Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	**☐ Non Production ☒Production**	**Script Purpose**	**☐ Install ☒ Verify**	**Run #**	

APPROVERS are not required

SCRIPT DETAILS

Objective

Purpose

This document details the steps involved in verifying that the CMS-2 Source Monitor and Analyzer, developed by the Rowan Ducks team, meets the requirements set forth by ASRC-FMS. The document is aimed at testers at ASRC-FMS.

Strategy

Verification of requirements will be performed by running automated tests on the software using the testing framework provided by python, “unittest.” Tests will involve running the Source Monitor and Analyzer on prepared sample CMS-2 code. The data output by the Source Monitor and Analyzer will be compared to correct data found in files that correspond to each sample code file. The software will pass tests if each required field of the monitor and analyzer output matches that of the corresponding field in the file that specifies the correct output.

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

Requirements

Source Analyzer Functional Requirements

Requirement Reference #	**Description**
1.1	SAN shall be able to detect the total number of lines in a file.
1.2	SAN shall be able to detect the number of executable statements in a file.
1.2.1	SAN shall be able to identify and count the number of GOTO statements in a file
1.3	SAN shall be able to detect the number of notes in a file.
1.4.1	SAN shall be able to detect and count single line comments in a file.
1.4.2	SAN shall be able to detect and count multi-line comments in a file.
1.5.1	SAN shall be able to detect and count instances of files with incorrect extensions.
1.5.2	SAN shall detect and count instances of the key phrase “END-SYSTEM” being misplaced. This phrase is only allowed at the end of a block of CMS-2 Direct
1.5.3	SAN shall detect and count instances of multiple components being in the same file
1.5.4	SAN shall detect and count instances of component name and filename mismatch
2.1	The detected elements shall include lines, comments, multi-line comments, notes, multi-line notes, data statements, executable statements, and multi-line statements.
3.1	The report generated by the Source Analyzer should use an identical format to the current report in use by ASRC-FMS.
3.2	The report generated by the Source Analyzer shall report the number of occurrences of files with incorrect extensions and component name / file mismatches.
3.3	The report shall list all files with procedures longer than 250 lines.
3.4	The report shall contain the Source Analyzer starting time, completion time, and runtime duration.

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

Source Monitor Functional Requirements

Requirement Reference #	**Description**
4.1	The Source Monitor shall be able to identify changes between either two High Level CMS-2 files or two CMS-2 Direct files
4.2	The detected changes shall include lines, comments, multi-line comments, notes, multi-line notes, data statements, executable statements, and multi-line statements.
4.3	SM shall generate a report that indicates the file and line number of all additions, modifications and deletions in its input.

Non-Functional Requirements

Requirement Reference #	**Description**
5	The product shall be created without the inclusion of any proprietary software or third party tools which are not freely available.
6	The product shall be created using Python and run on a UNIX/Linux environment.
6.1	The product shall run from a command line interface.
7.1	The product design should allow ASRC-FMS to extend functionality to the analysis of other programming languages.
7.2	The product should be easily compatible with the addition of a graphical user interface.
7.3	The design of the product should allow for easy adaptation and revision of the SAN and SM report format.
8.1	The runtime of the Source Monitor/Analyzer should require less than 5 seconds if the input is less than 5 files.
8.2	The runtime of the Source Monitor/Analyzer should require less than 4 hours if the input is the full CMS-2 code base.

9	The Source Monitor shall be triggered to run automatically when a file is checked in to the code repository. Its input shall be these newly checked-in files.

Prerequisites and Setup

Test environment must have python installed (version 3.2 or greater).

Testing environment must have the "testing" subdirectory of "src" installed. This directory has the following children:

- **data:** directory containing sample input files for the source monitor and analyzer.
- **expected_output:** directory containing data corresponding to the correct output for each test.
- **source_monitor:** directory containing python scripts that specify tests of the source monitor.
- **source_analyzer:** directory containing python scripts that specify tests of the source analyzer.

The working directory must be set to **src/testing** prior to running tests. The automated script to run is `python3 -m unittest discover -v [sub_dir_name]`, where sub_dir_name is either source_monitor or source_analyzer.

STEPS

Step	Description	Expected Results	Actual Results/Comments	Pass/Fail
Test source analysis results – Requirements validated: 1.1-1.5.4, 2.1, 3.2-3.4				
1.	Run 'python3 -m unittest discover -v source_analyzer' This command will run the scripts of steps 1.1 – 1.5, 2.1, and 3.1-3.4 and their substeps automatically.	15 tests each run and return an "ok" status.		

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

Step	Description	Expected Results	Actual Results/Comments	Pass/Fail
1.1	Run test_linenums (in test_stmts.py)	`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.		
1.2	Run test_exec_stmts (in test_stmts.py)	`test_exec_stmts…ok` Will print ok if and only if the number of executable statements returned for each file of sample code matches the corresponding number in the exec_stmts.txt file in the expected_output directory.		
1.2.1	Run test_goto_stmts (in test_stmts.py)	`test_goto_stmts…ok` Will print ok if and only if the number of goto statements returned for each file of sample code matches the corresponding number in the goto_stmts.txt file in the expected_output directory.		

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

Step	**Description**	**Expected Results**	**Actual Results/Comments**	**Pass/Fail**
1.3	Run test_notes (in test_stmts.py)	`test_notes...ok` Will print ok if and only if the number of notes returned for each file of sample code matches the corresponding number in the num_notes.txt file in the expected_output directory.		
1.4.1	Run test_sl_comments (in test_stmts.py)	`test_sl_comments...ok` Will print ok if and only if the number of single line comments returned for each file of sample code matches the corresponding number in the num_sl_comments.txt file in the expected_output directory.		
1.4.2	Run test_ml_comments (in test_stmts.py)	`test_ml_comments...ok` Will print ok if and only if the number of multi-line comments returned for each file of sample code matches the corresponding number in the num_ml_comments.txt file in the expected_output directory.		

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

Step	Description	Expected Results	Actual Results/Comments	Pass/Fail
1.5.1	Run test_bad_ext (in test_improper.py)	`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.		
1.5.2	Run test_misplace_end_sys (in test_improper.py)	`test_misplace_end_sys...ok` Will print ok if and only if the number of instances of misplaced "END-SYSTEM" statements for each file of sample code matches the corresponding number found in the num_misplaced_end_sys.txt file in the expected_output directory.		
1.5.3	Run test_multi_component (in test_improper.py)	`test_multi_component...ok` Will print ok if and only if the number of instances of multiple components being in a file matches the number found in the num_multi_component.txt file in the expected_output directory.		

Script Identifier	Burke_DucksVerificationScript			Script Version	1.0
Environment	☐ Non Production ☒Production	Script Purpose	☐ Install ☒ Verify	Run #	

Step	Description	Expected Results	Actual Results/Comments	Pass/Fail
1.5.4	Run test_name_mismatch (in test_improper.py)	`test_name_mismatch...ok` Will print ok if and only if the number of mismatches between component name and filename matches the number found in the num_name_mismatch.txt file in the expected_output directory.		
2.1	Run test_all_stmts (in test_stmts.py)	`test_all_stmts...ok` Will print ok if and only if the resulting CMS2File structure contains fields for lines, comments, multi-line comments, notes, multi-line notes, data statements, executable statemetns, and multi-line statements for each sample code file corresponding to those found in the all_stmts.txt file in the expected_output directory.		
3.1	Run test_report_format (in test_report.py)	`test_report_format...ok` Will print ok if and only if the columns string returned by the report for one sample input file matches that found in the report_column_headings.txt file in the expected_output directory.		

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	**☐ Non Production ☒Production**	**Script Purpose**	**☐ Install ☒ Verify**	**Run #**	

Step	Description	Expected Results	Actual Results/Comments	Pass/Fail
3.2	Run test_num_errors (in test_report.py)	`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.		
3.3	Run test_list_gt_250 (in test_report.py)	`test_list_gt_250...ok` Will print ok if and only if the list of large (more than 250 lines) files in the sample code files directory matches that found in the list_250.txt file in the expected_output directory.		
3.4	Run test_times (in test_report.py)	`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.		
Test source monitor results – Requirements validated: 4.1-4.4				

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

Step	Description	Expected Results	Actual Results/Comments	Pass/Fail
4.	Run ‘python3 -m unittest discover source_monitor’ This command will run the scripts of steps 4.1-4.4 automatically.	Four tests each run and return an “ok” status.		
4.1	Run test_changes (in test_monitor.py)	`test_changes...ok` Will print ok if and only if the CMS2FileDiff structure returned for each pair of new and old High Level CMS-2 files and CMS-2 Direct files contains numbers of changes that match those found in the num_changes.txt file in the expected_output directory.		
4.2	Run test_change_types (in test_monitor.py)	`test_change_types...ok` Will print ok if and only if the source monitor report generated for each pair of new and old CMS-2 sample code files includes changes in lines, comments, multi-line comments, notes, multi-line notes, data statements, executable statements, and multi-line statements corrresponding to those found in the change_types.txt file in the expected_output directory.		

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

Step	Description	Expected Results	Actual Results/Comments	Pass/Fail
4.3	Run test_monitor_report (in test_monitor.py)	`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.		
4.4	Run test_monitor_format (in test_monitor.py)	`test_monitor_format...ok` Will print ok if and only if the columns string returned by the source monitor for a pair of new and old sample code files matches that found in the monitor_headings.txt file in the expected_output directory.		
Non-functional requirements				
5.1	Ensure the python language and standard library are free to use by going to python.org -> Downloads.	Python can be downloaded without payment.		
5.2	Ensure the gitpython package is free to use by going to https://github.com/gitpython-developers/GitPython and cloning the repository.	GitPython can be downloaded without payment.		

Script Identifier	**Burke_DucksVerificationScript**	**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**
		Run #	

Step	Description	Expected Results	Actual Results/Comments	Pass/Fail
6.1	Run SourceMonitor.py and Report.py from a terminal on a UNIX computer.	The source monitor and analyzer can successfully be run in a terminal window.		
7.1	Verify that regex.py can use a different output data structure and regular expressions to support languages other than CMS-2.	Definitions of a new output data structure and regular expressions for different types of statements of a non-CMS-2 language can be used to generate a Source Analyzer report.		
7.2	Verify that data structures output by the source monitor and source analyzer and the functions that display these data are separate, and the data structures can be imported into code for a graphical user interface.	CMS2File and CMS2FileDiff data structures can successfully be imported and used in graphical user interface code.		
7.3	Verify that the format of the SAN and SM reports can be changed easily.	Columns of the SAN and SM reports can be rearranged easily.		
8.1	Run the source monitor and analyzer on input containing 4 files and check the amount of runtime they output.	The source monitor and analyzer each run for a combined total of less than 5 seconds.		
8.2	Run the source monitor and analyzer on the full CMS-2 code base.	The source monitor and analyzer each run for a combined total of less than 4 hours.		
9	Run a checkin_code.sh shell script that checks code in and runs the source monitor.	Code input is succesfully checked in and script outputs source monitor analysis of the new files.		
System test				
10	User runs 'python3 Report.py SampleDirectory –d'	Tests 10.1-10.3 are successful.		

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

Step	Description	Expected Results	Actual Results/Comments	Pass/Fail
10.1	Check file contents.	Files with valid CMS-2 extensions: contents are output to command line and match those of the corresponding file Files without valid CMS-2 extensions: filename is printed with indication that the extension is invalid		
10.2	Check contents of resulting CMS2File structures.	CMS2File structures are printed to command line, and data matches those found in system_test_analyzer_data.txt in the example_output directory.		
10.3	Check format of resulting SourceAnalysis report.	SourceAnalysis report is printed to command line, and format matches that found in system_test_analyzer_format.txt in the example_output directory.		
11	User runs 'python3 SourceMonitor.py SampleDirectory –d'	Tests 11.1-11.3 are successful.		
11.1	Check file contents.	Contents of CMS-2 files are printed in the order of old_file, new_file, old_file, new_file, etc. Code output to command line matches that found in the corresponding files.		
11.2	Check contents of resulting CMS2FileDiff structures.	CMS2FileDiff structures are printed to command line, and data matches those found in system_test_monitor_data.txt in the example_output directory.		

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

Step	Description	Expected Results	Actual Results/Comments	Pass/Fail
11.3	Check format of resulting SourceMontor report.	SourceMonitor report is printed to command line, and format matches that found in sy		

RUN COMMENTS:

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	**☐ Non Production ☒Production**	**Script Purpose**	**☐ Install ☒ Verify**	**Run #**	

Indicate location of evidence (e.g., attached to script, referenced files)

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	☐ **Non Production** ☒**Production**	**Script Purpose**	☐ **Install** ☒ **Verify**	**Run #**	

ATTESTATIONS

Tester: By Signing below, the Tester(s) attest to completing the steps identified below

Printed Name and Signature		**Steps Completed:**	**Date:**

Script Identifier	**Burke_DucksVerificationScript**			**Script Version**	**1.0**
Environment	**☐ Non Production ☒Production**	**Script Purpose**	**☐ Install ☒ Verify**	**Run #**	

Revision History

Version	Date	Description of Changes
1.0	April 21, 2017	Initial Release