Generation of String Test Input from Web using Regular Expression

Sneha Shelke
Department of Computer Engineering
K J Somaiya College of Engineering
Mumbai University
Mumbai, India

Sangeeta Nagpure
Department of Computer Engineering
K J Somaiya College of Engineering
Mumbai University
Mumbai, India

ABSTRACT
To generate realistic test data is a challenging for software testers. Realistic test data generation for certain input type is hard to automate and therefore laborious. So in this paper, string inputs will be used from the internet by performing web queries based on the key identifiers appearing in the source code of the program under test. The resultant URLs will be generated, tokenised and the collected regular expressions are matched with the text and get valid string values.

General Terms
Software testing, Web query, Web pages

Keywords
String inputs, Valid inputs, Web queries, Regular expressions, Web pages

1. INTRODUCTION
Automatic generation of test cases is an integral part of the software automation testing system [8].There are many approaches including dynamic symbolic execution and search based testing [7]. One important activity in test data generation involves deriving suitable values for string inputs. Strings present additional challenges for basic data types, such as integers [4]. In general, strings may be of variable length, contributing to enormous input domain sizes, and consequently very large search spaces. Many forms of real world data may be naturally represented as strings; for example resource locators, dates of different localised formats, international banking codes, and national identity numbers.

Until now an issue is concerning the problem of generating realistic values that tester likely to generate and use in the program. Test generators can perform action from program code only and generate any value rather than natural and readable values. Internet web pages are a rich source of examples of string data that may be re-used as a natural source of inputs for a wide variety of programs. Here string inputs can be brought from the internet by performing web queries based on the key identifiers appearing in the source code of the program under test. The resultant URLs will be generated, tokenised and the collected regular expressions are matched with the text and get valid string values. The paper presents 5 java classes from 5 open source projects. The code studied validation for a number of different data types based on the strings. The web query was capable of retrieving valid, well formed string inputs.

The contribution of this paper is as follows:

1) The study shows that valid, well formed string inputs can be found for the program by formulating program identifiers into web queries.
2) This approach also uses regular expression to identify valid values.

2. APPROACH

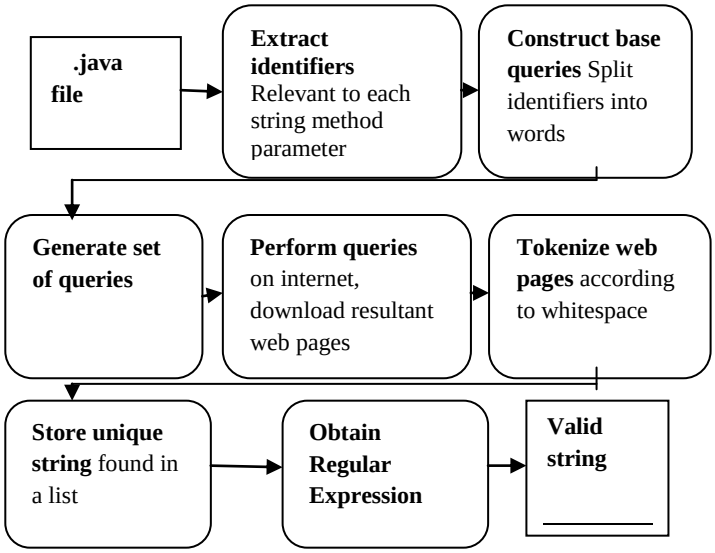

Fig 1: Overview of the approach

Figure 1 presents the overview of the approach that takes a .JAVA program source as input. First the information about the program identifiers is extracted from the source code. These identifiers are splits into words to construct base queries and then generate web queries. The queries are then sent to a search engine. Finally, the textual contents of the URLs produced in the search results are downloaded, from which the values are identified using regular expressions obtained previously. The rest of the section provides details for each part of the approach.

2.1 Extracting Identifiers
Web queries are generated by using key identifiers found in the program under test [10]. Program identifier describes different types of the data. For example, an identifier name is "email" which indicates the value to be an email address. So by performing web query email gives, email addresses from the web. Here web queries are performed using three program identifiers related to the string method parameter for which input values are sought:

2.1.1 The string method parameter identifier. Anticipated values are brought from the name of the method parameter which is recognizable source of potential information about types of values that will be used for. [6]

2.1.2 The method identifier. Sometimes programmers use non-informative generic method parameter names, such as text, string, or value. This may be because the concept is already embodied in the method name involving the string parameter. [6]

2.1.3 The class identifier. The method name may also have a non-informative generic name if the class has few responsibilities other than to represent a data type itself and the operations that may be performed on it. In this case, the name of the parameter's class may be a valuable source of keywords. [6]

2.2 Generate Web Query

There are some steps to generate web queries. The first step is to generate base query. Program identifiers are concatenations of terms which need to be separated back into individual words to form useful web queries. The base query is formed by splitting identifiers into words. For example identifier "EmailAddress" is separate into email address.

Additional web queries can be obtained by applying combination of operations, to get valid string inputs. [6]

These are as follows:

2.2.1 Pluralisation: The base query is pluralised by pluralising the last word. For example, email address becomes email addresses.

2.2..2 Prefixing: In order to direct the web search towards pages containing lists of examples for the identifier, versions of the base query are formed by prefixing the query with 'list of'; for example 'list of email addresses'.

2.2.3 Quoting: Quoting words in a web query signals to the search engine that those terms must be found as a complete phrase in the web pages to be retrieved. This is useful to ensure that identifier words are kept together in the web pages returned in the search engine's results [6].

email address (base query)
email addresses (pluralisation)
list of email address (prefixing)
"email address" (quoting)
"email addresses"
"list of email address"

Fig 2: Queries generated for the identifier 'email address'

2.3 Performing Web Queries and Processing Resultant Web Pages

Queries are performed using the internet search engine to retrieve the results. The localisation was set to 'en-GB', with URL results of a non-HTML content type to be ignored. The API limits the results to the first 10 web pages for each query. [6] The first 10 URLs returned in the search results are then downloaded and parsed using the JSoup parser [9].

2.4 Identifying Values

Finally the regular expression and downloaded web pages identifies valid values. The query results are processed by first downloading the contents of each URL and stripping out HTML tags and regular expressions are matched the remaining text and all unique matches are identified as potential string values.

3. RESULT

3.1 Extract identifiers: Giving .java file as input and identifiers are extracted from the source code. The resulting figure no.2 is appended below.

3.2 Generating base query: Here, the identifiers extracted from the source code and are split into words. The resulting figure no. 3 is appended below.

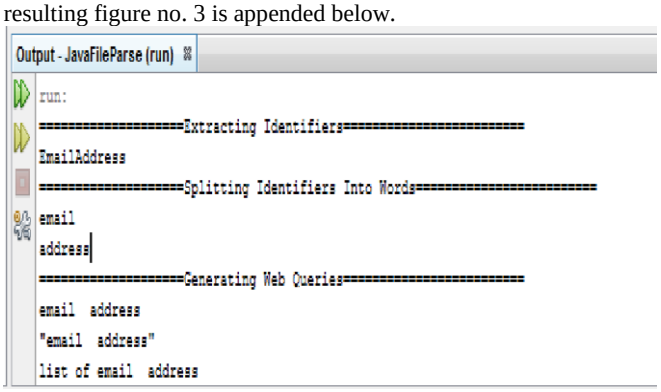

Fig 3: Extracting Identifiers and generating base query

3.3 Generating Web Queries: Operations such as prefixing, quoting, and pluralisation are applied to generate base queries. The resulting figure no.4 is appended below

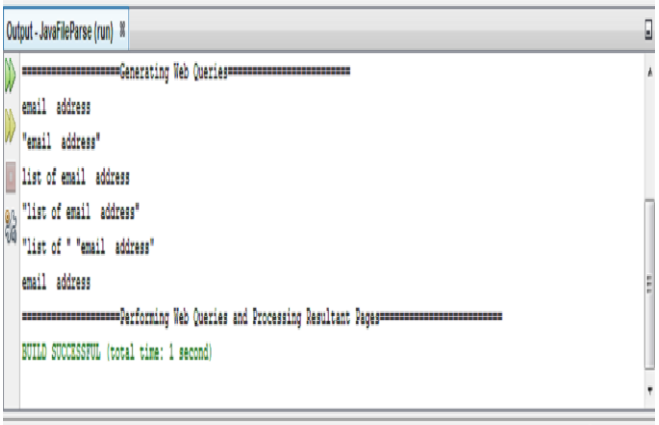

Fig 4: Geneating Base Queries

3.4 Performing web queries

Queries are performed using the internet search engine to retrieve the results. The query results are processed by first downloading the contents of each URL and stripping out HTML tags. The resulting figure no.5 is appended below.

Fig 5: Performing Web Queries

3.5 Identifying Values:

The query results are processed by first downloading the contents of each URL and stripping out HTML tags and regular expressions are matched the remaining text and all unique matches are identified as potential string values.The resulting figure no.6 is appended below.

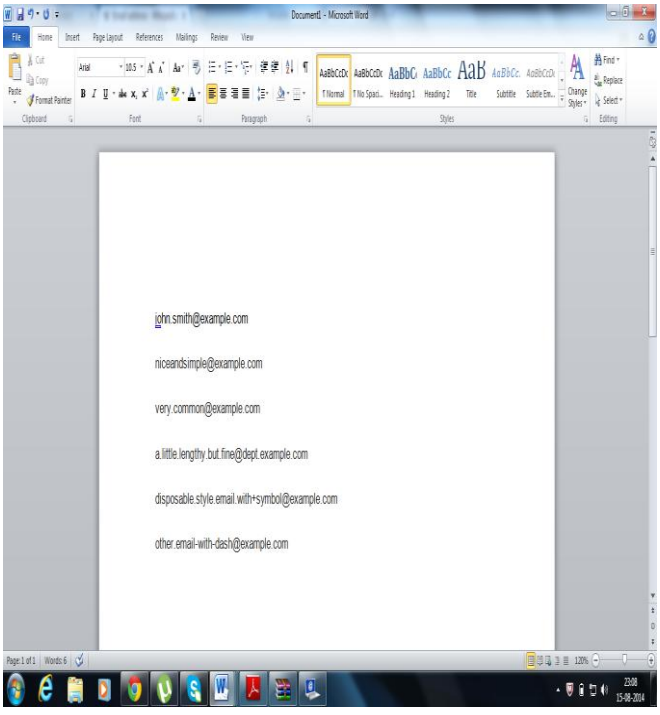

Fig 6: Valid String

4. CONCLUSION

This paper's approach generates valid values for string data types using web searches and regular expressions. And generated values are realistic which human tester can understand easily and can be used in automatic test case generation technique [11]. Here string inputs are extracted from the internet by performing web queries based on the key identifiers appearing in the source code of the program under test. The resultant URLs are downloaded and tokenised and from that we get unique strings.

These valid values can be used as a fitness function in SBT [12] in evolutionary searches that reduce the time [6].

5. ACKNOWLEDGMENTS

This paper is made possible through the help and support from everyone, including: parents, teachers, family, friends, and in essence, all sentient beings. Especially, please allow to dedicate acknowledgment of gratitude toward the following significant advisors and contributors:

First and foremost, would like to thank Sangeeta Nagpure for her most support and encouragement. She kindly read paper and offered invaluable detailed advices on grammar, organization, and the theme of the paper.

6. REFERENCES

[1] Phil McMinn, "Search-Based Software Testing: Past, Present and Future", University of Sheffield, Department of Computer Science Regent Court, 211 Portobello, Sheffield, S1 4DP, UK

[2] P. McMinn, "Search-based software test data generation: A survey," Software Testing, Verification and Reliability, vol. 14, no. 2, pp. 105–156, 2004.

[3] M. Bozkurt and M. Harman, "Automatically generating realistic test input from web services," in International Symposium on Service-Oriented System Engineering, 2011.

[4] M. Alshraideh and L. Bottaci, "Search-based software test data generation for string data using program-specific search operators," Software Testing, Verification and Reliability, pp. 175–203, 2006.

[5] Mark Harman, Phil McMinn, Jereson Teixeira de Souza, and Shin Yoo, " Search Based Software Engineering: Techniques, Taxonomy, Tutorial

[6] P. McMinn, M. Shahbaz, and M. Stevenson. "Search-based test input generation for string data types using the results of web queries". In ICST, pages 141–150, 2012.

[7] Software testing. http://en.wikipedia.org/wiki/Software_testing

[8] Automation. http://en.wikipedia.org/wiki/Test_automation

[9] JSoup. http://www.jsoup.org.

[10] S. Butler, M. Wermelinger, Y. Yu, and H. Sharp. Improving the tokenisation of identifier names. In ECOOP, 2011.

[11] Aditya P. Mathur. Foundations of Software Testing. Addison-Wesley Professional, 1st edition, 2008.

[12] M. Harman and P. McMinn. A theoretical and empirical study of search-based testing: Local, global and hybrid search. IEEE Transactions on Software Engineering, 36:226–247, 2010.