AUTOMATION TESTING OF WEB APPLICATION BASED ON THE NAVIGATION USING JSON

S.K. Muthusundar¹ Research scholar, Bharath University, Chennai.skmuthusundar@gmail.com
Dr. Paul Rodrigues², Dean-CSE Vellamal Engineering College, Chennai, Arul Ganapathy and jawaharRufus³,Dept of IT, Kings Engineering College, chennai

ABSTRACT

Testing such application manually is a tedious one. Therefore, we go for automation testing. Automation testing is performed by navigating the web application using a browser. For modeling the navigation xml notation is used. It is difficult when xml is used for exchanging highly structured data between web applications. Hence, this paper discuss about the navigation modeling using JSON (Java Script Object Notation), which is the open standard designed for data interchange.

1. INTRODUCTION

Software testing is done to ensure the quality of software and to find defects. It is in general a difficult and time-consuming task. Web testing may be even more difficult since it have enormous number of web pages and links [1].This piece of research presents an approach to test web application by automating its navigation. The tasks that have to be carried out in navigation testing are creation of test cases for each path, generating test data, test case execution and finally test report is generated [2]. A web application is tested using different combinations of input and state to reveal bugs. These bugs are caused due to the ambiguous functional requirements. To achieve the automated functional testing, functional requirements should be given in a form that can be understood by software tools [3]. In order to automate the testing process we need to model the web navigation. To find path in web navigation several algorithms (The Chinese Postman Problem (CPP),The Shortest Path Problem(SPP),The node Reduction algorithm) are used. Among these CPP suits the problem of checking websites[4].CPP covers all the links of a web site. Therefore,
CPP is the selected algorithm to find out set of path. For modeling web navigation we use JSON [5] instead of XML. Section 2 discusses about XML and its drawbacks and section 3 discuss about JSON (XML alternative).

2. XML

Extensible Markup Language (XML) provides an easy way to store and share and information [6]. It is a textual data format with strong support via Unicode for the language of the world[7]. Although the design of xml focuses on documents, it is widely used for representation of arbitrary data structures for web applications. To support service oriented architecture, the XML data provider can query web services by parsing the XML structure of the SOAP response directly[8]. This requires knowledge of the web services which includes the namespace, method, SOAP action, parameter and schema of the response body.

To provide the formal declaration of this XML format, XML schema language (XSD) will be used to perform the formalization of the navigation constraints[9]. This XSD schema defines a website as a collection of states (pages) and transitions (links). The initial page is called home, and it is unique. Links are used to connect finite number of web pages (states).

A simple example that illustrates the navigation path of a Website

XSD-SCHEMA of WEBSITE.xml

<?xml version="1.0" encoding="ISO-8859-1" ?>

<website>
  <home id="signin">
    <data locator="username">
      <value>user</value>
    </data>
    <data locator="password">
      <value>pass</value>
    </data>
  </home>
  <transition from="signin">
    <action target="fromdata" event="double click"/>
    <to state="init"/>
  </transition>
</website>
An XML parser converts this XML document into an XML DOM object. The following code fragment parses an XML document into an XML DOM object.

```javascript
If(window.XMLHttpRequest)
{
    Xmlhttp=new XMLHttpRequest();
}
Else
{
    Xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
Xmlhttp.open("GET","WEBSITE.xml",false);
Xmlhttp.send();
xmlDoc=xmlhttp.responseXML;
```

XML DOM represents an XML document as a tree structure. XML deals with data model and traversing but it is not effective in the case of data interchange [10]. For large applications data interchange in XML is quite complex. Therefore, an alternative (JSON) for XML is discussed below.

### 3. JSON

When designing an application that will communicate with a computer, a data format must be selected. There is variety of standardized options and the ideal choice depends on the applications requirements and existing functionality. SOAP – based web services format the data in an XML payload wrapped within a SOAP envelope. XML does not provide any notation of data types[10]. One must rely on Schema for
adding type information. But JSON provides scalar data types and the ability to express structured data through arrays and objects.

JSON is an open, text-based data exchange format. Like XML, it is human-readable, platform independent [11]. Data formatted according to the JSON standard is lightweight and can be parsed by JavaScript implementations with incredible ease, making it an ideal data exchange format for Ajax web applications [12]. It can be used in virtually any scenario where applications need to store structured information as text as follows.

While XML works well for many applications scenarios, it has some drawbacks that make it less than ideal for others. It is not ideal with Ajax-style web applications. JavaScript Object Notation provides a standardized data exchange format that is better-suited for Ajax-style web applications [13].

The following example JSON code implements a website model.

```json
{
  "Home":
  {
    "webpage1":
    {
      "form":
      {
        "Name":"jabas",
        "user_id":"3002",
        "address":"chennai"
      }
    }
  }
}
```
In the above example, arrays are used for data interchange purpose. Since array can deal with enormous amount of data.

4. TEST AUTOMATION FRAMEWORK

Data driven testing approach is used to test a web application based on its navigation [14]. JSON document is given into the JSON parser. JSON parser process and validate the JSON document. In order to store the test data and expected outcome a tabular data file will be used. This file will store test data and expected outcomes [15]. Therefore, this method has one strong prerequisite (i.e) there should be a model of the navigation behavior of the web under test. As shown before, navigation is modeled by using JSON. First test cases for the paths are generated and then the input data’s are added [16].

![Figure 1: Test Automation Framework](image-url)
Data Driven Scripts

Data driven scripts are those application-specific scripts captured in the automation tool’s proprietary language and then modified to accommodate variable data. Here, the processed navigation model is given into the scripts generator. Scripts will be in the format understandable by the automation tool [17]. Variables will be used for key application input fields and program selections allowing the script to drive the application with external data supplied by the calling routine or the shell that invoked the test script.

White Box parser

It translates the navigation model to graphs. White Box testing is also known as structural testing [18]. Here it takes into account the navigation path structure and design the required test cases.

Black Box parser

It retrieves test data (input) and expected outcome (output) from the input navigation model. Regarding input data, this black box parser should extract the value and the data type[19].

Outcome Analyzer

The response data from the SUT are carried into the outcome analyzer. From the response data the navigation state information and the actual data returned by the application are analyzed and extracted.

Reports

Based on analyzed data, reports are generated. Reports indicate the status and description of all test cases. Status may be passing or fail [20]. And the description depicts the reason for the corresponding status.

CONCLUSION

This paper has presented the method for modeling the web navigation using JSON (Java Script Object Notation) instead of XML (Extensible Markup Language). JSON supports data interchange in complex web applications. In functional testing this model is given as an input to the software program to generate automated test case.

Future work will extend the presented approach through the automation of testing and analysis of non-functional requirements such as performance, security, compatibility, usability and accessibility.
REFERENCES


