Lorena Nacoste

Chris Durham

CS 362: Final Project Report

Manual Testing:

Using different possible inputs, the isValid() function was called to test for valid and invalid url inputs against a known valid authority: google with the intent of finding either an invalid return from what is known to be a valid url, or a valid return from what is known to be an invalid url.

Testing was first done to test the http:// protocol against urls with the expected results. From there ports were tested, then pathing was tested, then queries were finally tested. Unexpected results from URLS were:

www.google.com: was expected to be valid
http://www.google.com?action=view: was expected to be valid
http://www.google.com?action=view&hi=hello: was expected to be valid
http://www.google.com:3000: was expected to be valid
http://www.amazon.com/ file: was expected to be invalid

The full list of urls tested are found within the testManualTest() function.

Input Partitioning:

For partition testing, the input domain was separated into url parts which were then each tested. First scheme, authority, port, path, and queries.

The errors found were:

http://: was expected to be true
" ": was expected to be false
?action=view: was expected to be true
:3000: was expected to be true

Unit Testing:

We wrote our unit test as testIsValid() in URLValidatorTest.java. The test uses the testURL data from the original URLValidatorTest.java to build all the possible URL combinations.

However, it uses a different method for building URLs than the original testIsValid(). It involves one outer loop that iterates for the number of possible URLs. Then, it appends the first scheme to the first authority and continues this pattern for the port, path and query. As each part is appended, the expected test result of that URL part is stored. Once the end of a URL part list has been reached, the

index is reset and the pattern continues. After the entire URL has been built, it is sent as input to the isValid() function. The stored expected result is compared to the actual result.

If they don't match, a failure message is printed. After all the URLs have been tested, the number of failed tests is also printed. After execution of all the test URLs, 3108/35910 test failed. This translates to about 8.6% of all the tests failed. Though this may seem like a large number, the URLs that failed were all very similiar to each other. This lead the team to believe that the vast majority of the cases probably shared a common cause.

Collaboration:

Chris had no prior experience with Java. So, he and Lorena decided that he should work on manual testing and input partitioning while Lorena worked on the unit test. After all testing had been completed, we got together and compiled our bug reports and discussed the bug causes that we found. We completed the report together with each team member writing the sections of the report that pertained to the work that was completed by that individual. We collaborated well together and had no problems integrating our work.

Bug Report 1:

URLValidator fails when validating URLs that contain queries

Description: The Apache Commons classs org.apache.commons.validator method isValid() fails when validating URLs that contain queries, like "http://www.google.com:80/test1?action=view" or "h3t://255.com:80?action=view".

How to Reproduce/ How the Bug Was Found: These failures were found using the testIsValid() unit test that the team created. However, they can also be produced by manually giving these URLs and similar URLs as input to isValid(). These URLs produced a false result but the expected result is true.

Failure Cause: I suspect that the cause of this failure is line 446 in UrlValidator.java. The line reads "return !QUERY_PATTERN.matcher(query).matches()". This causes the function isValidQuery() to return an opposite result than intended. This bug cause was found using the Eclipse debugger. Breakpoints and print statements were inserted all around the UrlValidator.java file. We then determined exactly what lines were executed when a URL with a query was entered as input. we found the wrong result was being returned from isValidQuery.

Bug Report 2:

URLValidator fails when validating URLs that contain port numbers with more than three digits

Description: The Apache Commons classs org.apache.commons.validator method isValid() fails when validating URLs that contain port numbers with more than three digits, like "http://go.com:65535" and "http://go.com:65536".

How to Reproduce/ How the Bug Was Found: These failures were found using the testIsValid() unit test that the team created. However, they can also be produced by manually giving these URLs and similar URLs as input to isValid(). These URLs produced a false result but the expected result is true.

Failure Cause: I suspect that the cause of this failure is line 158 in URLValidator.java. The line reads "private static final String PORT_REGEX = "^:(\\d{1,3})$". This line says the port number should be one to three digits from 0 to 9. The latter part of this is correct. However, the port number should allow from one to five digits at least. Five digit port numbers, like 65535 and 65536, are allowed according to the test data given. Therefore, line 158 must be an error. This bug cause was found using the Eclipse debugger. Breakpoints and print statements were inserted all around the UrlValidator.java file. We then determined exactly what lines were executed when a URL with a port was entered as input. Then, we studied the lines for any errors and noticed the error in line 158.

Bug Report 3:

URLValidator fails when passing whitespaces in urls.

Description: The Apache Commons classs org.apache.commons.validator method isValid() fails when validating URLs that contain invalid whitespace such as 'http://www.google.com /'.

How to Reproduce/ How the Bug Was Found: These failures were found using the testManualTest() unit test that the team created. However, they can also be produced by manually giving these URLs and similar URLs as input to isValid(). These URLs produced a true result but the expected result is false.

Failure Cause: Line 377 of isValidAuthority removes all trailing whitespace as a means of separating the port. This is allowing non-valid urls to pass. This can be reproduced manually by entering 'http://www.google.com /' into isValid(). This bug was discovered while doing manual testing in the testManualTest() function.

Agan's Principles Used:

A few of Agan's principles were used when debugging URLValidator. Rule #2 ("Make IT Fail") was definitely used the most. Once we received a failing result from the unit test, we manually fed the failed URL back into isValid() and used the debugger to study the execution of the program when validating the URL (Rule #3: "Quit Thinking and Look"). Then we tried to determine which part of the URL caused the failure by studying the failing tests to see which URLs were similar (Rule #4: "Divide and Conquer"). Using the above principles, along with the other methods we learned in this class, allowed us to find bugs and determine their causes efficiently.