# A Web Testing Platform Based on Hybrid Automated Testing Framework

Zhe Sun, Yi Zhang, Yunqiang Yan*
Institute of Computer Applications, China Academy of Engineering Physics
Mianyang, China
soonther@sina.com, tigerkids@caep.cn, yyq.caepstc@189.cn*

*Abstract*—**With the widespread use of large-scale web application systems, web automated testing has met many challenges. Faced with the requirements of short cycle, large-scale automated test cases and differentiated data, an efficient test execution infrastructure supporting high concurrency is needed. In order to solve the problem of test data differentiation and test case explosion in the testing process of web application system and reduce the cost of automation maintenance, a web testing platform based on hybrid automation testing framework is proposed, which combines the technical advantages of keyword-driven and data-driven testing framework. We adopt hierarchical design pattern to separate test data from test script, object library and function library. Test data are randomly combined according to numeric, character and non-character text types, which can generate a large number of differentiated test data and can be loaded into test scripts files in parametric form for execution. The separation of object library and function library can support the free and flexible combination of test scripts on demand, reduce the scale of test cases and lowered the maintenance cost of automated testing work.**

***Keywords—web application test; automated test framework; web automation test platform; data-driven; keyword-driven***

## I. INTRODUCTION

In order to save cost, ensure high efficiency and high quality version iteration, automated testing has been widely used in stress testing, functional testing, performance testing, regression testing of web systems, and gradually formed the trend of "automated testing as the main part" in software quality engineering. With the continuous change of business requirements and rapid iteration of versions, especially for large e-commerce websites whose product online cycle is measured by "days" or even "hours", the automated testing of web systems has met many challenges. Faced with the requirements of short-term and large-scale automated testing, a large number of differentiated data requirements in the testing process need to be platformed and serviced in the preparation of test data, and ultimately achieve a unified platform. The rapid growth in the number of automated test cases has also changed the original focus from "how to implement automated test steps with automated scripts" to "how to build automated scripts with low maintenance costs and flexible assembly". It requires automated test scripts to be hierarchically designed, page object modelling and business logic modelling and applied to automated testing framework. Faced with the execution of a large number of test cases, both graphical user interface (GUI) and application programming interface (API) require an efficient test execution infrastructure that supports high concurrency, but the existing web test platform Single function cannot solve a large number of use cases and data differentiation problems in short-cycle, large-scale automated testing.

In terms of functional testing of Web application systems, many researchers have done a lot of work using different technologies. However, Convergence uses the Selenium tool to implement automated testing based on Web services, but it cannot be adapted to the mobile client or HTML5 [1]. Shi N.N. proposed a Web Automation Test Framework using a script template library, but there is still room for improvement in test type and script template maintenance [2]. On the basis of the data-driven engine and test plan driver framework, Wang S.J. proposed a new function-driven framework FDF based on the test case organization form, but it needs a well-designed high-fault test script, which is temporarily unable to provide a large number of differentiated test data. [3]. Zhang L. et al. proposed an automated test method under the IBM open source test framework STAF/STAX. Different test technologies can be loaded with different test drivers, which makes the testers pay attention to the package of the test service and the preparation of the driver, which improves the driver. Reusability and support for a remote distributed test to improve the efficiency of automated testing [4]. However, Zhang L. et al. have not further studied how to package the playback tool as a service to test the GUI test in different environments distributedly and how to parse the EJB test script command with the script parsing module.

In order to effectively solve the problems of inefficiency and complexity of manual testing in regression testing stage, some optimized automated testing frameworks are proposed based on commonly used automated testing frameworks. Fan F.X. et al. designed a Web automated testing framework based on data-driven methodology, but the framework needs to design good test cases and function libraries to avoid the problem of high relevance of test cases, that is, failure of one use case results in failure of all relevant test cases [5]. Wang J. and others have implemented an automated testing framework with keyword driver as the core and Ruby as the development language. There is also a problem of high relevance of test cases [6]. After summarizing and analyzing the existing automation testing framework, Jie H. et al. proposed a keyword-driven automation testing framework based on the system architecture, and gave the key technologies involved. Finally, taking the actual Linux desktop application as an example, this framework was compared with the existing framework [7]. In addition, based on the research of several basic software testing frameworks and the practice of software automation testing in practical projects, Zhu J. et al. proposed a data-driven automation testing framework [8].

## II. AUTOMATED TEST FRAMEWORK

The automated test framework is a program framework for automated testing that provides reusable automated test modules that provide the most basic automated testing capabilities (such as opening a browser) or architectural modules that provide automated test execution and management capabilities. It is a collection of tools consisting of automated test infrastructure modules, management modules, and statistical modules. At present, the commonly used automated testing frameworks include modular testing framework, test library framework, keyword-driven testing framework, data-driven testing framework and hybrid automated testing framework [5].

The modular test script framework typically creates small, independent reusable modules, fragments, and scripts for the application under test. Using the modular packaging idea of programming, these small scripts of tree structure combine to form a script for a specific test case. The modular framework is the easiest to master and use, while also improving the maintainability and scalability of automated test suites.

The test library framework is similar to the modular test script framework and has the same advantages. The difference is that the test library framework breaks down the application under test into procedures and functions rather than scripts. This framework requires the creation of reusable modules, fragments, and function library files for the application under test.

Keyword-driven testing is an application-independent automation framework that requires the development of data tables and keywords to drive test script code for the application and data to be tested. In a keyword-driven test, the functionality of the application under test is written to a table along with the execution steps of each test. It can generate a large number of test cases with very little code. The same code is reused while using data tables to generate individual test cases.

The input and output data of the data-driven test is read from the data file, and the code script generated by the test tool or manually generated is loaded into the variable. Variables are not only used to store input values but also to store output validation values. Throughout the program, the test script reads the value file and records the test status and information. This is similar to keyword-driven testing. In keyword-driven testing, its test cases are contained in data files rather than in scripts. For data, a script is simply a drive or a transport agent. In data-driven testing, only data is included in the data file. This framework is intended to reduce the number of test scripts needed to execute all test cases. Data drivers require very little code to generate a large number of test cases, which is very similar to keyword drivers.

The hybrid automated test framework is a comprehensive type framework evolved from the four frameworks described above after a long period of use. The most common implementation framework is a combination of all the techniques described above, taking advantage of its strengths to make up for its shortcomings. In the course of project practice, most of the single-point technology with different execution frameworks has not formed a well-defined hybrid automated testing framework. Therefore, we will present a web application testing platform based on a hybrid automated testing framework.

## III. A WEB TEST PLATFORM BASED ON HYBRID AUTOMATED TEST FRAMEWORK

To solve the test data differentiation and test case explosion in the web application system test process and reduce the automation maintenance cost, we will combine the characteristics of the keyword-driven and data-driven test framework to provide a web test platform based on the hybrid automation test framework. The overall architecture of the automated test platform includes an automated test management layer, an automated test engine layer, and third-party test tools, as shown in Fig. 1.

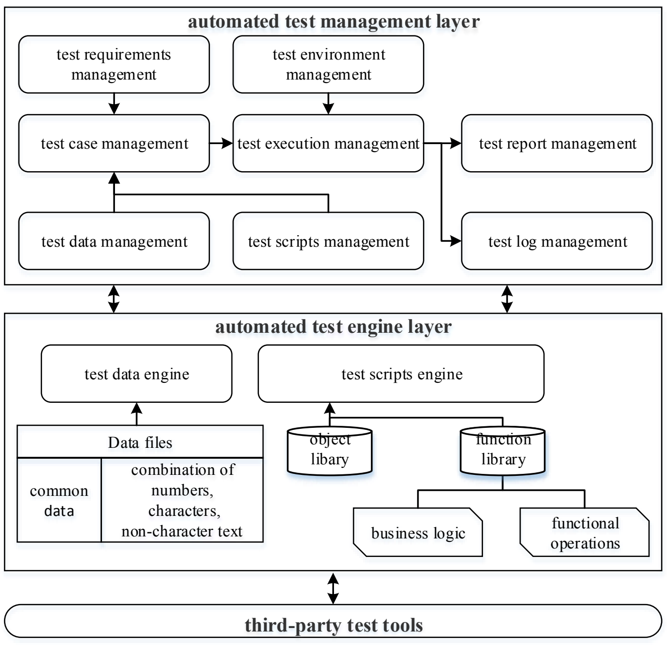

Fig. 1. The architecture of a web Automation test platform.

Automated test management layer is used for the whole process management of web automation test, supporting test implementers to carry out test requirements analysis, test design and test result analysis and summary. The test requirement management module is used for reporting the content of the system under test, the test environment management module is used for the required software and hardware testing resource management, the test case management module is used for the test case design according to the test requirements, and the test script management module is used for test script development and execution management. The test data management module is used to uniformly maintain and manage the test data required for test execution. The test report management module is used for statistical analysis based on the report template integration test results, and the test log management module is used to record the execution process and generate detailed log information for problem location and traceability.

The automated test engine layer is used for the packaging and provision of test data, hierarchical design of test scripts, page object modelling, and business logic modelling, including data-driven modules and script-driven modules. The test data engine module is used to test data packaging and is provided to the test case in a parameterized form. The test data engine module interacts with the test script engine and the system under test in the form of a data file, including three data types: a combination of numbers, a combination of characters, and a non-character text combination. It can support different file formats, including txt files, Excel files, xml files, csv files, ODBC source files, DAO objects [2-4]. The test script engine modules are used to model page objects and business logic and provide common functions to test script development and execution tools. The script driver module encapsulates the elements of the page that need to be tested into object libraries by identifying page elements and calling various business operations in the function library to verify that the results are as expected [2-4]. The elements tested include text boxes, links, drop-down lists, and buttons. The various business logics in the system under test are packaged into a function library, which include user login, addition, deletion, and modification of objects, and operations such as browser operations, database processing, file reading, and mail sending and receiving.

Third-party testing tools are used to develop automated test scripts, execute test suites, feedback and track results.

## IV. Web Automated Test Implementation Process

A web application system is mostly composed of multiple modules, which interact with each other according to business logic requirements. Web application pages are the main media for users to interact with the system. Users request page resources through web browsers. After the web server parses the access request, it sends the data access request to the database driver. After the database driver completes the corresponding data operation, the execution result is returned to the web server, and then displayed to the user through the page.

According to the technical characteristics of web application system, the process of automated test implementation of web test platform mainly includes test requirement analysis, test data preparation, test case generation, test case suite construction, test automation execution and test result analysis, as shown in Fig. 2.

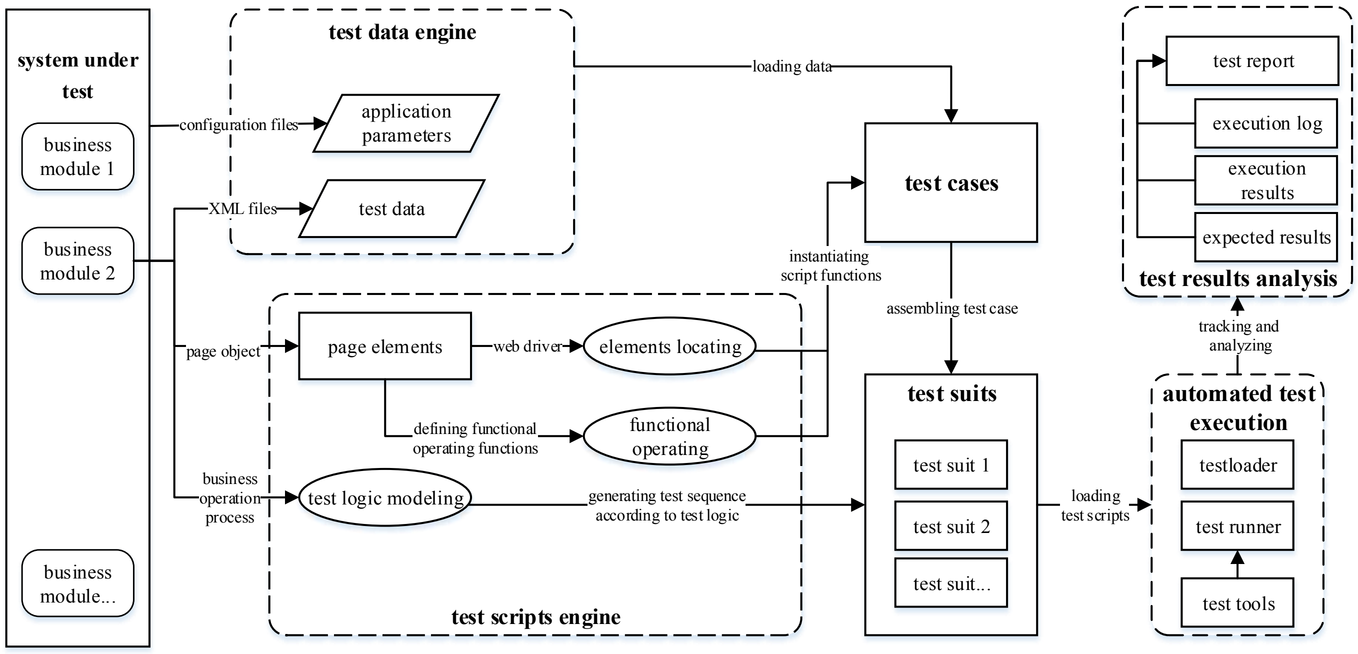

Fig. 2. The iplementation process of a web Automation test platform.

### A. *Test Requirement Analysis*

According to the technical background of the system under test, the implementers analyses the test requirements of the system under test, and find out similar business modules and configurations related to the whole system.

### B. *Test Data Preparation*

For illustration, the test data driver used in this platform will be processed in the form of formatted XML file, which is easy to save and supports cross-platform. The XML data file defines the test data format used by the test cases and shares a test data file with the test case set of business logic. Each data file mainly contains relevant data of the system under test and user test data. In the formatted XML data file, the relevant data of the system under test is described as application parameters, and the custom data is described as user test data. The test data of common parameters and corresponding test cases are a set of < Name, Value > key-value pairs. The name of the XML data file is the same as the name of the test case set, and the name of the test case data corresponds to the name of the test case, as shown in Fig. 3.

```
<Data>
  <Application Parameters>
    <Parameter Name="TempUserAccount" Value="TempUser"/>
    <Parameter Name="TempUserPassword" Value="12345"/>
  </Application Parameters>
  <Test Data>
    <Test Case Name="TestCaseLogin1">
        <Parameter Name ="UserLogin.ByAccount"
                    Value="zhangsan"/>
        <Parameter Name ="UserLogin.ByPassword"
                   Value="12233"/>
    </Test Case>
    <Test Case Name="TestCaseLogin2">
        <Parameter Name ="UserLogin.ByAccount" Value="lisi"/>
    </Test Case>
    <Test Case Name="TestCaseLogin3">
         <Parameter Name ="UserLogin.ByPassword"
              Value="wert12"/>
   </Test Case>
  </Test Data>
</Data>
```

Fig. 3. An example of test data in xml file format.

## C. *Test Case Generation*

According to the content of test requirement analysis, implementers use Page Object design pattern to call Web Driver interface to locate elements and define functional operation functions of page objects in corresponding modules. The interoperability between elements and objects and the message passed can be used as the basis of the function and its parameters when it is implemented. The type of parameters of function operation restricts the value of user test data.

Element relations and dependencies among modules of the system under test are recorded in the form of formatted files (e.g., XML files) and stored according to certain partial order relations to generate test logic. The test logic file stores the module name, message name or operation name and message parameters of each business workflow in strict specification. Combining test data file with test logic file, we can get the object, operation and parameters of each step of the test process, so as to generate test case of automated test platform. Combining the test logic flow with the independent files of test data, the test case script functions and initialization are generated according to the mapping method.

## D. *Construction of Test Case Suite*

Test case is regarded as the running entity of the test unit in the platform. Test case implementers use command and composite design pattern to generate test sequences through test logic files, and combine multiple test cases into test suite, so that multiple test cases can be completed automatically in one test. Test suite can contain not only multiple test cases, but also multiple test suites, which can form a larger test suite.

## E. *Automated Test Execution*

According to the business logic constraints of the system under test, test case executors select the required set of test cases and prepare the corresponding test data to complete the execution of test cases. This platform uses test-runner as the basic execution environment of test cases to drive the whole test process. In the process of test automation execution, all test methods are named according to the convention (beginning with test). The platform uses test-loader to search all test methods contained in the module and execute them automatically. At the same time, in order to facilitate the efficient execution of test case sets, the third-party test tools will be called to assist in the construction of function libraries, IDE environment, data loading, file parsing and other tasks in the automated test environment.

## F. *Analysis of Test Results*

After each round of test case execution, the corresponding test report will be automatically generated. The platform records and tracks the execution process of test scripts, and forms test logs in the form of text or screenshots. According to the test requirement, the test result analyst compares with the expected test result, and carries on the statistical analysis to the automated test result, completes the customization of the test report. The result analysis includes the analysis of the execution process of test script and the execution result of test case, so as to provide data for the product quality evaluation of the software to be tested.

## V. CONCLUSIONS

In this paper, we clarify the concept of large-scale network survivability from the perspective of complex system, analyze the survivability association of large-scale network based upon the association characteristics of complex system, and establish large-scale network survivability association structure model and analyze its natures. Then, we define the survivability association function of large-scale network based upon set pair analysis theory, which is used to depict the degree of survivability association among subsystems in large-scale network. Finally, we validate the effectiveness of the proposed model through case study. The future research work includes the dynamics of survivability association in large-scale network and the application of survivability association model to large-scale network analysis.

## REFERENCES

[1] D. Ningyun, "Design and Implementation of Automated Testing Framework Based on Selenium," Yunnan Normal University, Kunming, China, 2016.

[2] S. Ningning, "An Automated Testing Framework for Applications, " Jilin University, Changchun, China, 2011.

[3] W.Shijun, "Research & Implementation of Software Test Automation Framework," East China Normal University, Shanghai, China, 2006.

[4] Z. Lei, W. Xiaojun, "Automated Testing Based on STAF Framework," Computer Technology and Development, vol. 20, pp. 116-120, Mar. 2010.

[5] F. Fuxing, H. Daqing, Z. Wei, "Research and realize of automation test framework based on Web," Electronic Design Engineering, vol. 20, pp. 36-38, Oct. 2012.

[6] W. Jun, M. Fanpeng, "Research and implementation of automated testing based on keyword driven," Computer Engineering and Design, vol. 33, pp. 3652–3655, April 1955.

[7] J. Hui, L. Yuqing, L. Pei, "Keyword driven automated testing framework," Application Research of Computers, vol. 26, pp. 927-929, Mar. 2009.

[8] Z. Ju, W. Zhijian, Y. Xue, "A Software Automation Test Frameworks Based on Data- Driven Automation Methodology," Computer Technology and Development, vol. 16, pp. 68-70, May, 2006.