CMS-2 Source Monitor and Analyzer

Validation Plan

Version: 1.1

Author: Chris Curreri

Date: April 21, 2017

Sponsor: ASRC Federal Missions Solutions

Representatives: Chris Bartley, Craig Sobieralski

Table of Contents

1 Purpose

This document is the Validation Plan for CMS-2 Source Monitor and Analyzer version 1.0. It defines the scope and goal of validation testing to ensure that the program does what our sponsor requires. The CMS-2 Source Monitor and Analyzer must replace the outdated Fortran tool successfully without any critical failures. This document will explain the tests that have been completed before the program's release and the quality assurance plans that have been put in place.

2 Scope

The scope of this effort is to verify that the solution functions according to the design specifications and satisfies its requirements. The following types of verification will be performed:

- Testing: Verification will be done by running automated test using the python unit testing framework called "unittest".
- CMS-2 input: Sample CMS-2 and CMS2Y sample directories will be created to test if the program can properly analyze the samples. The files created test a specific analysis portion of the program. A file that contains multiple analysis components might be created in the future.

2.1 Exclusions/Assumptions

The validation did not include a larger CMS-2 file that test all components being analyzed at once. However, the regular expressions that were used to analyze CMS-2 and CMS2Y work in independence of each other. In case a regular expression counts the wrong component, unit testing thoroughly tested the regex.py class so that it would count the right number of components.

2.2 Limitations

Since the original CMS-2 Source Monitor and Analyzer is classified, validation could not compare the new program's code with the old one. This created a black box on the program's functionality making it difficult for Team Ducks to ensure that analyzed components in the generated reports are what our sponsor wants.

3 Validation Strategy

Team Ducks ensures quality assurance by writing proper documentation, having a solid Validation Testing Strategy, and meeting our sponsor's Acceptance Criteria. These steps insure that we are developing the right product for our sponsor's needs. The program's functionality can be traced back to the original requirements we were given and rigorous testing validates the product's quality.

3.1 Documentation

The Tractability Matrix was created to trace the software development process from requirements, to design, to testing. The software requirements were documented in the requirements document and the structure of the program that would fulfill these requirements was recorded in the design document. The Verification/Testing Script documents the steps that took place to test the program as well as the verification testing strategy.

3.2 Validation Testing Strategy

The Validations testing strategy involves creating multiple CMS-2 test files and Verification through unit testing. The CMS-2 and CMS2Y files were created to test one or more analysis component. Some were created for a specific analysis in mind such as analyzing the number of data statements. A control file use to analyze all components at once has not been created However CMS-2 is a simple program. All analysis processes run independently from one another so we didn't find it necessary to create one. If any of the analysis components count the wrong component, it would have been caught in unit testing. Team Ducks also has one week left in the final sprint so adding a control file may be possible.

A sample of the testing scripts are shown below:

- 1.1 `test_linenums…ok`- Will print ok if and only if the number of lines returned for each file of sample code matches the corresponding number in the total_lines.txt file in the expected_output directory.
- 1.5.1 `test_bad_ext...ok`- Will print ok if and only if the number of files with incorrect extensions detected in the sample folder matches that found in the num_bad_ext.txt file in the expected_output directory.
- 3.2 `test_num_errors...ok`- Will print ok if and only if the summary returned by the report for the directory of sample code files contains numbers of files with incorrect extensions and component name / filename mismatches that correspond to those found in the report_num_errors.txt file in the expected_output directory.

- 3.4 `test_times...ok-` Will print ok if and only if the main report string for the sample code files directory contains a valid start time and completion time, and that the duration between them is correct.
- 4.3 `test_monitor_report...ok-` Will print ok if and only if the CMS2FileDiff structure returned by the source monitor for each pair of new and old sample code files contains file and line numbers for additions, modifications, and deletions that correspond to those found in the test_file_line_no.txt file in the expected_output directory.

3.3 Acceptance Criteria

The testing scrips can either pass or fail on each unit test. If all steps pass, the script will be marked as passed and Team Ducks will divert their time to either extend the functionality of the program or create a control file for the input. If any of the tests fail, the team will do either one of two things. If the tests that fail are minimal, then the team will divert their time to fixing all bugs for the remainder of the sprint. If a majority of the tests fail, the team will divert their time to fix the damage manageable within the final week and document the failures of the program scripts that were beyond repair. However, risk of this outcome is minimal.

4 Roles and Responsibilities

Role	Responsibilities
Johan Burke	Verification/Testing Script and all unit testing
Tom Harker	Requirements Document
Matthew Gimbut	Design Document
Chris Curreri	Tractability Matrix and Validation Plan

5 Terms and Definitions

Term or Acronym	Definition
analyze	Refers to the components that need to be counted for the final report.
component	The items that need to be counted for the final report such as comments, data statements, and executable statements as well as additions, modifications, and deletions, for comparing files.
validation	Testing that encompasses all aspects of the program. It ensures that we are creating the right product for our sponsors.
verification	Refers to the unit testing documented in the verification script.

6 Revision History

Version	Version Date	Revisions
1.0	April 14, 2017	Present Version