Deliverable 2: Testing Process

Red Team

September 29, 2015

The testing process

The initial tests we are running concern cloning a project in Mercurial

Test Case	1: Generate a test repository
Requirement	Command line arguments generate a repository for use in further tests
Component	Commit
Method	commit –m test
Test Input(s)	“Test text”
CL Arguments	Echo, add, commit
Expected Outcome	A test repository is created and committed, with the commit text reading “Test text”
Comments	Text will be generated using “echo” functions

Test Case	2: Create a file log
Requirement	The repository has a data file containing text added to it, then the file is committed, and the repository is checked for the addition of the new files in its file log
Component	commit
Method	Write, cat, commit
Test Input(s)	Randomly generated numbers and text
CL Arguments	none
Expected Outcome	The test verifies files created are present in the repository

Test Case	3: Clone Command Test

Requirement	Clones files to a specified local directory (verification only of the command, not of the directory contents)
Component	clone
Method	clone
Test Input(s)	new local directory path
CL Arguments	N/A
Expected Outcome	We expect a new directory matching the name specified by the input
Comments	Due to how the system verifies cloning, we do not want to verify the files with this command, merely check that the creation happened and that the cloning command executed.

Test Case	4. Clone to Invalid Destination
Requirement	Clone command aborts
Component	clone
Method	clone
Test Input(s)	directory listed as ‘’
CL Arguments	‘’
Expected Outcome	clone command aborts

Test Case	5. List files in store
Requirement	Verifies the data was correctly copied over
Component	clone
Method	echo
Test Input(s)	None

CL Arguments	N/A
Expected Outcome	Files are matched with their counterparts in the cloned directory

Requirements traceability

Users are most interested in the system meeting its requirements and testing should be planned so that all requirements are individually tested.

- Command line arguments generate a repository for use in further tests
- The repository has a data file containing text added to it, then the file is committed, and the repository is checked for the addition of the new files in its file log
- Clones files to a specified local directory (verification only of the command, not of the directory contents)
- Clone command aborts
- Verifies the data was correctly copied over

Tested items

- Test generating a repository with test-clone.t
    - Ensure the script generates a repository that can be modified
- Creating a File Log
    - Test the adding of a file to the above generated repository
- Cloning files to a local directory
    - A new local directory is created with the repository files
- Invalid Cloning Destination
    - An attempt to clone to an invalid destination is aborted
- Listing files
    - Match files with those cloned into the local directory

Testing schedule

An overall testing schedule and resource allocation. This schedule should be linked to the more general project development schedule.

- September 29th: Deliverable #2
    - 5 of 25 test cases developed
    - Detailed Test Plan Due
- October 3rd: Research on how to develop test framework for Mercurial

- October 7th: Begin developing test framework
- October 11th: Continue to develop test framework
- October 22nd: Deliverable #3 Due
    - Testing framework designed and built
- November 12th: Deliverable #4 Due
- November 24th: Deliverable #5 Due

Test recording procedures

It is not enough simply to run tests; the results of the tests must be systematically recorded. It must be possible to audit the testing process to check that it has been carried out correctly.

Test Case: <testNumber> + " " + <testName>

Requirement: <requirementText>

Component: <componentName>

Method: <methodName>

Test Input: <specified test input> **Command Line Arg:** <clArg>

Expected Outcome: <outcome>

Hardware and software requirements

This section should set out the software tools required and estimated hardware utilisation.

- Firefox to display results of testing output
- Ubuntu (or other flavor of Linux)
- Python

Constraints

Constraints affecting the testing process such as staff shortages should be anticipated in this section.

- Scheduling Conflicts
  May have other school work priorities
- Travel Constraints
  Potential asynchronous meetings off campus in the future.