CAS 741: Test Plan

Aqueous Speciation Diagram Generator

Steven Palmer
palmes4

October 17, 2017

Revision History

Table 1: Revision History

Date	**Developer(s)**	**Change**
10.13.2017	S. Palmer	First revision of document

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. There is no explicit testing plan for the validation of the requirements.
- The tests outlined in this document are limited to dynamic tests only. Static tests (code review, syntax checking, etc.) will be carried out as the SpecGen software is developed, but are not explicitly listed/described.
- 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 majority of testing for SpecGen will be carried out using automated unit tests (both true unit tests as well as integration tests). Since SpecGen is small in scope and will be implemented by a single developer, other forms of automated testing (continuous integration, regression testing, etc.) will not be considered.

A test coverage analysis of the automated testing will also be carried out to ensure that testing is as complete as possible. The target for this analysis is 100% statement coverage. [Since SpecGen will be relatively small, an expectation of 100% coverage is not unreasonable? —SP]

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 as implementation proceeds. —SP]

3.1 Tests for Functional Requirements

3.1.1 Input Testing

Equation Parser Tests

T1: 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

T2: 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]

T3: 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

[Unit tests + integration tests? —SP]

T4: Equation Parser, Integration

Type: Functional, Automatic, Integration

Initial State: ?

Input: Input file with general chemical reaction

Output: Data structure of reaction equation components

How test will be performed: Automated integration test

T5: Input Data Structure, Integration

Type: Functional, Automatic, Integration

Initial State: ?

Input: Input file

Output: Data structure works – probably several tests

How test will be performed: Automated integration test

3.1.2 Derived Input Testing

Derived Species Tests

T6: 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

T7: Species List from Equation, Integration

Type: Functional, Automatic, Integration

Initial State: ?

Input: Input file

Output: List of species contained in reaction

How test will be performed: Automated integration test

3.1.3 Input Constraint Testing

Element Concentration Tests

T8: Negative Element Total Concentration

Type: Functional, Automatic, Unit

Initial State: ?

Input: Negative concentration

Output: Error

How test will be performed: Automated unit test

T9: Zero Element Total Concentration

Type: Functional, Automatic, Unit

Initial State: ?

Input: Zero concentration

Output: Error

How test will be performed: Automated unit test

T10: Positive Element Total Concentration

Type: Functional, Automatic, Unit

Initial State: ?

Input: Positive concentration

Output: Positive concentration

How test will be performed: Automated unit test

T11: Element Total Concentration, Integration

Type: Functional, Automatic, Integration

Initial State: ?

Input: Input file

Output: Concentration

How test will be performed: Automated integration test

3.1.4 Input Conversion 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, but none of this corresponds to an actual requirement. Should I have explicit tests for this even though it doesn't map to a requirement? Even without tests here, this will still end up being testing in the integration tests for calculations and output. —SP]

3.1.5 Calculation Testing

Non-Linear Equation Solver Tests

T12: 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

T13: Non-Linear Equation System Solution, Integration

Type: Functional, Automatic, Integration

Initial State: ?

Input: Input file

Output: Solution that satisfies original system of equations

How test will be performed: Automated integration test

3.1.6 Output Testing

Speciation Diagram Generation Tests

T14: 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.

T15: Speciation Diagram, Integration

Type: Functional, Manual, Integration

Initial State: ?

Input: Input file

Output: Speciation diagram

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 Tests

T16: Readability of generated speciation diagram

Type: Nonfunctional, Manual

Initial State: ?

Input: Input file

Output: Speciation diagram

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 Test Cases and Requirements

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

Table 2: Requirements Traceability

Requirement	**Test(s)**
R1	T1, T2, T3, T4, T5
R2	T6, T7
R3	T8, T9, T10, T11
R4	[I don't have any tests for this requirement. This requirement means I will have some hardcoded information available (probably in a data structure). I could make a silly test that my hardcoded data structure has the right data. Should this be something other than a requirement? —SP]
R5	T12, T13
R6	T14, T15

5 Unit Testing Plan

[Unit testing plans for internal functions and, if appropriate, output files —SS]

[I'm not sure what goes here? Unit tests are all described in the System Test Description section. —SP]