CAS 741: Test Plan

Aqueous Speciation Diagram Generator

Steven Palmer
palmes4

October 25, 2017

Revision History

Table 1: Revision History

Date	**Developer(s)**	**Change**
10.13.2017	S. Palmer	First revision of document
10.25.2017	S. Palmer	Revision 0 submission

Symbols, Abbreviations and Acronyms

symbol	description
SpecGen	The Aqueous Speciation Diagram Generator program
SRS	Software Requirements Specification
T	Test

Contents

1 General Information

This document provides a detailed description of the testing that will be carried out on the Aqueous Speciation Diagram Generator program (herein referred to as SpecGen).

1.1 Purpose

The purpose of this document is to provide a comprehensive plan for testing the SpecGen software against the requirements described in the SpecGen SRS.

1.2 Scope

The test plan is narrowed to the following scope:

- The tests outlined in this document are limited to the verification of the requirements given in the SpecGen SRS. The validation of the requirements will be carried out via correspondance with Dr. Scott Smith.
- The tests outlined in this document are limited to dynamic tests only. Due to the small size and low complexity of the SpecGen program, no formal static testing (code walkthroughs, code inspections, etc.) will be carried out.
- The SpecGen software will be written in Python. The testing of implementations in other languages will not be considered in this document.

2 Plan

2.1 Software Description

Chemical speciation refers to the stable (equilibrium) distribution of chemical species in a given chemical system. Speciation diagrams, which plot species concentrations against an independently varied parameter of the system, are useful tools for displaying speciation data in a concise and easy to use format.

SpecGen will produce a speciation diagram given a set of chemical reactions, equilibrium constants, and element totals that define a chemical system. SpecGen will be specific to speciation of ions in aqueous systems under varying pH, which is of particular importance in the fields of aqueous process engineering and hydrometallurgy. The diagram generated by SpecGen will plot speciation of all aqueous species (excluding H^+ and OH^-) across the pH range 0 to 14.

2.2 Test Team

The test team includes the following members:

- Steven Palmer

2.3 Automated Testing Approach

The automated testing for SpecGen will be carried out using a set of unit and integrations tests. A test coverage analysis of these tests will be carried out to ensure that testing is as complete as possible. The target for this analysis is 100% statement coverage.

Regression testing will be used during the implementation stage and for any future changes. Since SpecGen is small in scope and will be implemented by a single developer, other forms of automated testing, such as continuous integration testing, will not be considered.

2.4 Verification Tools

The following tools will be used to facilitate testing:

1. **PyUnit** (a unit testing framework for Python) will be used to write and run unit tests
2. **Coverage.py** will be used to assess test coverage
3. **make** will be used to automate the building and execution of the test program

2.5 Non-Testing Based Verification

N/A

3 System Test Description

[Intial states, inputs, outputs will be formalized/revised as implementation proceeds. —SP]

3.1 Tests for Functional Requirements

3.1.1 Input Testing

T1: Equation Parser, Integration

Type: Functional, Automatic, Integration

Initial State: N/A

Input: Input file with general chemical reaction
[TODO: change this to the actual input file text with implementation —SP]

Output: Data structure with input reaction equation components
[TODO: change this to the actual output structure when implemented —SP]

How test will be performed: Automated integration test

T2: Input Data Structure, Integration

Type: Functional, Automatic, Integration

Initial State: N/A

Input: Input file
[TODO: change this to the actual input file text with implementation —SP]

Output: Input parameters stored by the program match the input file
[TODO: change this to the actual output structure when implemented —SP]

How test will be performed: Automated integration test

3.1.2 Derived Input Testing

T3: Species List from Equation, Distinct Species in Equation

Type: Functional, Automatic, Integration

Initial State: N/A

Input: Input file with reaction:
$H_2CO_{3(aq)} \rightleftharpoons H^+ + HCO_3^-$
[TODO: change this to the actual input file text with implementation —SP]

Output: List of species [H_2O, H^+, OH^-, $H_2CO_{3(aq)}$, HCO_3^-]

How test will be performed: Automated integration test

T4: Species List from Equations, Repeated Species in Equations

Type: Functional, Automatic, Integration

Initial State: N/A

Input: Input file with reactions:
$H_2CO_{3(aq)} \rightleftharpoons H^+ + HCO_3^-$
$HCO_3^- \rightleftharpoons H^+ + CO_3^{2-}$
[TODO: change this to the actual input file text with implementation —SP]

Output: List of species [H_2O, H^+, OH^-, $H_2CO_{3(aq)}$, HCO_3^-, CO_3^{2-}]

How test will be performed: Automated integration test

3.1.3 Input Constraint Testing

T5: Element Total Concentration, Integration

Type: Functional, Automatic, Integration

Initial State: N/A

Input: Input file with Fe_{tot} = -1.0
[TODO: change this to the actual input file text with implementation —SP]

Output: Error

How test will be performed: Automated integration test

T6: Element Total Concentration, Zero, Integration

Type: Functional, Automatic, Integration

Initial State: N/A

Input: Input file with Fe_{tot} = 0.0
[TODO: change this to the actual input file text with implementation —SP]

Output: Error

How test will be performed: Automated integration test

T7: Element Total Concentration, Positive, Integration

Type: Functional, Automatic, Integration

Initial State: N/A

Input: Input file with Fe_{tot} = 1.0
[TODO: change this to the actual input file text with implementation —SP]

Output: Fe_{tot} = 1.0

How test will be performed: Automated integration test

3.1.4 Calculation Testing

T8: Non-Linear Equation System Solution, Integration

Type: Functional, Automatic, Parallel, Integration

Initial State: N/A

Input: Input file with same data as the MATLAB implementation
[TODO: change this to the actual input file text with implementation —SP]

Output: Solution that matches the MATLAB implementation
[TODO: get data from MATLAB implementation —SP]

How test will be performed: Automated integration test

T9: Non-Linear Equation System Solution, Integration

Type: Functional, Automatic, Integration

Initial State: N/A

Input: Input file with redundant equation:
$Fe^{3+} \rightleftharpoons Fe^{3+}$, K = 1.0
$Fe_{tot} = 0.1$
[TODO: change this to the actual input file text with implementation —SP]

Output: $[Fe^{3+}] = 0.1$ M at all pH

How test will be performed: Automated integration test

3.1.5 Output Testing

T10: Speciation Diagram, Integration

Type: Functional, Manual, Integration

Initial State: N/A

Input: Input file with:
$CO_{2(aq)} + H_2O \rightleftharpoons H_2CO_{3(aq)}$, $K = 1.7 \times 10^3$
$H_2CO_{3(aq)} \rightleftharpoons H^+ + HCO_3^-$, $K = 2.5 \times 10^4$
$HCO_3^- \rightleftharpoons H^+ + CO_3^{2-}$, $K = 4.69 \times 10^{11}$
$C_{tot} = 0.1$
[TODO: change this to the actual input file text with implementation —SP]

Output: Correct carbonate speciation diagram
[TODO: should I link to diagram here? —SP]

How test will be performed: The generated speciation diagram will be manually reviewed to ensure that it matches the diagram expected by the input specifications.

3.2 Tests for Nonfunctional Requirements

3.2.1 Output Readability Testing

T11: Readability of generated speciation diagram

Type: Nonfunctional, Manual

How test will be performed: This is a qualitative test to ensure that the diagrams generated by SpecGen are readable (axis labels visible, curves distinguishable from eachother, legend/labelling of curves, etc.).

4 Traceability Between System Tests and Requirements

A trace between system tests and requirements is provided in Table 2.

Table 2: Requirements Traceability

Requirement	Test(s)
R1	T1, T2
R2	T3, T4
R3	T5, T6, T7
R4	T8, T9
R5	T10
NF1	T11 [TODO: add NF1 to SRS —SP]

5 Unit Testing Plan

[Just a rough idea of unit tests – will update with implementation. —SP]

5.1 Input Module Testing

T12: Equation Parser, General

Type: Functional, Automatic, Unit

Initial State: ?

Input: String representation of chemical reaction

Output: Data structure of reaction equation components

How test will be performed: Automated unit test

T13: Equation Parser, Edge Case 1

Type: Functional, Automatic, Unit

Initial State: ?

Input: String representation of chemical equation that hits edges cases (want full coverage)

Output: Data structure of reaction equation components

How test will be performed: Automated unit test

[Edge cases will be added during implementation (after data structures and exact format of input is decided). For simplicity going forward, edge cases are ignored (since at this point I don't know the number/nature of the edge cases). They will be added as the implementation is developed. —SP]

T14: Input Data Structure

Type: Functional, Automatic, Unit

Initial State: ?

Input: Input data (?)

Output: Data structure works – probably several tests

How test will be performed: Automated unit test

5.2 Derived Input Module Testing

T15: Species List from Equation

Type: Functional, Automatic, Unit

Initial State: ?

Input: String representation of chemical reaction

Output: List of species contained in reaction

How test will be performed: Automated unit test

5.3 Input Constraints Module Testing

T16: Negative Element Total Concentration

Type: Functional, Automatic, Unit

Initial State: ?

Input: Negative concentration

Output: Error

How test will be performed: Automated unit test

T17: Zero Element Total Concentration

Type: Functional, Automatic, Unit

Initial State: ?

Input: Zero concentration

Output: Error

How test will be performed: Automated unit test

T18: Positive Element Total Concentration

Type: Functional, Automatic, Unit

Initial State: ?

Input: Positive concentration

Output: Positive concentration

How test will be performed: Automated unit test

5.3.1 Conversion Module Testing

[SpecGen will convert inputs into a set of non-linear equations to be solved by a non-linear solver. This will involve some sort of module/code to be tested here. —SP]

5.4 Calculation Module Testing

T19: Non-Linear Equation System Solution

Type: Functional, Automatic, Unit

Initial State: ?

Input: System of non-linear equations

Output: Solution that satisfies original system of equations

How test will be performed: Automated unit test

5.5 Output Module Testing

T20: Speciation Diagram

Type: Functional, Manual, Unit

Initial State: ?

Input: Output specifications

Output: Speciation diagram

How test will be performed: The generated speciation diagram will be manually reviewed to ensure that it matches the specified output.

6 Traceability Between Unit Tests and Modules

A trace between unit tests and modules is provided in Table 3.

Table 3: Module Traceability

Module	**Test(s)**
[TODO: fill in once implementation is done —SP]	