HubListener: Verification & Validation Plan

Zed Ahmad, Prakhar Jalan, Pedro Oliveira, Piranaven Selvathayabaran

February 14, 2019

1 Revision History

Date	**Version**	**Notes**
November,21 2018	1.0	Template Setup

2 Symbols, Abbreviations and Acronyms

The symbols, abbreviations and acronyms used in this document include those defined in the table below.

symbol	**description**
MIS	Module Interface Specification
MG	Module Guide
TC	Test Case
VnV	Verification and Validation

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 General Information **1**
- 3.1 Summary 1
- 3.2 Objectives 1
- 3.3 References 1

4 Plan **2**
- 4.1 Verification and Validation Team 2
- 4.2 SRS Verification 2
- 4.3 Design Verification Plan 2
- 4.4 Implementation Verification Plan 3

5 System Test Description **3**
- 5.1 Expected Program Output 3
- 5.2 User Interaction and Navigation 4
- 5.3 Installability, Help, and Storage 5
- 5.4 Traceability Between Test Cases and Requirements 7

6 Unit Testing **7**

7 Automated Testing **7**

8 Appendix **8**
- 8.1 Usability Survey Questions 8

List of Tables

1 Traceability matrix showing the connections between functional requirements and test cases 7

This document outlines the system verification and validation plan for the HubListener software. General information regarding the system under test and the objectives of the verification and validation activities are provided in Section 3. Overviews of the verification plans for the SRS, design, and implementation are given in Section 4, along with a summary of the validation plan for the software.

3 General Information

3.1 Summary

The software being tested is the open-source software, HubListener. HubListener provides users with relevant metrics, trends, and information regarding their GitHub project, and in some cases, push the user to make appropriate changes (can be organizational) that are intended to lean the project towards a more 'successful' one.

3.2 Objectives

The purpose of the verification and validation activities is to confirm that HubListener exhibits the desired software qualities. The primary objective is to build confidence in the correctness of the software. The tests described in this document cannot definitely prove correctness, but they can build confidence by verifying that the software is correct for the cases covered by test.

3.3 References

Information about the purpose and requirements of HubListener can be found in the SRS document. The latest documentation for HubListener can be found on GitHub: HubListener

4 Plan

4.1 Verification and Validation Team

The HubListener Team, which includes all the authors of this document will be responsible for the verification and validation of HubListener. Input from our supervisor, Dr. Smith will also contribute to the VnV Plan and he will ensure proper procedures take place.

4.2 SRS Verification

SRS Verification will be carried out by reviews. The HubListener team will do two reviews. One review will take place on January 31st, 2019, the expected launch date of Version 1. A second review will occur at the launch date of Version 2 which will be determined at a later date. These reviews will be done as a collective in separate meetings to confirm theories and models in the SRS are correct. Any issues identified during this review will be noted down and new Issues on GitHub will be created. These issues will have top priority and will be completed in the next possible Sprint by one member of the HubListener team. This review will be followed up by additional reviews by Dr. Spencer Smith and Dr. Anand (if he wishes). Again, any issues identified by these reviewers will be recorded through the issues tracker on GitHub. A focus of these reviews will be to verify the functional and non-functional requirements of correctness and understandability by identifying information from the SRS that is incorrect or ambiguous.

4.3 Design Verification Plan

The design of HubListener will be outlined in the Module Guide (MG) and Module Interface Specification (MIS) documents. The design will be verified by review of these documents. Dr. Smith will review this documents. To verify correctness, part of this review til be to ensure that every module traces to a requirement and that every requirement is traced to by a module. Any issues identified during this review will be noted down and new Issues on GitHub will be created. These issues will have top priority and will be completed in the next possible Sprint by one member of the HubListener team. Reviews of these design documents will also focus on ensuring the understandability by identifying descriptions and specifications that are ambiguous.

4.4 Implementation Verification Plan

The implementation of HubListener will be verified by review and by testing. The HubListener team will extensively review the implementation. This will be done via bi-weekly code-reviews. In the context of agile methodologies, every sprint, each member will be assigned code from a peers to review. This code is typically from the previous sprint. If the code passes code review, it will be merged into the master-branch. Any issues identified during this review will be noted down and new Issues on GitHub will be created. These issues will have top priority and will be completed in the next possible Sprint by one member of the HubListener team. These reviews will contribute to verifying correctness and understandability of the software by identifying code that is not traceable to any specifications described in the SRS , MG or MIS.

The implementation will also be verified through testing. Specific test cases are outline in Section 5 of this document. Test cases that are directly dependent on implementation details will be outlined at a later date. All tests will be written(where applicable), reviewed and executed by the HubListener Team.

5 System Test Description

In this section, we will describe and define the test cases that will be used for testing functional requirements.

5.1 Expected Program Output

The following test case covers the user input and its appropriate output.

TC1: test-metricsOut

Control: Automatic

Initial State: New session

Input: GitHub project link

Output: The following list of metrics - in any order - is expected given a github project link as input. This output will be provided on the CLI, and in JSON format. Once our UI design is confirmed, it will be outputted on-screen as well.

- Cyclomatic Complexity
- Cyclomatic Density
- Lines of Code
- Lines of Comments
- Logical Lines of Code
- Halstead Metric
- Number of Methods
- Number of Variables
- Number of Issues
- Number of Bugs
- Number of Stars
- Functional Coverage Score
- Condition Coverage Score

How test will be performed: Automated test on BATS testing framework.

TC2: test-trendsOut

Control: Manual

Initial State: New session

Input: GitHub project link

Output: Any pair(s) of the listed metrics above will be analyzed, and one or more trends will be displayed on the UI. These trends will be in the form of graphs, in order to easily notice any patterns, or make observations.

How test will be performed: A pair of metrics will be selected, and every combination of a graph will be manually created via

5.2 User Interaction and Navigation

The following set of test cases are intended to cover Hublistener's UI navigation.

TC3: test-interact1

Control: Manual

Initial State: New Session

Input: User selects which metrics they wish to analyze via the User Interface (website). This will be accomplished through the use of check boxes for every metric.

Output: Only the selected metrics will be displayed on the UI.

How test will be performed: Visual inspection of output

TC4: test-nav1

Control: Manual

Initial State: New Session

Input: TBD

Output: TBD

How test will be performed: TBD

5.3 Installability, Help, and Storage

The following set of test cases are intended to cover Hublistener's installability, help dialogue functionality, and analysis storage within the database.

TC5: test-inst1

Control: Automatic

Initial State: New Session

Input: To install the tool, the 'npm install hublistener' command is entered as input into the CLI. To then uninstall, the 'npm uninstall hublistener' command is entered.

Output/Result: The appropriate packages (files, dependencies etc.) are successfully installed and a large set of details regarding what is installed is displayed in sequence onto the CLI (with no errors).

How test will be performed: This test will consist of 2 components; installing, and uninstalling the HubListener tool. It will be automated using BATS testing framework.

TC6: test-update1

Control: Automatic

Initial State: New Session

Input: To update the tool, the ‘npm update’ command is entered into the CLI.

Output/Result: The appropriate packages (files, dependencies etc.) are successfully updated in any sequence, or a verification message regarding the tool being up-to-date, is outputted on the CLI.

How test will be performed: This test is for updating the HubListener package(s). It will be automated using BATS testing framework.

TC7: test-help

Control: Automatic

Initial State: New session

Input: Either the ‘hublistener -h’ OR ‘hublistener –help’ command will be used to bring up the help dialogue within the CLI.

Output/Result: The help dialogue is successfully displayed onto the CLI, details of which are TBD.

How test will be performed: Automated using BATS testing framework.

TC8: test-db1

Control: Automatic

Initial State: New session

Input: Github link

Output/Result: An entry is successfully added to the database.

How test will be performed: The database will be queried for the appropriate github link and its respective metrics; which verifies that it was added to the database. This will be automated using Jasmine testing framework.

5.4 Traceability Between Test Cases and Requirements

The purpose of the traceability matrix shown in Table 1 below, is to provide easy references on which requirements are verified by which test cases, and which test cases need to be updated if a requirement changes. If a requirement is changed, the items in the column of that requirement that are marked with an "X" may have to be modified as well.

	R1	R2	R3	R4	R5	R6	R7
TC1	X	X					
TC2			X				
TC3				X			
TC4				X			
TC5 - TC6					X		
TC7						X	
TC8							X

Table 1: Traceability matrix showing the connections between functional requirements and test cases

6 Unit Testing

Unit Testing will be completed using either Mocha.js or Jasmine.js . Both tools provide the resources required to get thorough unit testing complete. We are still in the research phase and not have decided on a set tool. Jasmine.js remains the front-runner as one of members on the HubListener team has experience with this tool.

7 Automated Testing

Automated testing will further enhance our testing efforts. We are looking into testing frameworks for the command line, such as Bats: Bash Automated Testing System which can be used for some of our system tests. We are also investigating Travis-CI, which is a free web based service that allows to register a trigger on GitHub so that every time a commit is pushed to GitHub an isolated Ubuntu container with the correct container that we want to test, builds the software (if needed) and then runs the test.

8 Appendix

This is where you can place additional information.

8.1 Usability Survey Questions

Here is where the Usability Survey Questions will go.