Automated Testing Framework For Amara

By
Josh Jettie and John Maruhn

Our Choices

Team Blue was originally between three H/FOSS projects to build our framework for.

An free software developed to help international aid organizations manage the information from their projects and it was written in Java.

A subtitling software that allows anyone to subtitle youtube videos.

An electronic medical record system.

Amara

- We ended up choosing Amara because it was written in Python and we were all generally interested in it.
- Amara is an award winning subtitling software that allows anyone to subtitle any video on Youtube.
- Amara's primary use is to allow anyone to subtitle a YouTube video, or translate a YouTube video making the video more accessible to others.

Learning Points

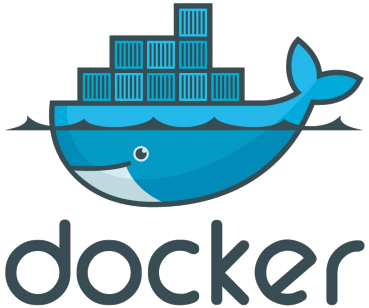
- To develop Amara in linux we had to Download and Install VirtualBox.

VirtualBox

- VirtualBox supports the creation and management of guest virtual machines.
- We Learned to allot yourself PLENTY of space within VirtualBox.

Deliverable #1

Deliverable one is to clone the project from the repository and build Amara. Within our VirtualBox



Step 1

Download and install Docker.

Step 2

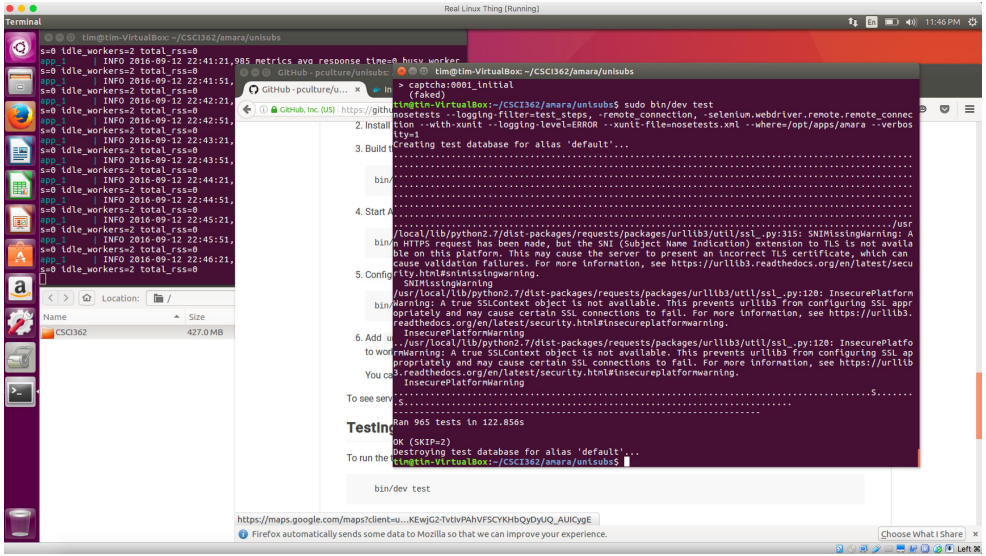
Clone Amara into Git and Docker.

Step 3

Build and run Amara!

Deliverable #1

- When Amara was build we found their built in test suite and ran it.



- Screenshot of the test suite after it was run. It has 965 tests.

Deliverable #2

Deliverable 2 was all about creating 5 of the eventual 25 test cases to be included in our automated framework.

Test One	Test Two	Test Three
Test one tested to see if the data we sent to the database was stored. Method being tested: check_user_data	Test two instantiates a blank user Method being tested: test_create_user_blank_data	Test three checks if a non staff user can create comments Method being tested: user_can_edit_subtitles

Deliverable #2

Test 4	Test 5
Test 4 updates the comments section when given a post request. Method being tested: update_comments	Test 5 Tests to see if we can create a video from a specific URL Method being tested: test_create_videos

Deliverable #3

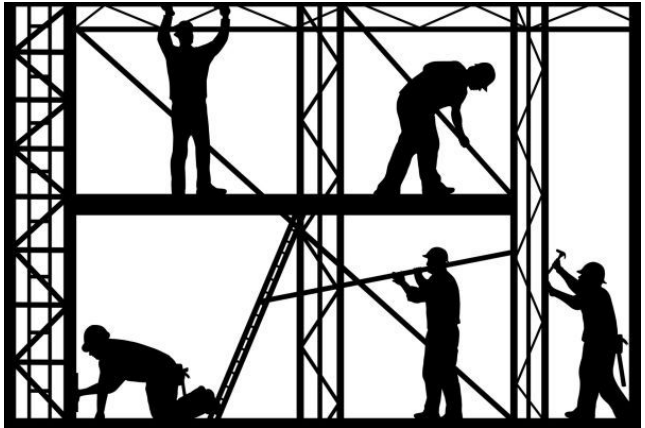
Deliverable 3 was to build an automated testing framework that we will use to implement our test plan.

We wrote our Framework in Python.

We created a driver the reads a test text file and runs the

Method and compares the result with the oracle.

After the method is run the results are printed to an HTML.



Deliverable #3

How to Install and run the testing framework.

Step 1.

Install Git

```
john@john-VirtualBox:~$ sudo apt install git
[sudo] password for john:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-arch git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 277 not upgraded.
Need to get 3,760 kB of archives.
After this operation, 25.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 liberror-perl all 0.17-1.2 [19.6 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 git-man all 1:2.7.4-0ubuntu1 [735 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 git amd64 1:2.7.4-0ubuntu1 [3,006 kB]
Fetched 3,760 kB in 1s (3,401 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 172652 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17-1.2_all.deb ...
Unpacking liberror-perl (0.17-1.2) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.7.4-0ubuntu1_all.deb ...
Unpacking git-man (1:2.7.4-0ubuntu1) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.7.4-0ubuntu1_amd64.deb ...
Unpacking git (1:2.7.4-0ubuntu1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up liberror-perl (0.17-1.2) ...
Setting up git-man (1:2.7.4-0ubuntu1) ...
Setting up git (1:2.7.4-0ubuntu1) ...
```

Step 2.

Clone our Repo

```
john@john-VirtualBox:~$ git clone https://github.com/CSCI-362-02-2016/Blue
Cloning into 'Blue'...
Username for 'https://github.com': maruhntt@g.cofc.edu
Password for 'https://maruhntt@g.cofc.edu@github.com':
remote: Counting objects: 288, done.
remote: Compressing objects: 100% (142/142), done.
remote: Total 288 (delta 86), reused 0 (delta 0), pack-reused 128
Receiving objects: 100% (288/288), 1.83 MiB | 715.00 KiB/s, done.
Resolving deltas: 100% (135/135), done.
Checking connectivity... done.
```

Step 3.

Navigate to the Scripts directory.

Deliverable #3

Step 4:

Run the file runAllTests.py

with Python

Step 5:

An HTML page of the test

results should automatically

open at the end

file:///home/john/Blue/TestAutomation/temp/testOutput.html

Search

Simplified Results:

.....

Detailed Results:

Test ID: 0	
Requirement Tested: Requirement: log_time handles fractions of seconds correctly	Arguments: 0.1
Component Tested: LoggingTimer	Expected Result: * Time is: 0:0.1s
Function Tested: log_time	Pass/Fail: PASS
Test ID: 1	
Requirement Tested: Requirement: log_time formats multiple digits correctly	Arguments: 10
Component Tested: LoggingTimer	Expected Result: * Time is: 0:10.0s
Function Tested: log_time	Pass/Fail: PASS
Test ID: 2	
Requirement Tested: Requirement: log_time must display	Arguments: 10.1

Deliverable #4

For Deliverable 4 we were instructed to complete the design of our framework and test it with all 25 tests.

We located as many methods as we could that did not require a database and began writing tests on them.

The methods that we were able to test were:

LoggingTimer()

log()

log_nostar()

Deliverable #4

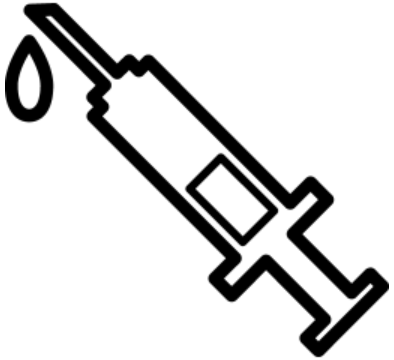
Our results after testing all 25 methods we that all tests passed.

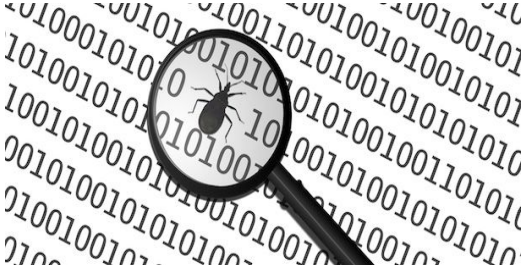
We did run into issues when we attempted to make a test fail

And our driver was not able to handle the failure.

Deliverable #5

Deliverable 5 had us inject faults into our program in order to simulate a third party changing our code. Our test cases we able to pick up these faults.





We were able to inject 5 separate faults into our code that cause some tests to fail.

Deliverable #5

Fault 1:

We deleted a star within the log method causing a formatting error.

```
58 def log(msg, *args, **kwargs):
59     #log_nostar("* " + msg, *args, **kwargs)
60     log_nostar(" " + msg, *args, **kwargs)
```

Results: all the tests that required the star to be within the format failed.

file:///home/john/Blue/TestAutomation/temp/testOutput.html

Simplified Results:
FFFFFFFFFFFFFFFFF.....

Detailed Results:

Test ID: 0	
Requirement Tested: Requirement: log_time handles fractions of seconds correctly	Arguments: 0.1
Component Tested: LoggingTimer	Expected Result: * Time is: 0:0.1s
Function Tested: log_time	Pass/Fail: FAIL
Test ID: 1	
Requirement Tested: Requirement: log_time formats multiple digits correctly	Arguments: 10
Component Tested: LoggingTimer	Expected Result: * Time is: 0:10.0s
Function Tested: log_time	Pass/Fail: FAIL
Test ID: 2	
Requirement Tested: Requirement: log_time must display	Arguments: 10.1

Deliverable #5

Fault 2:

We added an extra line to the method

```
62 def log_nostar(msg, *args, **kwargs):
63     sys.stdout.write(msg.format(*args, **kwargs))
64     #sys.stdout.write("\n")
65     sys.stdout.write("\n\n")
66     sys.stdout.flush()
```

Results: Caused all of our tests to fail because they all used the log_nostar() method.

file:///home/john/Blue/TestAutomation/temp/testOutput.html

Simplified Results:
FFFFFFFFFFFFFFFFFFFFFFFF

Detailed Results:

Test ID: 0	
Requirement Tested: Requirement: log_time handles fractions of seconds correctly	Arguments: 0.1
Component Tested: LoggingTimer	Expected Result: * Time is: 0:0.1s
Function Tested: log_time	Pass/Fail: FAIL
Test ID: 1	
Requirement Tested: Requirement: log_time formats multiple digits correctly	Arguments: 10
Component Tested: LoggingTimer	Expected Result: * Time is: 0:10.0s
Function Tested: log_time	Pass/Fail: FAIL
Test ID: 2	
Requirement Tested: Requirement: log_time must display	Arguments: 10.1

Deliverable #5

Fault 3:

Method: log_time()

Changed the division from 60 to divide by 1.

```
76      def log_time(self, msg, *args, **kwargs):
77          total_time = time.time() - self.start_time
78          #mins, secs = divmod(total_time, 60)
79          mins, secs = divmod(total_time, 1)
80          msg = msg.format(*args, **kwargs)
81          log("{}: {}:{:0.1f}s", msg, int(mins), secs)
82          #log("{}: {}:{:0.1f}", msg, int(mins), secs)
83          #log("{}: {}:{:0.1f}s", msg, int(mins), int(secs))
84          self.reset()
85
```

Results: Caused all the tests that relied on the seconds to be correct failed.

Simplified Results:
.FFFF.F.....

Detailed Results:

Test ID: 0	
Requirement Tested: Requirement: log_time handles fractions of seconds correctly	Arguments: 0.1
Component Tested: LoggingTimer	Expected Result: * Time is: 0:0.1s
Function Tested: log_time	Pass/Fail: PASS
Test ID: 1	
Requirement Tested: Requirement: log_time formats multiple digits correctly	Arguments: 10
Component Tested: LoggingTimer	Expected Result: * Time is: 0:10.0s
Function Tested: log_time	Pass/Fail: FAIL
Test ID: 2	
Requirement Tested: Requirement: log_time must display multiple digits and multiple units correctly	Arguments: 10.1

Deliverable #5

Fault 4:

Method: Log_time()

We altered the format in which Log_time() prints to stdout.

Results: This caused all the tests that used Log_time() to fail

```
76      def log_time(self, msg, *args, **kwargs):
77          total_time = time.time() - self.start_time
78          mins, secs = divmod(total_time, 60)
79          #mins, secs = divmod(total_time, 1)
80          msg = msg.format(*args, **kwargs)
81          #log("{}: {}:{:0.1f}s", msg, int(mins), secs)
82          log("{}: {}:{:0.1f}", msg, int(mins), secs)
83          #log("{}: {}:{:0.1f}s", msg, int(mins), int(secs))
84          self.reset()
85
```

file:///home/john/Blue/TestAutomation/temp/testOutput.html

Simplified Results:
FFFFFFF.....

Detailed Results:

Test ID: 0	
Requirement Tested: Requirement: log_time handles fractions of seconds correctly	Arguments: 0.1
Component Tested: LoggingTimer	Expected Result: * Time is: 0:0.1s
Function Tested: log_time	Pass/Fail: FAIL
Test ID: 1	
Requirement Tested: Requirement: log_time formats multiple digits correctly	Arguments: 10
Component Tested: LoggingTimer	Expected Result: * Time is: 0:10.0s
Function Tested: log_time	Pass/Fail: FAIL
Test ID: 2	
Requirement Tested: Requirement: log_time must display	Arguments: 10.1

Deliverable #5

Fault 5:

Method: Log_time()

Changed the Log_time() method to log seconds as Ints instead of Floats.

Results: This caused all the tests that had fractions of a seconds to fail.

```
76      def log_time(self, msg, *args, **kwargs):
77          total_time = time.time() - self.start_time
78          mins, secs = divmod(total_time, 60)
79          #mins, secs = divmod(total_time, 1)
80          msg = msg.format(*args, **kwargs)
81          #log("{}: {}:{:0.1f}s", msg, int(mins), secs)
82          #log("{}: {}:{:0.1f}", msg, int(mins), secs)
83          log("{}: {}:{:0.1f}s", msg, int(mins), int(secs))
84          self.reset()
```

file:///home/john/Blue/TestAutomation/temp/testOutput.html

Simplified Results:
F.F.FF.....

Detailed Results:

Test ID: 0	
Requirement Tested: Requirement: log_time handles fractions of seconds correctly	Arguments: 0.1
Component Tested: LoggingTimer	Expected Result: * Time is: 0:0.1s
Function Tested: log_time	Pass/Fail: FAIL
Test ID: 1	
Requirement Tested: Requirement: log_time formats multiple digits correctly	Arguments: 10
Component Tested: LoggingTimer	Expected Result: * Time is: 0:10.0s
Function Tested: log_time	Pass/Fail: PASS
Test ID: 2	
Requirement Tested: Requirement: log_time must display	Arguments: 10.1