International Journal of Computer and Communication Technology

Volume 6 | Issue 2 Article 12

April 2015

AUTOMATION OF MEDIA ACCESS CROSS CONNECT

GIRIDHAR TRIVEDI
Department of Information Science and Engineering M S Ramaiah Institute of Technology, Bangalore, INDIA, giritrivedi@gmail.com

AUTOMATION OF MEDIA ACCESS CROSS CONNECT

GIRIDHAR

Department of Information Science and Engineering
M S Ramaiah Institute of Technology, Bangalore, INDIA
Email:giritrivedi@gmail.com

Abstract:-Sanity testing is the process of testing the basic functionality of any software product. Sanity testing would take considerable amount of time by test engineer to test the product. If sanity testing itself fails, then the effort put on other testing techniques would go waste, as the build is declared un-usable if Sanity fails. Hence it is a good idea to automate sanity testing process so that, same sanity suite can be run as soon as a new build arrives. This automation tool is developed for a specific product, Media Access Cross Connect. The tool is being divided into two parts. First part, records all test cases in the background, whenever a test engineer performs sanity testing and saves it in XML file format. Second part, replays same configurations using a script and carries out verification and validation for the response data to decide success or failure of the test cases. It is record once run many times activity.

Keywords:- *XML, PERL, WebCT, JSON, url, HTML, DOM, GUI.*

I. INTRODUCTION:

Test automation tools have changed the way, testing is performed in this era. Testing is an important part of software development life cycle. If problems come up during implementation phase, delivery time of the product slides.This will add more pressure on testers, leading to inefficient testing or delaying the product release or more resources need to be allocated to testing team to meet aggressive deadlines [7]. Automation tools will reduce this time required for testing. Advantage of this automation tool is two fold. First, test engineers need not have to re-create failed test case scenarios as they would have already present in XML file. Second, developers can refer to saved XML file to find out, locality of occurrence of problems without contacting test engineers. One can rely to certain extent, on automation tools to perform testing even if the delivery time of the product slides. Our automation tool intends to help perform sanity testing. This tool belongs to the category of record and replay automation tools. This tool is being divided into two parts; first part captures all configurations in background, done by test engineer. Second part replays configurations captured in XML file. Verification and validation is carried out for response data. If result shows as successful then, execution of the test case is considered to be passed, else it is considered to be failed. Verification and validation is done to ensure that, software products can satisfy all requirements and certain performance [2].

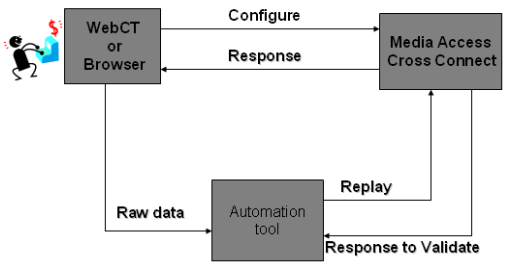


Fig 1: Automation Tool

Block diagram of automation tool is as shown in fig1. WebCT or Browser is used to configure the server. Media Access Cross Connect is the product for which this automation tool for sanity testing is developed. Automation tool is the tool under development used to automate sanity testing for the above said product.

User or test engineer configures Media Access Cross connect using either webCT or a browser. Media access cross connect takes all configurations and functions accordingly. The configurations done can be seen on the browser or on WebCT. Capturing part of automation tool, captures all these configurations done by test engineer. These configurations are saved in XML file. Structure of XML file is as shown in table 1. Replay part of automation tool takes this XML file as reference and replays all configurations. After completion of configuration, server will send response in JSON format. This JSON response is decoded by PERL packages, then verification and validation for all parameters present in the XML file is done.

II. RELATED WORK

Ben-menachem M and Marless G.S in [6] presents, a minor defective component can cause major adverse effects if the developed software is not thoroughly tested prior to its real-world implementation. Hence the tester has to test all components within the limited time under the pressure of delivery dates. Automating testing mechanism helps reduce time required for testing and improve the quality of product. According to [4] there are several types of test automation, namely, Test management, Unit test, Test data generation, performance test, functional/system/ regression test. Popular test automation category is “Test Execution” automation technology also known as capture/playback or record/playback. [5] calls software testing as the indispensable phase in modern soft-

ware life cycle. Software testing is done to ensure that the developed software is useful and can run properly in real world applications.
Automation is considered to be, the future trend of testing. The current practice of software test automation is based on recording manual test activities and replaying recorded test scripts for sanity and regression testing. It is imperative to reduce cost and improve effectiveness of software testing by automation testing process which contains many testing related activities using various techniques and methods [3]. Software automation can improve the productivity and quality of work to a great extent [2].

```
<root>
<serverType>
<REQ>
<POST> URL</POST>
<Param1>name=value</Param1>
<Param2>name = value </ Param2>
<Param3>name = value <Param3>
<Param4> name = value </ Param4>
.
.
<TMDLY>time</TMDLY>
<STEPID>id</STEPID>
</REQ>
.
.
</root>
```

Table 1 : XML file Format

III. RECORDING TOOL

Recording tool is implemented in java script. This tool will be used as an add-on to either Mozilla Firefox or Google chrome browser. As on today recording tool works fine with all the versions of Firefox browsers and it works only on one version of Google chrome browser. This tool takes help of an open source tool's source code just to capture POST methods generated, whenever test engineer tries to configure the server either through webCT or through a browser.

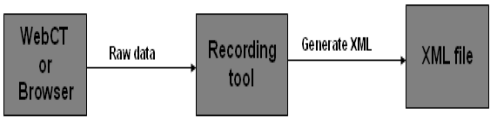


Fig 2: Recording Tool

Block diagram of recording tool is shown in fig 2. Product for which this tool is developed, contains different names on the GUI than that of corresponding names present in JSON response present on sever. Once all configurations have been done, test engineer clicks on save button, a POST method will be generated. This POST method will be saved and underlying DOM structure of underlying page is recursively iterated upwards to find the "form" tag of page. This form tag will give us underlying HTML form name. This HTML form will be opened and searched, to find out JSON parameter names present on GUI. This form tag contains a return block inside it. Return block contains mapping for fields present on form and corresponding JSON response. We will extract whole block at a time, search only for parameters present in POST method for verification and validation. All these validation and verification parameters along with POST method will be put in "REQ" block of XML file. In addition to all these, a TMDLY tag representing time delay will also be inserted into the "REQ" block. The tag STEPTYPE indicates, type of testing is being performed. Likewise for every configuration done, a separate "REQ" block will be generated. This XML file acts as a test case. Collections of such XML files form a Test Suite.
The pseudo code used to iterate through the DOM structure of the file is as shown in Algorithm 1.

```
WHILE (parentNode != Null)
   Obtain element's parent node.
   IF parent'snode = FORM
         Extract innerHTML of parent 2  level
         above
         Extract "return" block
        FOR(each line in return"Block)
            Separate JSON and form-
            name
         ENDFOR
         return JSON and form name
   ENDIF
ENDWHILE
```

Algorithm1: Parse JSON parameters

For read-only parameters present on the GUI, the same procedure will be repeated to extract the underlying JSON parameters except the upward iteration would be done till the "div" tag is found.

IV. PLAYBACK TOOL

Playback tool is implemented in PERL. PERL script takes captured XML file as a reference to replay all configurations present in "REQ" block. XML file will have to be parsed so that, we will have well defined data structures present in the PERL. This parsing helps us to play around the converted data for verification and validation at a later point in time.

The high level working of this playback tool is as follows. PERL script takes XML file as input. Reads each "REQ" block present in XML file. This "REQ" block contains all information pertaining to configurations done and various parameters to be verified and validated. These confi-

gurations will be replayed. Server will return configured data or response, in JSON format. This JSON data is again decoded into PERL data structures using available JSON decoders in PERL. The block diagram of the playback tool is shown in fig 3.

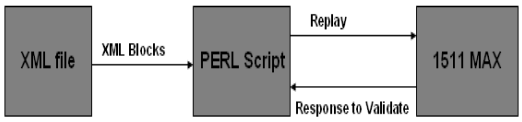


Fig 3: Replay Tool

We make use of XML parsers to parse the XML files, which contain test cases. There are basically two types of XML parsers, SAX parser and BNF parser [1]. PERL packages are used to parse this XML file into PERL hashes. The package used is XML::Simple. This package will help to parse XML file, returns a reference to converted PERL hashes. Data::Dumper package, available in PERL is used to dump converted hashes into some file or on to standard output. Returned reference from parser will be pointing to one among, hash, array of hashes or to scalars. We need to find out to which data structure the reference is pointing to. That can be done using following algorithm 2

```
FOR (eachkey in HASH )
        obtain the reference pointing to
      that key
        IF ref = HASH
                It is a hash
        ELSE IF ref = ARRAY
                It is an array
        ELSE
                It is a scalar
        ENDIF
ENDFOR
```

Algorithm 2 : To Parse JSON Response.

Once XML file is parsed, each block in the XML file will be considered as a single test step. This will be replayed using LWP::UserAgent package of PERL. There are two methods available in this package , GET and POST. This will return reference to response. When this reference is made to point to "contents" field, actual response will be available. Simplest use of this method is as shown in table 2.

```
$response = $useragent->get($URL);
$var = $response->content;
  or
$response = $useragent->post($URL);
$var = $response->content;
```

Table 2 : UserAgent Package Usage

$useragent is new instance of LWP::UserAgent package. This has to be created first, before performing above actions. The response obtained like this will be in the form of JSON. This response has to be decoded for further verification and validation of parameters present in the "REQ" block of XML file.

Response in JSON format will be decoded using JSON::PP package. Procedure to decode is as in table 3.

```
$str = $json->allow_barkey->decode(

$fileContents)
```

Table 3 : Decode JSON Response

$json is the new instance created for that package. $str contains reference to the response. Decode method will decode the response into definite PERL data structures. Reference would be pointing to scalar, hash, array, array of hashes, hash of hashes or any other combination available. This response is compared against the key-value pair of verification and validation parameters present in each REQ block of XML file. If all key-value pairs found to be present in response, then that test step is considered to be passed. If any one key-value pair is not present, then test step is considered to be failed. If any of the test steps present in XML file is said to be failed, then whole test case is considered to be a failed test case.

V. PERFORMANCE EVALUATION

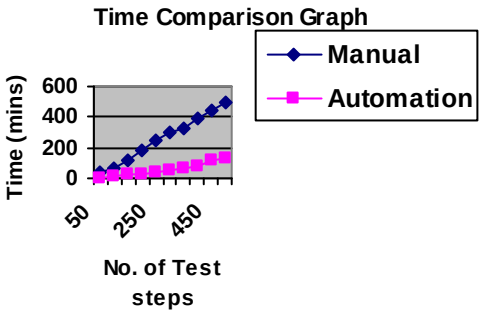
The configuration on which the tool is being used or tested contains a computer system with minimum of 512MB RAM, 40 GB Hard Disk, Media Access Cross Connect, WebCT or Browser, PERL Interpreter.

"Time" is one parameter which can be used for evaluating the performance of the automation tool. Here we compare the time and effort taken by the test engineer to execute certain number of test steps against the time taken by the automation tool. It is found that as the number of test steps increases, the time taken for manual testing also keeps increasing while the time taken by automation tool is far less than of the manual time. It is shown in graph 1.

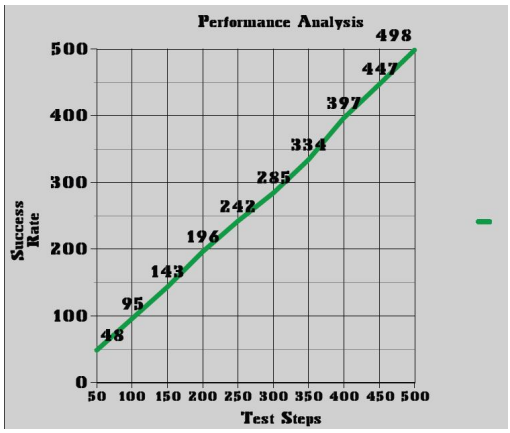
"Success Rate" is another parameter used to check stability of tool. 0 to 4 test steps in every 100 test steps are failing as on today. Once this tool is developed completely, we are expecting all the test steps to pass through. The graph of "success rate" is as shown in graph 2.

VI. CONCLUSION

Sanity testing process can be well automated to save time of a test engineer, which can be used for other useful purpose. This will set us the benchmark to automate other testing techniques too. Automation tool under development works fine with the product for which it is developed. Currently the time required for testing by one man week hour is being reduced to only two hours. The technology used in this tool can also be used for other products with little modification, which suits for the GUI structure designed for the product. The tool is in the testing phase, once testing is completed the enhanced version of the tool is released with improved "success rate".



Graph 1



Graph : 2

REFERENCES:

[1]. Akhil Rangan C K, Jayanthi J,2011," A Generic Parser to Parse and Reconfigure XML files", Recent Advances in Intelligent Computational Systems (RAICS), IEEE, Page(s) 823-827.

[2]. Liu Xialoi, 2009, "Process oriented analysis for software automation", Education Technology and Computer Science, First International Workshop on IEEE. Page(s) 1131-1133

[3]. Hong zhu, W.Eric Wong, Amit paradkar, 2007, "Automation of software test report", 29th international conference on software engineering. Page(s) 150-151.

[4].Tom Wissink and Carlos Amaro, 2006, "Successful test automation for software maintenance", 22nd IEEE International Conference on software maintenance. Page(s) 265-266.

[5]. Burnstein. I, 2003, "Practical software testing. A process-oriented approach", Springer.

[6]. Ben-menachem . m and Marless G.S, 1997, "Software quality: Producing practical, constituent software, slaying the software dragon series", International Thomson computer press, Boston.

[7]. Alex Cervantes,2009, "Exploring the Use of a Test Automation Framework", , IEEEAC paper #1477, Version 2, Updated January 9, 2009, Page(s) 1-9.

[7]. Alex Cervantes,2009, "Exploring the Use of a Test Automation Framework", , IEEEAC paper #1477, Version 2, Updated January 9, 2009, Page(s) 1-9.

[8]. Galin, D., 2004, "Software Quality Assurance: From Theory to Implementation", Pearson/Addison Wesley.

[9]. Zhenghong Gao, Yinfei Pan, Ying Zhang, Kenneth Chiu, 2007, "A High Performance Schema-Specific XML Parser", Third IEEE International Conference on e-Science and Grid Computing, Page(s) 245-252.

[10]. Horch, J. W., 2003, "Practical Guide to Software Quality Management", Second Edition, Artech House Publishers, Nonvood, MA.

[11]. Noah S.Prywes, 1974, "Automatic generation of software systems", ACM SIGMIS Database, 6(2), New York, Page(s) 7-17.

[12]. Liu Xiaoli, 2009, "Process Oriented Analysis for Software Automation", First International Workshop on Education Technology and Computer Science, Page(s) 1131-1133.

[13]. Zhou Yanming, Qu Mingbin, 2006, "A Run-time Adaptive and Code-size Efficient XML Parser", Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06), IEEE, Page(s) 18-21.

◈◈◈