Comparison of Web Service Testing Tools

Prabhjot Bal
MTech CSE Student, LPU, Punjab, India

Dalwinder Singh
Assistant Professor, CSE, LPU, Punjab, India

Dr. R. C. Gangwar
Associate Professor, CSE, BCET, Punjab, India

DO: 10.23956/ijarcsse/SV7I5/0301

Abstract—Web Services are considered an essential services-oriented technology today on networked application architectures due to their language and platform-independence. Their language and platform independence also brings difficulties in testing them especially in an automated manner. In this paper, a comparative evaluation of testing techniques based on TTCN-3 and SoapUI, in order of contributing towards resolving these difficulties is performed. Aspects of TTCN-3 and SoapUI are highlighted, including test abstraction, performance efficiency and powerful matching mechanisms in TTCN-3 that allow a separation between behaviour and the conditions governing behaviour.

Keywords—Web Services Testing, Automated Testing, Web Testing, SoapUI, TTCN-3, Titan TTCN-3, Testing

I. INTRODUCTION

Software testing is the process of validating and verifying a program or application. It involves checking for the correct functionality of the system. It also confirms that a project meets the technical and business requirements as expected. The testing procedure includes measurement and comparison of the expected outcome with the actual outcome, which helps in detecting the errors which might occur during the analysis, design and development of the project. Software Testing can be classified into the following forms [2]:

- Black Box Testing and White Box Testing.
- Static Testing and Dynamic Testing.

Black Box is a type of testing in which the test developer and executor need not have any knowledge about the internal workings of the software under test. By using automated testing the testing is done with the help of automated tools and test scripts (test suites). This technique leads to extensive tests of the software quality in completeness. Automated testing is less laborious, more reliable and faster to execute. There is a test suite development phase followed by a test system execution and subsequent results analysis phases which are usually automated also at least in partial. Dynamic testing is used to test software once it is in execution. The software is tested for operating as expected and satisfactorily working according to specifications. The actual results from the system under test are compared with the expected results in order to identify and resolve problems in the software under test. The web service testing under consideration in this paper involves black box, automated and dynamic testing of web services.

The language and platform independent nature of Web Services brings difficulties in testing their functional as well as non-functional behavior using a uniform set of automated testing tools. Until some time ago the major communication protocol used was Simple Object Access Protocol (SOAP) being mainly XML over HTTP. The data exchanged with web services followed precise formatting rules in the form of XML Document Type Definitions (DTD) or more recently the proposed XML Schema. More recently after its introduction, REST which stands for Representational State Transfer, is an architectural style for networked hypermedia applications, primarily used to build Web Services that are lightweight, maintainable, and scalable. A web service based on REST is called a RESTful service. REST has become one of the most important technologies for developing Web Applications. Its importance is likely to continue growing quickly as all technologies move towards an API centric orientation.

Every major development platform now includes frameworks for building SOAP and RESTful Web services. REST however is not dependent on any protocol, but almost every RESTful service uses HTTP as its underlying protocol. Recent times have seen the advent of SOAP as well as REST Web Service testing technologies which evaluate functionality and load handling among other aspects to measure how a Web Service performs for single clients and scales as the number of clients accessing it increases. This paper applies to such automated testing of Web services by usage of the Testing and Test Control Notation (TTCN-3), as well as SoapUI, software testing open source technologies and tools.

This paper concentrates on the various technology aspects for the development and execution of automation test suites for REST based Web Services with Titan TTCN-3 and SoapUI.

II. TEST TOOLS USED: A BRIEF OVERVIEW

TTCN-3 [1] is a standardized test specification and test implementation language developed based on the experiences from previous TTCN versions used at the European Telecommunication Standardization Institute (ETSI). It's applicable
for all kinds of black-box testing for reactive and distributed systems, e.g. Telecom systems (ISDN, ATM); Mobile (Telecom) Systems (GSM, UMTS); Internet (has been applied to IPv6); CORBA based systems. The TTCN-3 technology has built in constructs for, Configuration: dynamic and concurrent test configuration with test components (parallel and serial), Communication: various communication mechanisms (synchronous and asynchronous, message based and procedure based), Control: test case execution and selection mechanisms. The most important concept used is Black-Box Testing with TTCN-3, wherein a Test System (TS) comprising of a collection of Test Cases (TC) and Test Components (TCs) executes against a System Under Test (SUT) or an Implementation Under Test (IUT). The Test System is composed of Test Components (Serial and Parallel ones) along-with an interconnected network of Communication Ports. A particular arrangement of these components in a Test System is referred to as a Test Configuration. The Test Components and Communication Ports support an extensive set of possible Test Configurations suitable for constructing any possible Test System. Each Test Component has its own local verdict (state representing testing progress status), which can be set and read during Test executions. Titan TTCN-3 is a toolset developed and maintained by Ericsson which has been released into the open source domain recently.

SoapUI is a leading open source cross-platform API Testing tool. It enables Test System developers to execute automated functional, regression, compliance, and load tests on different kinds of Web based API’s. It supports all the standard protocols and technologies to test all kinds of Web based API’s. Its interface is simple and it enables both technical and non-technical users to use it productively and seamlessly. SoapUI is not only a functional API testing tool but also lets one perform non-functional testing such as performance and security tests on different kinds of Web API’s. It has built in functions to perform Security Testing with the capability to perform a complete set of vulnerability scans against Web API’s. It pro-actively prevents SQL Injection to secure the databases relied on by Web Applications. It scans for stack overflows, cross site scripting, while performing fuzzy and boundary scans to detect and avoid erratic behaviour of the Web Services under test. It has built in support for Load Testing by distributing the Load Tests across multiple agent components. It can simulate high volume and real-world load testing with ease allowing advanced custom reporting to capture various performance parameters for end-to-end System Performance Monitoring. SoapUI has a most comprehensive Protocol Support for Web API’s with built in support for, HTTP, WSDL, SOAP, REST, JDBC, and JMS etc.

III. COMPARISON OF SELECTED TESTING TOOLS

This section introduces the comparison of the two open source web service testing tools along with their observed results. The comparison can aide the practicing test developer to determine the efficiency of the tools and to select the suitable test tool for their test project needs [3]. A simple RESTful User Management web service is used to compare the two test tools which was developed using JAX-RS facilities in Java. A common interface for the User Management service was used by test suites built in both the tools as the system under test.

System Requirements All the test cases were run on a student pc with Intel Core i3 processor and 4GB RAM, Microsoft Windows 7 Home Premium, and 2Mbps Internet connection (though no internet requirements for the test system existed as SUT was on same local system). This comparison is made between the tools with the testing of the same web service i.e. the User Management service. Testing of the tools requires configuration which in turn includes installation of Titan TTCN-3 and SoapUI, setting up the test environment, data collection, selection of parameters and analytical survey. The sample web service used i.e. User Management System is tested with these respective Titan TTCN-3 and SoapUI configured tools.

Methodology Followed

Test Automation using the TTCN-3 development and execution techniques is used to make the Web Services testing process efficient. The main aim has been to present methods of developing an extensive and rich test suite to enable the tester to find out maximum number of faults in the least amount of time i.e. as early in the software development life cycle as possible. At the same time the testing budget has to be considered also as organizations attempt to strike a balance between the extent of testing on products and the quality of those products. Hence the attempt to illustrate a method of testing Web Services using TTCN-3. This leads to using newer techniques which are more efficient as compared the other alternatives (such as SoapUI). The methodology that is followed for this comparison of Web Service testing techniques is concisely elaborated in the following stages:

Developing the Research Problem: As stated earlier the objective is to compare and determine the effectiveness of using TTCN-3 over SoapUI for functional and performance testing of Web Services. The clear intention is to explore gains in efficiency with test development cycles, test coverage, and simplification of the test framework, leading towards a more effective Web Services testing process. It was this paper’s direction to use open source tools, TTCN-3 (Titan) and SoapUI, in a test development environment setup consisting of a common platform on a personal PC.

The Basic Research Execution Plan: The test comparison approach followed was to develop a common Systems Under Test. It started by using certain Systems Under Test (target set of Web Services) on cloud based platforms to emulate Web Services on Web Servers in a real world environment. In order to perform equivalent tests on the system under test from two separate test environments such as those of TTCN-3 and SoapUI both the toolsets were installed on the common platform which was Windows 7 [9]. The Test Systems (both for TTCN-3 and SoapUI based) were setup on the local student PC with TCP/IP access to the Tomcat Web Server ports where the RESTful services were hosted. The test scenario case studies were designed and developed first, followed by the coding of test suites using both the client tools being used for comparison, Titan TTCN-3 and SoapUI.
Collecting Relevant Test Reports: Post the development of Web Service system under test case studies and subsequent coding of test cases based on them, and upon their subsequent execution in the respective tool environments there was an earnest attempt to capture execution parameters in the test reports obtained, relevant to test coverage, performance parameters etc. Primarily data was obtained about individual test case and overall test suite execution times in terms of milliseconds spent during execution of similar test systems using both toolsets.

Analysis of Comparative Test Results: In this stage the activity was to perform a comparative evaluation of development cycle results and execution cycle results for both the client tools. The development cycle comparisons would be based on theoretical models of man day efforts and lines of code among other parameters, while the execution parameters would be compared on more granular terms based on machine performance cycles and resources consumed by test execution.

Comparison Summary and Interpretation: In this stage there was an attempt to be impartial in efforts to get and report accurate measures from the previous steps. But since TTCN-3 toolset from Titan is expected to give better runtime results [6] it puts it in good light against SoapUI as an alternative test system development environment. Reporting on these matters was to be followed by usage of simple comparison models for comparing development cycle time and costs for the developed test suites using both the client tools.

IV. RESULTS AND DISCUSSION

In the previous studies it has been reported that, different tools and techniques are utilized to provide reliable Web Service testing and hence quality of functionality and performance. In light of shorter test development cycles due to time to market limitations it becomes very important to utilize tools that find faults in the shortest period of time. The higher the number of faults detected in unit time the greater the quality of the product to market. It is very useful for a test developer and executor to be able to use a rich toolset for performing the required Web Services testing.

Typical Web Service testing applications spend roughly 80% of the code on parsing and formulating of test messages and responses [4]. In TTCN-3 much of the constructs (adapters and codecs) are reusable thus reducing this coding effort, while separating out the 20% of the code that is test logic making tests easier to maintain and understand. In this research work the attempt is to display this effectiveness of TTCN-3 by comparison with a well-established open source testing tool, SoapUI, used currently in the industry for Web Service-Oriented testing.

![Figure 1: Test Development Efforts Distribution](image)

The first set of experiments performed after the development of a common test suite in both Titan TTCN-3 and SoapUI against a common User Management RESTful Web Service (System Under Test) [5], was to perform a comparison of the test execution performance using both the toolsets. In order to collect the execution performance timings of test cases the following method was used, where T denotes the time taken:

\[ T_i (<\text{test case}>) = \text{Time taken for executing the } i^{th} \text{ run of } <\text{test case}> \]

The \( T_i \) value indicates the time taken by the \( i^{th} \) run of the test case being considered. For each test case executed against the system under test a record of the end time timestamp and the start time timestamp for the test case is made from the execution logs of the respective toolset used and using the different between the end time and the start time the time taken for executing each run i.e. \( T_i (<\text{test case}>) \) is derived.

\[ T_i (<\text{test case}>) = \text{End Timestamp of } T_i - \text{Start Timestamp of } T_i \]

In order to get a measure of the average execution performance of a test case with a particular toolset the following method was used to arrive at the average test case execution timings.

\[ \bar{T}_{avg} (<\text{test case}>) = \frac{\text{Sum} (T_1 (<\text{test case}>) , T_2 (<\text{test case}>) , T_3 (<\text{test case}>) , T_4 (<\text{test case}>) , T_5 (<\text{test case}>) )}{5} \]

As seen above the average is arrived at in this case by taking into consideration five test execution runs of each test case set using both the tools. The \( T_i \) values are added together and divided by the total test runs performed for the test case which was usually five to arrive at average timings.
In the following figures images of the execution logs of both the toolsets are displayed in order to clarify the collection method for timestamp data related to the execution of test cases using both the toolsets. Also in the following paragraphs the execution performance results are presented for both toolsets for a common test cases developed on both. The first test case is a simple ‘GetAllOptions’ call on the User Management RESTful Web Service for getting the list of all methods (‘OPTIONS’) supported by the same. It is clearly noticed that the Titan TTCN-3 test case executes faster than the SoapUI test case in all the test runs executed.

In the above image one can notice the test case execution duration as being 178ms for the GetAllOptions() test case executed with SoapUI interface. These timings are reported in the bottom right window panel of the interface with log records related to the test executions in SoapUI.

In the figure above and below are shown the test execution logs view of the Titan TTCN-3 tool, both the summary view and the detailed logs view. The execution time in this case for the GetAllOptions() test case is 25ms. In this manner is calculated the execution timing values of each test execution of a test case or a set of test cases from the readings that the Titan TTCN-3 and SoapUI toolsets provide in their test execution logging mechanisms. This same method is utilized in all of the test results that are reported in the following section using a tabular format.
In the following table the execution performance results are presented for both toolsets for common test cases developed on both. The first test case is a simple ‘GetAllOptions’ call on a RESTful Web Service for getting the list of all methods (‘OPTIONS’) supported by the same. It is clearly noticed that the Titan TTCN-3 test case executes faster than the SoapUI test case in all the test runs executed.

Table 1: Test Case GetAllOptions Execution Times

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>SoapUI Time Taken</th>
<th>Titan TTCN-3 Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAllOptions</td>
<td>&lt;operations&gt;GET, PUT, POST, DELETE&lt;/operations&gt;</td>
<td>170</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>52</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>34</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Average Time In Milliseconds</td>
<td>68.8</td>
<td>19.4</td>
</tr>
</tbody>
</table>

Several other test cases were composed in various combinations giving results similar to what is shared in the above performance comparison views and tables. These results lead to confirmation of one of the paper’s objectives in terms of saving time in test execution so as to enable shorter testing cycles and quicker ready to market timelines especially for intermediate or large scale Web Service product environments [7]. In commercial settings with a large scale system there are potentially hundreds and thousands of test cases to be executed in order to determine the current status of a system under test.

It is apparent what a test developer can gain by writing their test systems using Titan TTCN-3 toolset so that they get the development strength of a standardized testing language along with fast execution of their test systems against the SUT. When drawing average execution speed figures the averages from all test runs was compiled for each of Titan TTCN-3 and SoapUI test suites for the test cases developed and executed, giving the following averaged results.

Figure 5: Averaged Execution Times (ms) across all runs
V. CONCLUSIONS AND FUTURE SCOPE

Over the past few years many Web Services testing techniques have been developed and used, each having its own set of advantages and disadvantages. The practicing test developer is on the lookout for ways to make his job easier through automation of the testing process, so as to make the test developer focus on writing the test logic rather than worrying about the toolset limitations and constraints in message formulations and parsing [9]. In this work a test automation technique using Titan TTCN-3 is proposed which makes the Web Services testing job more effective and efficient for a test developer in the market today. The comparison methods used have brought out the strengths of TTCN-3 as a wider scoped testing language which has the ability to adapt to a diverse set of testing needs. The scope of customization that a test developer gains with TTCN-3 is quiet astounding for their testing environment. All this power is available in the open source edition of Titan TTCN-3. The comparison with SoapUI [12] really served as a benchmark for TTCN-3 to prove itself useful for testing of RESTful web services and gladly TTCN-3 lived up to the expectations drawn from the research papers of earlier authors.

In this work study and observations were focused on many aspects of Web Services testing, as well as on the features of TTCN-3, which would be topics in themselves for further exploration into test techniques. For example, if one considers the details of the template matching mechanism in TTCN-3 and how it could be well utilized for parsing and analysing web responses, which is another focus of research. The TTCN-3 toolset [13] features could be applied in a richer manner to pursue security testing of Web Services, in fact a whole network of web services. In the near future the intention is to examine the efficiency of using Titan TTCN-3 in a scalable distributed mode with parallel executing components for testing a distributed system of Web Services. The addition of test data generation and reporting toolsets is another area of study. Test Automation with TTCN-3 is a growing area of study.

REFERENCES