

# System Verification and Validation Plan for ChemCode

Samuel Crawford

February 19, 2023

# 1 Revision History

Date	Version	Notes
Feb. 5, 2023	0.0	Create document and remove inapplicable content
Feb. 7-8, 2023	0.1	Add input tests
Feb. 8, 2023	0.1.1	Improve referencing of tests
	0.1.2	Add matrix conversion tests and improve input tests, including rationale, labelling, and chemical equations
Feb. 9, 2023	0.1.3	Add tests for trivial equation
	0.1.4	Add feasibility tests
Feb. 13, 2023	0.1.5	Clarify notion of matrices having the same solution after swapping rows and/or columns
Feb. 13-14, 2023	0.1.6	Add balancing and output tests
Feb. 14, 2023	0.1.7	Add accuracy test for balancing
Feb. 16, 2023	0.1.8	Move (reordered) “system tests” to unit test section (commented out), convert them to true system tests, and tweak balancing accuracy test appropriately
Feb. 18, 2023	0.1.9	Add static test for element support
	0.1.10	Improve formatting of existing tests, including adding references to requirements
Feb. 19, 2023	0.1.11	Add remaining tests for nonfunctional requirements (except for NFR4 <a href="#">[add link — SC]</a> )
	0.2.0	Start filling in <a href="#">Plan</a> section

# Contents

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>iv</b>
<b>3</b>	<b>General Information</b>	<b>1</b>
3.1	Summary . . . . .	1
3.2	Objectives . . . . .	1
3.3	Relevant Documentation . . . . .	1
<b>4</b>	<b>Plan</b>	<b>1</b>
4.1	Verification and Validation Team . . . . .	1
4.2	SRS Verification Plan . . . . .	2
4.3	Design Verification Plan . . . . .	3
4.4	Verification and Validation Plan Verification Plan . . . . .	3
4.5	Implementation Verification Plan . . . . .	3
4.6	Automated Testing and Verification Tools . . . . .	3
4.7	Software Validation Plan . . . . .	4
<b>5</b>	<b>System Test Description</b>	<b>4</b>
5.1	Tests for Functional Requirements . . . . .	4
5.1.1	Element Support Testing . . . . .	4
5.1.2	Feasible Reaction Testing . . . . .	5
5.1.3	Infeasible Reaction Testing . . . . .	6
5.2	Tests for Nonfunctional Requirements . . . . .	6
5.2.1	Accuracy Testing . . . . .	7
5.2.2	Understandability Testing . . . . .	7
5.2.3	Usability Testing . . . . .	7
5.2.4	Portability Testing . . . . .	7
5.2.5	Verifiability Testing . . . . .	8
5.3	Traceability Between Test Cases and Requirements . . . . .	8
<b>6</b>	<b>Unit Test Description</b>	<b>8</b>
6.1	Unit Testing Scope . . . . .	8
6.2	Tests for Functional Requirements . . . . .	9
6.2.1	Module 1 . . . . .	9
6.2.2	Module 2 . . . . .	10
6.3	Tests for Nonfunctional Requirements . . . . .	10

6.3.1	Module ? . . . . .	10
6.3.2	Module ? . . . . .	10
6.4	Traceability Between Test Cases and Modules . . . . .	11
<b>7</b>	<b>Appendix</b>	<b>12</b>
7.1	Symbolic Parameters . . . . .	12
7.2	Usability Survey . . . . .	12

## List of Tables

1	Table of Teammates and Their Roles . . . . .	2
---	--	---

## List of Figures

[Remove this section if it isn't needed —SS]

## 2 Symbols, Abbreviations and Acronyms

symbol	description
CAS	Computing and Software
SUS	System Usability Scale
T	Test

[symbols, abbreviations or acronyms — you can simply reference the SRS  
[1] tables, if appropriate —SS]  
[Remove this section if it isn't needed —SS]

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

## 3 General Information

### 3.1 Summary

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

### 3.2 Objectives

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: “build confidence in the software correctness,” “demonstrate adequate usability.” etc. You won’t list all of the qualities, just those that are most important. —SS]

### 3.3 Relevant Documentation

[Reference relevant documentation. This will definitely include your SRS and your other project documents (design documents, like MG, MIS, etc). You can include these even before they are written, since by the time the project is done, they will be written. —SS]

[1]

## 4 Plan

This section outlines the plan for verifying and validating ChemCode, which will be performed by a team. This plan includes verifying the SRS, design, VnV Plan, and implementation. It also includes the user of automated tools for testing and verification and external data for validation.

### 4.1 Verification and Validation Team

Each teammate has its role listed in [Table of Teammates and Their Roles](#) according to the following roles:

- **Author:** The person writing the documentation and Drasil implementation of ChemCode.
- **Project Supervisor:** The person in charge of the ChemCode project, including its implementation in Drasil.
- **Reviewer:** A person in charge of ensuring that the documentation and/or implementation of ChemCode meets the needs of the CAS 741 course (this may be limited to one artifact).
- **Validator:** A person in charge of ensuring that the documentation and/or implementation of ChemCode meets the goals of its implementation in Drasil.
- **Verifier:** A person in charge of ensuring that the implementation of ChemCode meets the requirements from its SRS.

Name	Role
Dr. Spencer Smith	Project Supervisor, Reviewer, Validator
Samuel Crawford	Author, Reviewer, Validator, Verifier
Jason Balaci	Reviewer, Validator
Deesha Patel	SRS Reviewer
Maryam Valian	VnV Plan Reviewer
Karen Wang	Drasil Implementation Reviewer
Drasil Team	Validators, Verifiers
Class of CAS 741	Reviewers

Table 1: Table of Teammates and Their Roles

## 4.2 SRS Verification Plan

The first round of review for the SRS will be done on its manual version. It will be reviewed first by Samuel Crawford during the writing process, then generally by the class of CAS 741, and then more rigorously by Jason Balaci, Deesha Patel, and Dr. Spencer Smith. After appropriate revisions and its implementation in Drasil, the generated version will then be reviewed by

Samuel Crawford, Dr. Spencer Smith and a subset of the Drasil team. Each review will check it against the general writing checklist, the SRS checklist and the SRS rubric for CAS 741 [\[add links —SC\]](#); the only exception is that the Drasil team will only verify the SRS within the scope of the Drasil project (i.e., not against the CAS 741 rubric).

### 4.3 Design Verification Plan

The design of ChemCode will be dictated by Drasil’s code generation. As Drasil does not currently generate design documentation [\[add source? —SC\]](#), this verification will be conducted based on the generated code itself. Verifying its design will be done by the Drasil team according to their process of design verification and is therefore out of the scope of this document.

### 4.4 Verification and Validation Plan Verification Plan

The VnV Plan will be reviewed first by Samuel Crawford during the writing process, then generally by the class of CAS 741, and then more rigorously by Jason Balaci, Maryam Valian, and Dr. Spencer Smith. After appropriate revisions, it will then be reviewed again by Samuel Crawford and Dr. Spencer Smith. Each review will check it against the general writing checklist and the VnV rubric for CAS 741 [\[add links —SC\]](#).

### 4.5 Implementation Verification Plan

The primary method of verifying the implementation of ChemCode will be performing the system and unit tests from Sections 5 and 6, respectively. Since the implementation will be generated by Drasil, a subset of the Drasil team will also inspect the code to ensure that it matches the expectations for generated code, especially focusing on the new code generation for solving linear systems. [\[Does the Drasil team already perform any formal static analysis? —SC\]](#) The implementation will also likely be reviewed by Drasil team members in the future, as related to future work on Drasil.

### 4.6 Automated Testing and Verification Tools

Since the initial implementation of ChemCode will be in Python, pytest will be used to automate testing and measure coverage where appropriate. Other



frameworks for automated testing will be added if/when code for ChemCode is generated in more languages. The Drasil repository has continuous integration set up to ensure that lint standards are adhered to and that the code compiles [\[double check —SC\]](#). Since the code for ChemCode will be generated by Drasil, the Drasil team may potentially use other tools as part of its verification (e.g., linters). [\[Are Makefiles generated? —SC\]](#)

## 4.7 Software Validation Plan

[\[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS\]](#)

[\[You might want to use review sessions with the stakeholder to check that the requirements document captures the right requirements. Maybe task based inspection? —SS\]](#)

[\[This section might reference back to the SRS verification section. —SS\]](#)

# 5 System Test Description

## 5.1 Tests for Functional Requirements

The tests in each section are given in order of increasing complexity/likelihood of the situation arising during use of ChemCode.

### 5.1.1 Element Support Testing

#### T1: Test for Element Support

Test Case Derivation: In order for the user to be able to work with any possible chemical reaction, they must be able to enter each of the currently known 118 elements.

How test will be performed: Static analysis will be performed against the list of elements from [\[2\]](#), ensuring that the symbol of every element can be used throughout the execution of ChemCode. This includes the input stage (from R1 from SRS [\[add link —SC\]](#)), the conversion stage (from R2 from SRS [\[add link —SC\]](#)), and the output stage (from R4 and R6 from SRS [\[add link —SC\]](#)).

### 5.1.2 Feasible Reaction Testing

The following tests are for equations of feasible chemical reactions and will be performed automatically. Providing the correct output for an inputted feasible chemical reaction satisfies R1, R2, R3, R5, and R6 from SRS [\[add link —SC\]](#). Note that the input/output format for each test is an abstract representation, as the specific format of each input/output is a design decision that is not made at this stage.

#### T2: Test for Small Valid Equation

Input:  $\text{O}_2 \rightarrow \text{O}_3$  [\[3\]](#)

Output:  $3 \text{O}_2 \rightarrow 2 \text{O}_3$  [\[3, p. 6\]](#)

Test Case Derivation: The inputted chemical equation is valid and trivial.

#### T3: Test for Valid Equation

Input:  $\text{C}_2\text{H}_6 + \text{O}_2 \rightarrow \text{CO}_2 + \text{H}_2\text{O}$  [\[4\]](#)

Output:  $2 \text{C}_2\text{H}_6 + 7 \text{O}_2 \rightarrow 4 \text{CO}_2 + 6 \text{H}_2\text{O}$  [\[4, p. 523\]](#)

Test Case Derivation: The inputted chemical equation is valid and relatively small, but larger than the trivial one from T2.

#### T4: Test for Large Valid Equation

Input:  $\text{KMnO}_4 + \text{HCl} \rightarrow \text{MnCl}_2 + \text{KCl} + \text{Cl}_2 + \text{H}_2\text{O}$  [\[5\]](#)

Output:  $2 \text{KMnO}_4 + 16 \text{HCl} \rightarrow 2 \text{MnCl}_2 + 2 \text{KCl} + 5 \text{Cl}_2 + 8 \text{H}_2\text{O}$  [\[5\]](#)

Test Case Derivation: The inputted chemical equation is valid and larger than the one from T3.

#### T5: Test for Valid Equation with Nonstoichiometric Compound

Input:  $\text{Fe}_{0.95}\text{O} + \text{O}_2 \rightarrow \text{Fe}_2\text{O}_3$  [\[6\]](#)

Output:  $80 \text{Fe}_{0.95}\text{O} + 17 \text{O}_2 \rightarrow 38 \text{Fe}_2\text{O}_3$

Test Case Derivation: The inputted chemical equation contains a non-stoichiometric compound (i.e., one with a fractional subscript).

### 5.1.3 Infeasible Reaction Testing

The following tests are for equations of infeasible chemical reactions and will be performed automatically. Providing the correct output for an inputted infeasible chemical reaction satisfies R1, R2, R3, and R4 from SRS [\[add link —SC\]](#). Note that the input/output format for each test is an abstract representation, as the specific format of each input/output is a design decision that is not made at this stage.

#### T6: Test for Valid Equation that is Infeasible due to Conservation of Mass Violation

Input:  $\text{C}_2\text{H}_6 \rightarrow \text{CO}_2 + \text{H}_2\text{O}$

Output: “This reaction is infeasible because O is present on one side of the equation but not the other, violating the Law of Conservation of Mass.”

Test Case Derivation: The inputted chemical equation is infeasible since every element does not exist on both sides of the equation, which violates the Law of Conservation of Mass (TM1 from SRS [\[add link —SC\]](#)).

#### T7: Test for Valid Equation that is Infeasible due to Overconstrained System

Input:  $\text{K}_4\text{FeC}_6\text{N}_6 + \text{K}_2\text{S}_2\text{O}_3 \rightarrow \text{CO}_2 + \text{K}_2\text{SO}_4 + \text{NO}_2 + \text{FeS}$  (modified from [\[4\]](#))

Output: “This reaction is infeasible because it is overconstrained.”

Test Case Derivation: The inputted chemical equation is infeasible since each compound has more than one element, so changing any coefficient affects the number of some other element, causing a chain reaction that does not converge. There is no solution to this system other than the trivial solution (**0**) [\[4\]](#).

## 5.2 Tests for Nonfunctional Requirements

There are no tests for maintainability because this is the responsibility of the Drasil team, as Drasil is responsible for generated the actual implementation of ChemCode.

### 5.2.1 Accuracy Testing

#### T8: Test for Accuracy of Balancing

Type: Dynamic, Automatic

Test Case Derivation: Chemical equations are only useful if they are balanced [7], so computed coefficients should be exact. Since these coefficients should be the smallest possible whole numbers [7], there is exactly one possible set of coefficients for each feasible chemical equation and exact equality between this expected output and the actual output can be checked by a computer (e.g., by using integer comparison).

How test will be performed: This verification will be performed by the [Tests for Functional Requirements](#) using integer comparison and satisfies R1, R2, R3, R5, R6, and NFR1 from SRS [\[add link —SC\]](#).

### 5.2.2 Understandability Testing

#### T9: Test for Understandability of ChemCode

Type: Dynamic, Manual

How test will be performed: New typical users will be asked to input a valid chemical reaction to be balanced by the system. At least USER.FRAC of users should be able to do so within FIRST\_USE.TIME.

### 5.2.3 Usability Testing

#### T10: Test for Usability of ChemCode

Type: Dynamic, Manual

How test will be performed: Typical users will be told how ChemCode works and then instructed to input the chemical equations from the [Tests for Functional Requirements](#) to be balanced. They will then be asked the questions from the System Usability Scale (SUS) [8] (see [Usability Survey](#)). The average score should be at least SUS.SCORE.

### 5.2.4 Portability Testing

#### T11: Test for Portability of ChemCode

Type: Dynamic

Test Case Derivation: ChemCode may be used by a large variety of users who may have different systems.

How test will be performed: This verification will be performed by ensuring the correct programming language(s) is/are installed on both a Windows and macOS system and the [Tests for Functional Requirements](#) will be performed.

### 5.2.5 Verifiability Testing

#### T12: Test for Verifiability of ChemCode

Type: Dynamic, Static

How test will be performed: All other tests from this plan will be performed successfully.

## 5.3 Traceability Between Test Cases and Requirements

[\[Provide a table that shows which test cases are supporting which requirements. —SS\]](#)

# 6 Unit Test Description

[\[Reference your MIS \(detailed design document\) and explain your overall philosophy for test case selection. —SS\] \[This section should not be filled in until after the MIS \(detailed design document\) has been completed. —SS\]](#)

## 6.1 Unit Testing Scope

[\[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS\]](#)

## 6.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

### 6.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

#### 1. test-id1

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

#### 2. test-id2

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

#### 3. ...

### 6.2.2 Module 2

...

## 6.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

### 6.3.1 Module ?

1. test-id1

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

Type: Functional, Dynamic, Manual, Static etc.

Initial State:

Input:

Output:

How test will be performed:

### 6.3.2 Module ?

...

## 6.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

## References

- [1] A. Author, “System requirements specification.” <https://github.com/...>, 2019.
- [2] A. M. Helmenstine, “A List of the Elements of the Periodic Table,” May 2020.
- [3] D. W. Fahey and M. I. C. L. A. Hegglin, “Twenty Questions and Answers About the Ozone Layer: 2010 Update, Scientific Assessment of Ozone Depletion: 2010,” tech. rep., World Meteorological Organization, Geneva, Switzerland, Mar. 2011.
- [4] I. Hamid, “Balancing Chemical Equations by Systems of Linear Equations,” *Applied Mathematics*, vol. 10, pp. 521–526, July 2019.
- [5] S. Taylor, “Balancing Complex Chemical Equations with Chemical Equations Examples,” June 2021.
- [6] Doubtnut, “When non stoichiometric compound  $\text{Fe}_{0.95}\text{O}$  is heated in presence of oxygen then it converts into  $\text{Fe}_2\text{O}_3$ . Which of the following statement is correct?.”
- [7] L. Lund and Anoka-Ramsey Community College, *Introduction to Chemistry*. Cambridge and Coon Rapids, MN, USA: LibreTexts, Jan. 2023.
- [8] N. Thomas, “How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website,” July 2015. Section: Terminology.



## 7 Appendix

### 7.1 Symbolic Parameters

Symbolic constants used in test cases are defined here for easy maintenance.

- `FIRST_USE_TIME` = 10 minutes
- `SUS_SCORE` = 75 (note that “the average System Usability Scale score is 68” [8])
- `USER_FRAC` = 80%

### 7.2 Usability Survey

This survey was taken from [8]. The user will be asked to answer the following questions on a scale of one to five, where one means “Strongly Disagree” and five means “Strongly Agree.” The scores for all odd-numbered questions will be decremented by one, the even-numbered question scores will be subtracted from five, and the sum of these new values will be multiplied by 2.5 to yield the final score.

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.