Module Guide for Stoichiometry Mass-Mass Program

Deemah Alomair

December 5, 2019

1 Revision History

Date	Version	Notes
Date 1	1.0	First version of document

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
SMMP	Stoichiometry Mass-Mass Program
UC	Unlikely Change
GUI	Graphical User Interface

Contents

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 **?**. 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 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, 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: The specific hardware on which the software is running.

AC2: The structure and format of the input data with respect to chemical equations.

AC3: The assumption that only two reactants are involved in the chemical reaction.

AC4: The assumption that maximum number of compound elements are two.

AC5: Underlying structure used to get molecular weight value

AC6: The calculation method.

AC7: The format of the final output data.

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: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

UC2: The goal statements of SMMP.

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.

M1: Hardware-Hiding Module

M2: Input Module

M3: Atomic Mass Module

M4: ReactionT Module

M5: Mass Calculation Module

M6: GUI Module

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Module Atomic Mass Module ReactionT Module Mass Calculation Module
Software Decision	GUI Module

Table 1: Module Hierarchy

[I do not see where your other types will be defined. There is a ReactionT, but what about the building block types to get there? —SS]

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.

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. *SMMP* means the module will be implemented by the SMMP 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 behaviors.

Services: Includes programs that provide externally visible behavior 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 format of the valid inputs.

Services: : Converts valid input data into data structure that can be processed by SMMP.

Implemented By: SMMP.

7.2.2 Atomic Mass Module(M3)

Secrets: The algorithm for getting the atomic mass for for each chemical element from the library.

Services: Provides access to the library that can by used by SMMP.

Implemented By: SMMP.

7.2.3 ReactionT Module(M4)

Secrets: Balancing chemical reaction calculation algorithm.

Services: Produces the balance chemical reaction from unbalanced form.

Implemented By: SMMP.

7.2.4 Mass calculation Module(M5)

Secrets: Mass calculation algorithm.

Services: Calculates the correct value of the mass from given inputs.

Implemented By: SMMP.

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 GUI Module(M6)

Secrets: How to display the result to end user.

Services: Converts the result to understandable output.

Implemented By: SMMP.

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	M2
R2	M6
R3	M3 ,M4 , M5
R4	M2, M3 ,M4 , M5

Table 2: Trace Between Requirements and Modules

[Why isn't M1 used for anything? —SS]

AC	Modules
AC1	M1
AC2	M2
AC3	M4
AC4	M4
AC5	M3
AC6	M5
AC7	M6

Table 3: Trace Between Anticipated Changes and Modules

[One module (M4) tied to two anticipated changes is not a good sign. This should either be changed, or explicitly addressed in your write-up. —SS]

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. **?** 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.

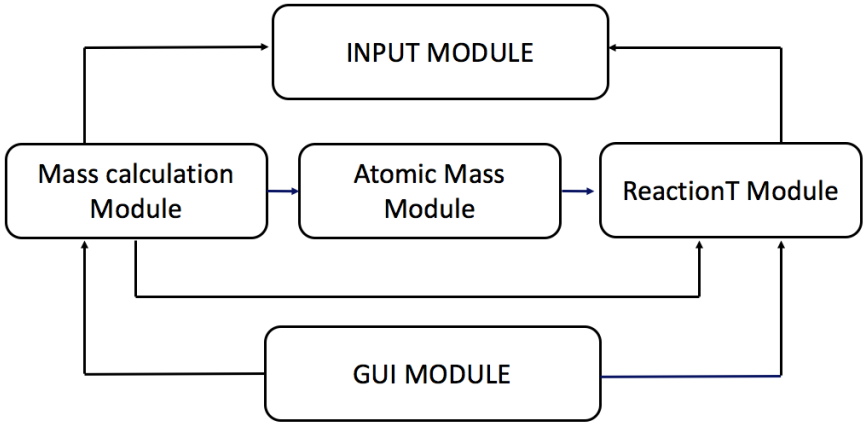

Figure 1: Use hierarchy among modules

References

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.

[I fixed one of the references. Please fix the others. The original file in the Blank Project Template should have worked with no modifications. —SS]