# System Verification and Validation Plan for Stoichiometry Mass-Mass Program

Deemah Alomair

November 20, 2019

# 1 Revision History

| Date | Version | Notes |
|---|---|---|
| 28/10/2019 | 1.0 | First version of document |

# Contents

# 2 Symbols, Abbreviations and Acronyms

| symbol | description |
|---|---|
| T | Test |
| SRS | Software Requirements Specification |
| SMMP | Stoichiometry Mass-Mass Program |
| R | Functional requirement |
| NF | Non-functional requirement |

# 3 General Information

This document is to build verification and validation plan for Stoichiometry Mass-Mass Program and provide description about the testing that will be carried out on the system. The main goal of this document is to check whether SMMP meets the SRS and fulfill its intended purpose. This document will be used as the reference and guidance for testing SMMP.

## 3.1 Summary

Stoichiometry Mass-Mass Program is a software that convert unbalanced chemical reaction to balanced one. Then do some necessary calculations to get the mass of one of the reactants involved in that chemical reaction.

## 3.2 Objectives

The objectives of the verification and validation plan is to ensure that SMMP is Reliable. In most cases it provides correct answer and behave as intended. In addition, usability because this software may be used by chemistry student who has less experiences in dealing with computer software. Maintainability is also need to be achieved because this is the first draft of the software and continues changes may be done in next versions.

## 3.3 Relevant Documentation

Relevant Documents that need to be visited while reading this document include the following:

SRS, which can be found here

[Please list all of the documents in your project, even the ones that haven't been written yet. :-) —SS]

# 4 Plan

[There should generally be text between all section headings. If there is no text, then you can provide a "roadmap" of the section. —SS]

## 4.1 Verification and Validation Team

The test team has one member: Deemah Alomair

[You should also list your classmates and the course instructor. —SS]

## 4.2 SRS Verification Plan

SRS Verification Plan will include a feedback from the domain expert, secondary reviewer and Dr. Smith.

[I'm looking for more detail here and for the design verification. Peter has a good start on integrating the plan with the course structure, including using our checklists. —SS]

## 4.3 Design Verification Plan

Design Verification Plan will include a feedback from the domain expert, secondary reviewer and Dr. Smith.

## 4.4 Implementation Verification Plan

Implementation Verification Plan will include the followings:

- Dynamic and automatic testing that includes unit testing for each individual function and coverage testing. [What specific tools? —SS]
- Linters will be considered throughout the implementation of SMMP. [What specific tools —SS]
- Parallel testing.
- Code walkthrough for SMMP will be carried out after the implementation by the domain expert and Dr. Smith. [Code walkthroughs are a full technique. You should give details, along with a reference. —SS]

## 4.5 Software Validation Plan

N.A [I agree, but you should explain to the reader why this is N.A. —SS]

# 5 System Test Description

This section lists the system tests to be performed to verify whether or not the program fulfills the functional requirements and to test how well it meets the non-functional requirements. Note that some of the tests for the functional requirements are unit tests.

## 5.1 Tests for Functional Requirements

### 5.1.1 Input Tests

**Input constrains tests**

1. **T1: Mass value test**

   Control: Functional, dynamic, automatic.

   Initial State: Not Applicable.

   Input: -7. [You should list all of the system level inputs. Otherwise, the tester will not know what values to enter. To prevent this document from getting too large, you can use a base case test with your innocuous inputs, and then use a table to show the delta from this base test case to give your erroneous input test cases. —SS]

   Output: Error message indicates that entered mass value should be greater than 0.

   Test Case Derivation: The expected mass value is greater than 0.

   How test will be performed: Automatic unit testing.

2. **T2: Mass format test**

   Control: Functional, dynamic, automatic.

   Initial State: Not Applicable.

   Input: “H”.

   Output: Error message indicates that entered mass value should be numeric.

   Test Case Derivation: The expected entered mass value is a positive number.

How test will be performed: Automatic unit testing.

3. **T3: Reactant/product format test**

   Control: Functional, dynamic, automatic.

   Initial State: Not Applicable.

   Input: "H2".

   Output: Error message indicates that entered Reactant value should be combination of coefficient + chemical element. [This error should be caught by a type error. —SS]

   Test Case Derivation: The expected entered reactant value is positive coefficient + chemical element like : "2H" , "$2H_2O$"

   How test will be performed: Automatic unit testing.

4. **T4: Chemical reaction format test**

   Control: Functional, dynamic, automatic.

   Initial State: Not Applicable.

   Input: "$H_2O = 2$" . [Your inputs imply that you are planning on doing string parsing for your chemical reaction inputs. I think this might be more work than you are prepared to do. As we discussed, I think you can use a GUI with drop down menus and edit boxes to enter the parts of the equation. You can avoid parsing that way. —SS]

   Output: Error message indicates that this is not a correct chemical reaction format.

   Test Case Derivation: The expected entered chemical reaction is set of reactants and products like: $NaOH + H_2SO_4 \rightarrow H_2O + NA_2SO_4$

   How test will be performed: Automatic unit testing.

### 5.1.2 Output Tests

**Output constrains tests**

1. **T5: Result test**

Control: Functional, dynamic, automatic.

Initial State: Not Applicable.

Input: chemical reaction : "$NaOH + H_2SO_4 \rightarrow H_2O + NA_2SO_4$"
mass of known reactant : "3.10"
name of known reactant: "$H_2SO_4$"

Output: mass = 2.53 g.

Test Case Derivation: The expected value of the mass or error message otherwise.

How test will be performed: coverage testing, parallel testing. [This is what kind of test it is, not how it will be performed. Parallel with what? —SS]

2. **T6: Chemical reaction balancing test**

Type: Functional, dynamic, automatic.

Initial State: Not Applicable.

Input: $N_2 + 3H_2 \rightarrow NH_3$

Output: $N_2 + 3H_2 \rightarrow 2NH_3$ .

Test Case Derivation: The expected balance equation or error message otherwise.

How test will be performed: Parallel testing.

[You do not have nearly enough test cases with the actual functionality of your software. You need multiple test cases to balance an equation and you need multiple test cases to determine the unknown masses. —SS]

## 5.2 Tests for Nonfunctional Requirements

### 5.2.1 Usability

1. **T7: Usability testing**

Type: Nonfunctional,Dynamic, Manual.

Initial State: Not Applicable.

Input/Condition: Unbalanced chemical reaction/ mass of one reactant.

Output/Result: Second reactant mass.

How test will be performed: Ask domain expert to use the system then get a feedback of the ease of using the system. [You need more detail here. This is not rigorous. You should think from the perspective of a third party that you give this document and ask to complete the tests. That third party needs enough information to complete the tests. —SS]

### 5.2.2 Reliabilty [spell check —SS]

1. **T8: Reliability testing**

Type: Nonfunctional, Dynamic, Manual.

Initial State: Not Applicable.

Input/Condition: Unbalanced chemical reaction/ mass of one reactant.

Output/Result: Second reactant mass.

How test will be performed: using the system with different inputs and measuring the percentage of correct answers. [more detail - what inputs?, what outputs? how are the percentages calculated? —SS]

### 5.2.3 Maintainability

1. **T9: Maintainability testing**

Type: Nonfunctional,Dynamic, Manual.

Initial State: Not Applicable.

Input/Condition: Existing SMMP system.

Output/Result: New version of SMMP.

How test will be performed: Try to add new feature or fix some constrains [proof read —SS] and observe system reaction. [How is this going to be measured? What feature is being added? Who is adding it? —SS]

## 5.3 Traceability Between Test Cases and Requirements

A trace between system tests and requirements is provided in Table 1.

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 |
|---|---|---|---|---|---|---|---|---|---|
| R1 | X | X | X | X | | | | | |
| R2 | | | | | X | X | | | |
| R3 | | | | | X | X | | | |
| R4 | X | X | X | X | X | X | | | |
| NF1 | | | | | X | X | | X | |
| NF2 | X | X | X | X | X | X | | | |
| NF3 | | | | | | | X | | |
| NF4 | | | | | | | | | X |

Table 1: Traceability Matrix Showing the Connections Between Requirements and system tests