CFS: System Verification and Validation Plan

Malavika Srinivasan

October 17, 2018

1 Revision History

Date	Version	Notes
Oct 16, 2018	1.0	First draft by Malavika

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

Also see the table of symbols in CA at: https://github.com/Malavika-Srinivasan/CAS741/tree/master/docs/SRS/CA.pdfTable of Symbols

Contents

List of Tables

List of Figures

Verification and Validation .....[Needs more explanations —Malavika]

This document explains the verification and validation plan to improve the quality of CFS. There are several standards available for software quality and according to the quality model of ISO 9126, software quality is described as a structured set of characteristics namely - Functional suitability, Performance, efficiency, Compatibility, Usability, Reliability, Security, Maintainability and Portability **?**.

Here we restrict ourselves to Functional suitability, Maintainability and portability.

3 General Information

This section explains the summary of what is being tested in this document, the objectives of this document and references for this document.

3.1 Summary

This document will summarize the plan for verification and validation of CFS for compliance with the requirements specified in the CA document. The goal statement as found in CA document is presented below.

"Given the set of data points, the choice of software from CFS and the variabilities of the software the CFS should:

1. compute the parameters of the curve which is the best possible fit through the set of data points. "

The process explained in the goal statement is called curve fitting. There different choices like regression, interpolation and smoothing. This is explained better in the Figure 1.

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

3.2 Objectives

The goal of verifying and validating is to increase improve the quality of the CFS and obtain confidence in the software implementation. The qualities

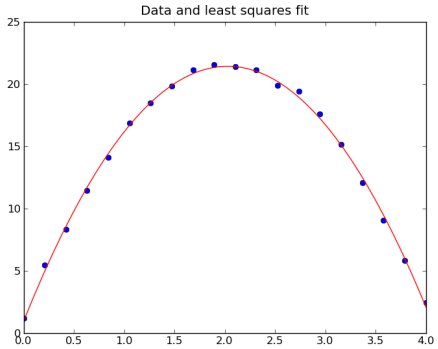

Figure 1: Typical example of a curve fitting process

which are important concerning CFS are correctness(Functional suitability), maintainability, re-usability, portability. The definitions of the above mentioned qualities are explained below.

3.2.1 Functional Suitability

This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. It can be further characterized into completeness, correctness and appropriateness.

Correctness The degree to which a product or system provides the correct results with the needed degree of precision.

3.2.2 Maintainability

This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in the environment, and in requirements. Reusability is an important characteristic of maintainability.

Reusability The degree to which an asset can be used in more than one system, or in building other assets.

3.2.3 Portability

The degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

[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 References

Throughout this document, we refer to the terminologies that have been already explained in the CA document for Program CFS. [Reference relevant documentation. This will definitely include your SRS —SS]

4 Plan

4.1 Verification and Validation Team

1. Malavika srinivasan

[Probably just you. :-) —SS]

4.2 SRS Verification Plan

The CA document for CFS will be reviewed by Dr.Spencer Smith and my classmate Mr.Robert White.

[List any approaches you intend to use for SRS verification. This may just be ad hoc feedback from reviewers, like your classmates, or you may have something more rigorous/systematic in mind.. —SS]

4.3 Design Verification Plan

My design will be verified with the help of my supervisor Dr.Spencer Smith and my classmates Ms.Jennifer Garner and Mr.Brooks MacLachlan. [Plans for design verification —SS]

4.4 Implementation Verification Plan

My implementation will be verified by the tests listed in this document and the unitVnVplan document. My classmate Ms.Vajiheh Motamer will help me verify the document. [You should at least point to the tests listed in this document and the unit testing plan. —SS]

4.5 Software Validation Plan

[I am a little confused about this section. I am not sure if this is relevant for my general purpose tool. I have other softwares from which I can do parallel testing but not an external data for vaidation. Needs to be confirmed. —Malavika]

[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]

5 System Test Description

System testing is a process in which only the overall working of the system. The instance models in CA document will be tested here. This does not test the inidividual units or modules of the system. It is a black box testing approach.

5.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

5.1.1 Interpolation Testing

[Not sure if I can have an area for interpolation and regression, but I have it here as a place holder —Malavika]

Testcase for Polynomial interpolation

1. **T1: Simple example of monomial interpolation**

   Control: Automatic [Pytest or robot framework —Malavika]

   Initial State: NA

   Input: Data points (-2,-9), (-1, -15), (0,-5), (1,-3), (2,39)

   Output: (5, 4, 7, 2, 3) (coefficients of the increasing powers of x, starting from x^0)

   How test will be performed: [I am not sure what goes here —Malavika]

2. T2: Simple example of Lagrange's interpolation

   Control: Manual versus Automatic

   Initial State:NA

   Input: Data points (4.1168, 0.213631), (4.19236,0.214232), (4.20967, 0.21441, 4.46908, 0.218788)

   Output: (0.871839, -0.386008, 0.0695519, -0.00355245) (coefficients of the increasing powers of x, starting from x^0)

   How test will be performed: [I am not sure what goes here —Malavika]

3. T3: Simple example of Newton interpolation

   Control: Automatic [Pytest or robot framework —Malavika]

   Initial State: NA

   Input: Data points (0,1), (1,2.25), (2,3.75), (3,4.25)

   Output: 1, 1.25, 0.125, -0.20833 (coefficients of the increasing powers of (x - x_i, starting from $(x - x_i)^0$ where i = 0,1,2)

   How test will be performed: [I am not sure what goes here —Malavika]

4. T4: Simple example of Hermite Cubic interpolation

   Control: Automatic [Pytest or robot framework —Malavika]

   Initial State: NA

Input: Data points (0,1), (1,2.25), (2,3.75), (3,4.25)

Output: 1, 1.25, 0.125, -0.20833 (coefficients of the increasing powers of (x - x_i, starting from $(x - x_i)^0$ where i = 0,1,2)

How test will be performed: [I am not sure what goes here —Malavika]

5. T5: Simple example of BSpline interpolation

   Control: Automatic [Pytest or robot framework —Malavika]

   Initial State: NA

   Input: Data points (0,1), (1,2.25), (2,3.75), (3,4.25)

   Output: 1, 1.25, 0.125, -0.20833 (coefficients of the increasing powers of (x - x_i, starting from $(x - x_i)^0$ where i = 0,1,2)

   How test will be performed: [I am not sure what goes here —Malavika]

5.1.2 Regression

6. T6: Simple example of regression using normal equations

   Control: Automatic [Pytest or robot framework —Malavika]

   Initial State: NA

   Input: Data points

   Output:

   How test will be performed: [I am not sure what goes here —Malavika]

7. T7: Simple example of regression using Augmented system

   Control: Automatic [Pytest or robot framework —Malavika]

   Initial State: NA

   Input: Data points

   Output:

   How test will be performed: [I am not sure what goes here —Malavika]

8. T8: Simple example of regression using Orthogonal transformations

   Control: Automatic [Pytest or robot framework —Malavika]

   Initial State: NA

   Input: Data points

   Output:

   How test will be performed: [I am not sure what goes here —Malavika]

5.2 Tests for Nonfunctional Requirements

Testcase for Piecewise Polynomial interpolation

5.2.1 Correctness

Parallel testing

1. T9: Parallel Testing

   Type:

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

5.2.2 Maintainability

2. T9:

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.2.3 Area of Testing2

...

5.3 Traceability Between Test Cases and Requirements

The following table shows the traceability mapping for test cases, instance models and the requirements. [I feel adding instance models here makes the table clearer. Needs to be discussed —Malavika]

Table 1: Requirements Traceability Matrix

Test Number	Instance Models	CA Requirements
T1	IM3, IM1	R5, R6
T2	IM4, IM1	R5, R6
T3	IM5, IM1	R5, R6
T4	IM6, IM1	R5, R7
T5	IM7, IM1	R5, R7
T6	IM8, IM2	R5, R8
T7	IM9, IM2	R5, R8
T8	IM10,IM2	R5, R8

6 Static Verification Techniques

[In this section give the details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]

References

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]