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	01
Requirement	Break a given integer into the largest increments of time it can be broken into.
Component	progress.py
Method	fmtremaining(int)
Test Input(s)	100
Expected Outcome	1m40s

Test Case	02
Requirement	GIven the range of two points, calculate the point of overlap between them
Component	simplemerge.py
Method	intersect(ra, rb)
Test Input(s)	(0,100), (50,150)
Expected Outcome	(50,100)

Test Case	03
Requirement	Returns the number of processors available to the operating system (This test assumes you are running a single-processor virtual machine, so it will fail if more than 1 processor is available to the operating system)

Component	worker.py
Method	countcpus()
Test Input(s)	n/a
Expected Outcome	1

Test Case	04
Requirement	Return the length of the given string
Component	templatefilters.py
Method	count(String i)
Test Input(s)	‘’abcde”
Expected Outcome	5

Test Case	05
Requirement	Return elements in an array concatenated into one element in a larger array
Component	namespaces.py
Method	tolist(String)
Test Input(s)	“1”, ”2”, “3”, “4”
Expected Outcome	[‘1234’]

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 a time output given a number of seconds as input
  ○ Returns the number of seconds formatted as m/s or h/m or d/h, etc. depending on size of integer input

- Test to find the intersection of two lines.
  ○ If there is a point of intersection, then that point is returned. Otherwise nothing is returned

- Counts the number of processors available to the operating system
  ○ returns the number of processors.

- Count the length of a String
  - Given a string as input, return the length of that string
- Return an array as a list
  - Given an array, return all elements concatenated into a single list

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.