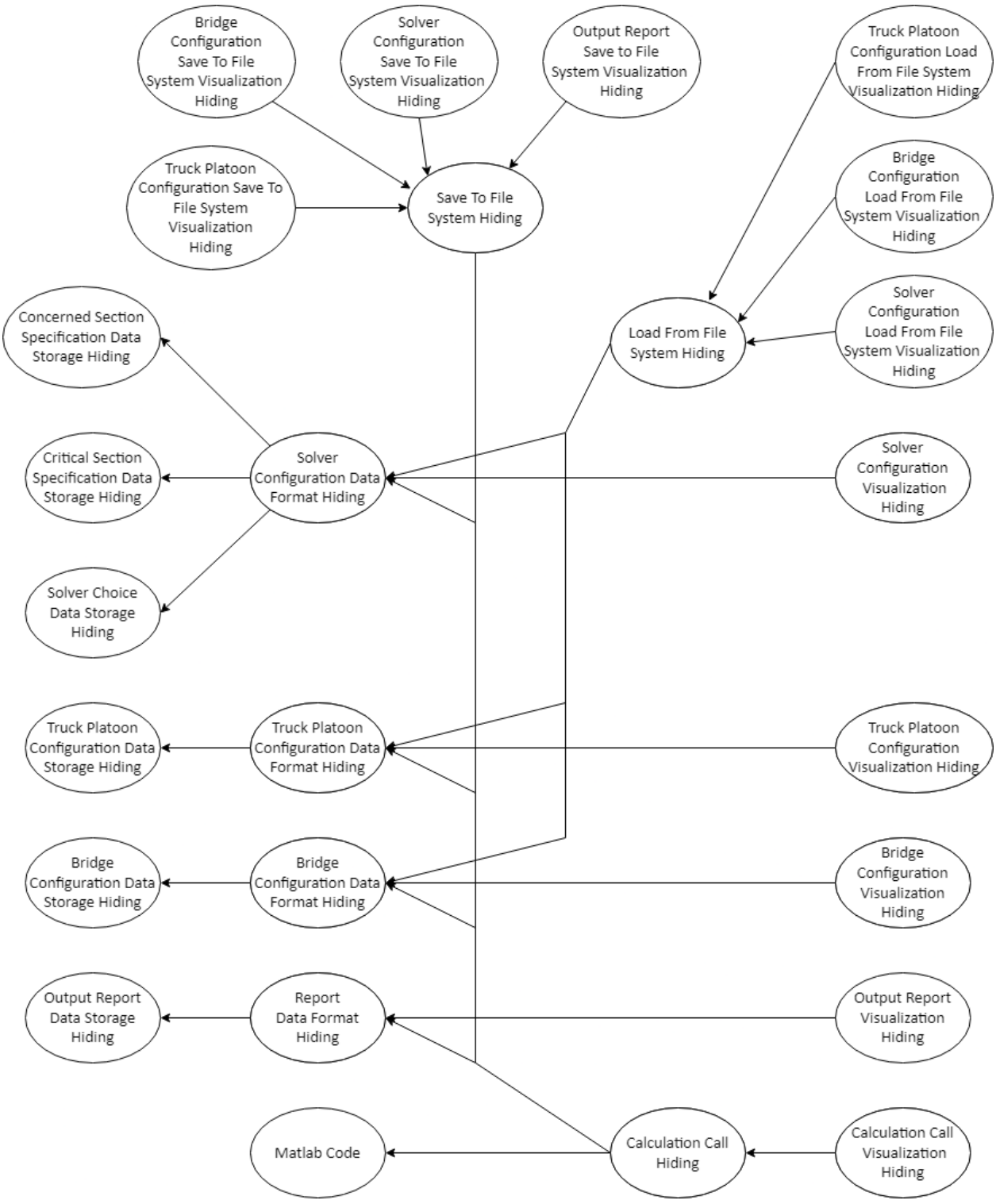

Figure 1: Use hierarchy among modules

References

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.