Project Title: System Verification and Validation Plan for MTOBridge

Team 15, Alpha Software Solutions
Badawy, Adham
Yazdinia, Pedram
Jandric, David
Vakili, Farzad
Vezina, Victor
Chiu, Darren

October 30, 2022

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

Contents

List of Tables

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

List of Figures

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

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

[symbols, abbreviations or acronyms – you can simply reference the SRS (**?**) tables, if appropriate —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 (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]

?

4 Plan

[Introduce this section. You can provide a roadmap of the sections to come. —SS]

4.1 Verification and Validation Team

[You, your classmates and the course instructor. Maybe your supervisor. You shoud do more than list names. You should say what each person's role is for the project. A table is a good way to summarize this information. —SS]

4.2 SRS Verification Plan

[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]

[Remember you have an SRS checklist —SS]

4.3 Design Verification Plan

[Plans for design verification —SS]

[The review will include reviews by your classmates —SS]

[Remember you have MG and MIS checklists —SS]

4.4 Implementation Verification Plan

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walk-throughs, code inspection, static analyzers, etc. —SS]

4.5 Automated Testing and Verification Tools

[What tools are you using for automated testing. Likely a unit testing framework and maybe a profiling tool, like ValGrind. Other possible tools include a static analyzer, make, continuous integration tools, test coverage tools, etc. Explain your plans for summarizing code coverage metrics. Linters are another important class of tools. For the programming language you select, you should look at the available linters. There may also be tools that verify that coding standards have been respected, like flake9 for Python. —SS]

[The details of this section will likely evolve as you get closer to the implementation. —SS]

4.6 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]

5 System Test Description

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]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. —SS]

5.1.1 Area of Testing1

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

Title for Test

1. test-id1

   Control: Manual versus Automatic

   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

   Control: Manual versus Automatic

   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:

5.1.2 Area of Testing2

...

5.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. —SS]

[Tests related to usability could include conducting a usability test and survey. —SS]

5.2.1 Look and Feel Requirements

Graphics Informative Value Test

1. test-id1

   Initial State: Bridge UI mockups containing graphic elements will be prepared

   Type:

   How test will be performed: Civil engineers will be presented with mockups and expected to identify what the graphic elements represent with at least 90% accuracy.

5.2.2 Usability and Humanity Requirements

Intuitiveness Test

1. test-id1

   Initial State: Application is opened with no past data

   Type:

Input/Condition: User inputs

Output/Results: Bridge analysis within 5 minutes of introduction

How test will be performed: Civil engineer will be asked to provide inputs

Display Resolution Test

1. test-id1

   Initial State: Application is opened with no past data

   Type:

   Input/Condition: Application is running on a computer with 1280x720 display

   Output/Results: All buttons and animation window remain visible and accessible

   How test will be performed:

2. test-id2

   repeat the above for 1366x768, 1920x1080, 2560x1440, 3840x2160

Text Resizing Test

1. test-id1

   Initial State: Application is opened with no past data

   Type:

   Input/Condition: Font size setting is changed to display as 8pt

   Output/Results: All buttons and animation window remain visible and accessible

   How test will be performed:

2. test-id2

   repeat the above for 16pt, 24pt, 32pt

Ease of Installation Test

1. test-id1

   Initial State: Application is not installed

   Type:

   Input/Condition: Application is downloaded and installed

   Output/Results: Application is ready to use within 30 minutes

   How test will be performed: By civil engineer

Consistent UI Test

1. test-id1

   Initial State: Visually similar UI elements are grouped into categories by developers

   Type:

   Input/Condition: Civil engineer asked to associate UI elements with their respective categories

   Output/Results: At least 90% of UI elements are sorted into their predefined categories

   How test will be performed:

5.2.3 Performance Requirements

UI Speed Test

1. test-id1

   Initial State: Application is opened with no past data

   Type:

   Input/Condition: Application used to generate a large bridge analysis

   Output/Results: Total delay will not exceed underlying MATLAB script's execution time by 10%

   How test will be performed:

5.2.4   Operational and Environmental Requirements

User Computer Test

1. test-id1

   Initial State: Application is opened on expected users' computer

   Type:

   Input/Condition: Application used to generate a large bridge analysis

   Output/Results: Delay on user computer will not exceed delay on developer computers by 10%(?)

   How test will be performed:

5.2.5   Maintainability and Support Requirements

Ease of Maintenance Test

1. test-id1

   Type:

   Input/Condition: Measure code file length in lines

   Output/Results: At least 75% of files will contain 750 lines of code or less

   How test will be performed:

2. test-id2

   Type:

   Input/Condition: Measure method length in lines

   Output/Results: At least 75% of methods will contain 75 lines of code or less

   How test will be performed:

3. test-id3

Type:

Input/Condition: Measure length of code lines

Output/Results: At least 75% of lines will contain **120** characters or less

How test will be performed:

4. test-id4

Type:

Input/Condition: Measure nesting depth of methods

Output/Results: At least 75% of methods will have a maximum nesting depth of 5 or less

How test will be performed:

5.2.6   Area of Testing1

Title for Test

1. test-id1

Type:

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:

5.2.7 Area of Testing2

...

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 and explain your overall philosophy for test case selection. —SS] [This section should not be filled in until after the MIS 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]

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]

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1.

2.