Project Title: Unit Verification and Validation Plan for GlassBR

Vajiheh Motamer

November 28, 2018

1 Revision History

Date	**Version**	**Notes**
27/11/18	1.0	initial UnitVnVPlan based on new template

Contents

List of Tables

[Do not include if not relevant —SS]

List of Figures

[Do not include if not relevant —SS]

2 Symbols, Abbreviations and Acronyms

symbol	**description**
T	Test

[symbols, abbreviations or acronyms – you can reference the SRS, MG or MIS tables if needed —SS]

This document ... [provide an introductory blurb and roadmap of the unit V&V plan —SS]

3 General Information

3.1 Purpose

[Identify software that is being unit tested (verified). —SS]

3.2 Scope

All introduced modules in MG are inside of the scope to UnitVnVPlan

4 Plan

4.1 Verification and Validation Team

Responsible member for the verification and validation of GlassBR is Vajiheh Motamer.

4.2 Automated Testing and Verification Tools

According to that all test files are going to develop based on Python. The following tools have been considered:

- Unit Testing Tools :
  - pytest: no API. Automatic collection of tests; simple asserts; strong support for test fixture/state management via setup/teardown hooks; strong debugging support via customized traceback. In additional, it is considered as tests runner which Selectivly run tests; Stop on first failure
  - unittest : Strong support for test organization and reuse via test suites

4.3 Non-Testing Based Verification

This section for GlassBRis not applicable.

5 Unit Test Description

[Reference your MIS and explain your overall philosophy for test case selection. —SS]

5.1 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]

5.1.1 Control Module

With reference to Section 4 from MIS, this section determines if the main program produces the correct output.

1. TstMain_1

   Type: Automotic

   Initial State: New Session

   Input: defaultInput.txt, output.txt
   from the following path: https://github.com/smiths/caseStudies/tree/master /CaseStudies/glass/src/Python/Test/Inputfiles

   Output: assertEqual(GenOutput.txt, output.txt) [Should I use sOut (Exported Constant in MIS) instead of GenOutput.txt? Besides, Should I use "GenOutput.txt, output.txt" for expected output instead of assert? —VM]

   Test Case Derivation: IM1 and T1 in SRS.

   How test will be performed: Unit testing using PyUnit.

2. TstMain_2

   Type: Automotic

   Initial State: New Session

   Input: TestInput1.txt, output1.txt
   https://github.com/smiths/caseStudies/tree/master /CaseStudies/glass/src/Python/Test/Inputfiles

   Output: assertEqual(GenOutput1.txt, output1.txt) [Should I consider only a test case which consists of a number of inputfile and outputfile instead of seperate test cases with same body and different input and output? —VM]

   Test Case Derivation: IM1 and T1 in SRS.

   How test will be performed: Unit testing using PyUnit.

5.1.2 Input Module

1. TstCheckConstraints
   Invalid input is input that defies the data constraints described in Section 6.2.4 of the SRS. This test case has considered to test verify params routin from MIS
   Type: Automotic

   Initial State: New Session

   Input: Table 8 and Table 1 from SystVnVPlan

   Output: assertEqual(("Specified Error in the Table 8 "), "Generated Error by GlassBR") [What should I consider instead of "generated error by glassbr"? —VM]

   Test Case Derivation: R1 and R2 from SRS.

   How test will be performed: Unit testing using PyUnit.

2. testInputFormat
   The following set of test cases is intended to ensure data is being read in from the input file correctly and it has considered to test of load_params(s) routin from MIS

   Type: Automotic

   Initial State: New Session

   Input: defaultInput.txt,testInput1.txt,testInput2.txt
   https://github.com/smiths/caseStudies/tree/master /CaseStudies/glass/src/Python/Test/Inputfiles

   Output: assertEqual(Params as maual(for example for length 1200), readed param from the input file) [Do you have better suggestion for the format of arguments of assertEqual? —VM]

   Test Case Derivation: R1 and R2 from SRS.

   How test will be performed: Unit testing using PyUnit.

3. testDerivedValues
   The following set of test cases is intended to ensure value from the derived quantities has been calculated correctly.

   Type: Automotic

   Initial State: New Session

   Input: defaultInput.txt,testInput1.txt,testInput2.txt
   https://github.com/smiths/caseStudies/tree/master /CaseStudies/glass/src/Python/Test/Inputfiles

   Output: assertEqual(Params as manual(for example for sdExpected 11.10180165558726), readed sd from the input file) [Should I have a sperate table for input and outputs similar to SystVnVPlan?? —VM]

   Test Case Derivation: R2 from SRS.

   How test will be performed: Unit testing using PyUnit.

5.1.3 Input Module

1. TstCheckConstraints
   Invalid input is input that defies the data constraints described in Section 6.2.4 of the SRS. This test case has considered to test verify params routin from MIS
   Type: Automotic

   Initial State: New Session

   Input: Table 8 and Table 1 from SystVnVPlan

   Output: assertEqual((”Specified Error in the Table 8 ”), ”Generated Error by GlassBR”) [What should I consider instead of ”generated error by glassbr”? —VM]

   Test Case Derivation: R1 and R2 from SRS.

   How test will be performed: Unit testing using PyUnit.

2. testInputFormat
   The following set of test cases is intended to ensure data is being read in from the input file correctly and it has considered to test of load_params(s) routin from MIS

Type: Automotic

Initial State: New Session

Input: defaultInput.txt,testInput1.txt,testInput2.txt
https://github.com/smiths/caseStudies/tree/master /CaseStudies/glass/src/Python/Test/Inputfiles

Output: assertEqual(Params as maual(for example for length 1200), readed param from the input file) [Do you have better suggestion for the format of arguments of assertEqual? —VM]

Test Case Derivation: R1 and R2 from SRS.

How test will be performed: Unit testing using PyUnit.

3. testDerivedValues
The following set of test cases is intended to ensure value from the derived quantities has been calculated correctly.

   Type: Automotic

   Initial State: New Session

   Input: defaultInput.txt,testInput1.txt,testInput2.txt
   https://github.com/smiths/caseStudies/tree/master /CaseStudies/glass/src/Python/Test/Inputfiles

   Output: assertEqual(Params as manual(for example for sdExpected 11.10180165558726), readed sd from the input file) [Should I have a sperate table for input and outputs similar to SystVnVPlan?? —VM]

   Test Case Derivation: R2 from SRS.

   How test will be performed: Unit testing using PyUnit.

   [I think this test case covers GlassTypeADT Module and ThicknessADT Module and Constants Module. Do you agree? —VM]

5.1.4 Calc Module

1. testCalculations
These set of tests are same to TC1 to TC7 in SystVnVPlan.

   [Is that enough to reference to SystVnVplan? —VM]

5.2 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]

5.2.1 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:

5.2.2 Module ?

...

5.3 Traceability Between Test Cases and Modules

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

6 References

7 Appendix

[This is where you can place additional information, as appropriate —SS]

7.1 Symbolic Parameters

[The definition of the test cases may call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. —SS]