# A Keyword Driven Framework for Testing Web Applications

[1]Rashmi
Centre for Development of Advanced Computing,
Noida, India.

[2]Neha Bajpai
Centre for Development of Advanced Computing,
Noida, India.

***Abstract*—The goal of this paper is to explore the use of Keyword driven testing for automated testing of web application. In Keyword driven testing, the functionality of the system-under-test is documented in a table as well as in step by- step instructions for each test. It involves the creation of modular, reusable test components. These components are then assembled into test scripts. These components can be parameterized to make them reusable across various test script. These test scripts can also be divided into various reusable actions. This saves a lot of recording procedure. The Existing tools for this testing uses Html, Xml, Spreadsheet, etc. to maintain the test steps. The test results are analyzed to create test reports.**

***Keyword-driven testing; test automation; test script;, test results; Html; test reports; test result; recording.***

## I. INTRODUCTION

Testing is an integral part of the software development. The goal of software testing is to find faults from developed software and to make sure they get fixed. It is important to find the faults as early as possible because fixing them is more expensive in the later phases of the development. The purpose of testing is also to provide information about the current state of the developed software from the quality perspective.

On a high level, software testing can be divided into dynamic and static testing. The division to these two categories can be done based on whether the software is executed or not. Static testing means testing without executing the code. This can be done with different kinds of reviews. Reviewed items can be documents or code. Other static testing methods are static code analysis methods for example syntax correctness and code complexity analysis.

Dynamic testing is the opposite of static testing. The system under test is tested by executing it or parts of it. Dynamic testing can be divided to functional testing and non-functional testing

The purpose of functional testing is to verify that software corresponds to the requirements defined for the system. The focus on functional testing is to enter inputs to the system under test and verify the proper output and state. The non-functional testing means testing quality aspects of software. Benefits of non-functional testing are performance, security, usability, portability, reliability, and memory management testing.

Automation testing means execution of test cases in an automated way without manual intervention. It was originated from simply record and playback which makes engineers repeat the work. This can be achieved either by using a third party tool like RFT, QTP, etc or by developing an in-house tool suited to the testing need. Test automation includes various activities like test generation, reporting the test execution results, and test management. All these test automation activities can take place on all the different test levels. These test levels are unit testing, integration testing, system testing, and acceptance testing. [2]

Automating the testing is not an easy task. There are several issues that have to be taken into account. These issues are like unrealistic expectations, poor testing practice, and an expectation that automated tests will find a lot of new defects, a false sense of security, maintenance, technical problems, and organizational issues. [2]

The test automation frameworks have evolved over the time. They have evolved into three generations. [3] Figure 1 shows evolution of Test Automation. In the beginning, there was record and playback script creation. In this, there were only stand-alone test scripts. After this, comes the Functional Decomposition. It consists of reusable functional; test modules.

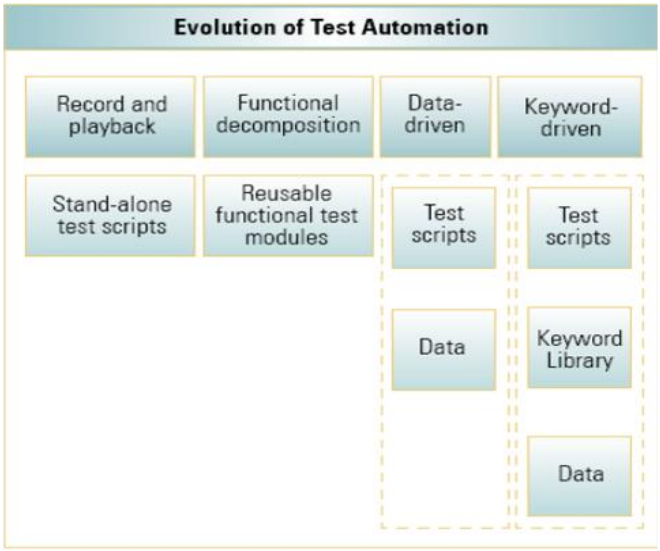


Figure 1. Evolution of Test Automation [7]

After that came data-driven testing. In this, test data is taken out of the scripts. This makes the test data variation easy and similar test cases can be created quickly.

Today, keyword-driven testing is getting more popular. It is a technique that separates much of the programming work from the actual test steps so that the test steps can be developed earlier and can be maintained with only minor updates. It consists of test scripts, keyword library and data. Table 1 shows Benefits and shortcomings of Automated Testing Approaches.

## II. KEYWORD DRIVEN TESTING

It involves the creation of modular, reusable test components that are built by test architects and then assembled into test scripts by test designers. This removes the biggest limitation of the data-driven testing approach.

Keywords can be divided into base and user keywords. Base keywords are keywords implemented in the libraries. User keywords are keywords that are defined in the test data by combining base keywords or other user keywords [3]. The ability to create new user keywords in the test data de-creases the amount of needed base keywords and therefore amount of programming. The Test scripts can be added, deleted and modified. The test script modification helps in the parameterization of the test and in dividing a test into multiple actions.

TABLE I. BENEFITS AND SHORTCOMINGS OF AUTOMATED TESTING APPROACHES [7]

| Approach | How it works | Benefits | Shortcomings |
|---|---|---|---|
| Record and Playback | Users' action are captured, then played back on the application | Ease of Scripting, not much technical expertise required | Difficult to maintain test scripts, not extendable, limited reusability, even small changes to the application require updates of scripts |
| Functional decomposition | Re-usable, repeatable snippet of functions are created | modular approach provides some flexibility, maintainability, reduces redundancy, larger test cases can be built in hierarchical fashion | Data exists within scripts, meaning limited reusability, ease of maintenance, depends largely on technical expertise, framework is high dependent on the framework. |
| Data-driven | Input/output data is maintained in external files | Size of the test pack is greatly reduced, improved maintainability | Depends on technical expertise of test team, maintenance and perpetuation are issues |
| Keyword-driven | Robust, application independent re-usable keyword libraries are built | Ease of maintenance and highly scalable reduced dependence on application availability | Requires great deal of efforts and is time consuming, expertise in test tool scripting language required by framework development |

While testing a web application, there may be needed to check how the web application performs the same operations with multiple sets of data. For example, how a Web application responds to ten separate sets of data is to be checked.

Ten separate tests could be recorded, each with its own set of data. The tool can be test the application with this different data without the need of recording with these data. The test must be saved before running.

Actions divide the test into logical sections. When a new test is created, it contains a call to one action. By dividing the tests into calls to multiple actions, more modular and efficient tests can be designed. This is another feature of reusability of keywords that makes Keyword driven Testing more efficient and modular than the Data Driven Testing.

Various tools for test automation that support this technique are also developed in industry, such as Mercury's Quick Test Professional (QTP) and WinRunner, IBM's Rational Functional Tester (RFT) and Robot on functional testing, and LoadRunner from Mercury, SilkPerformer from Borland, Grinder and JMeter from open source on performance testing.

There are various kinds of keywords which are handled in this technique. These are basically item or base level keywords, utility function and sequence or user keywords. In the keyword-driven testing also the keywords controlling the test execution are taken out of the scripts into the test data. This makes it possible to create new test cases in the test data without creating a script for every different test case allowing also the test engineers without coding skills to add new test cases. This removes the biggest limitation of the data-driven testing approach.

Table 2 shows an example of keyword-driven test data containing a simple test case for testing a login web application. The test cases consist of keywords Runapp, Username, Password and ok, and the arguments which are inputs and expected outputs for the test cases. As it can be seen, it is easy to add logically different test cases with-out implementing new keywords.

To be able to execute the tabular format test cases shown in table 3, there have to be mapping from the keywords to the code interacting with system under test (SUT). The scripts or code implementing the keywords are called handlers.

In Figure 2 can be seen the handlers for the keywords used in test data (table 2). In addition to the handlers, test execution needs a driver script which parses the test data and calls the keyword handlers according to the parsed data. If there is a need for creating high level and low level test cases, different level keywords are needed. Simple keywords like Username are not enough for high level test cases. There are simple and more flexible solutions.

Higher level keywords can be created inside the frame-work by combining the lower level keywords. The limitation of this approach is the need for coding skills whenever there is a need for new higher level keywords.

A more flexible solution proposed is to include a possibility to combine existing keywords in the keyword-driven test automation framework. This makes it possible to create higher level keywords by combining existing keywords inside the test data. These combined keywords as user keywords. [5]

TABLE II. KEYWORD-DRIVEN TEST DATA FILE

| # | TYPE | KEYWORD | OPERATION | PARAMETER |
|---|---|---|---|---|
| 1 | Function | Runapp | | http://in.yahoo.com/?p=us |
| 2 | Item | Username | SetValue | Username=rashmi |
| 3 | Item | Password | SetValue | Password=12345 |
| 4 | Item | ok | Click | |

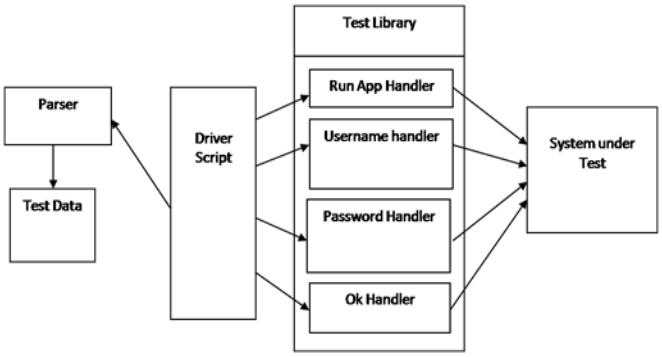


Figure 2. Handlers for keywords in Table 2

There are various advantages of using keyword driven testing techniques. These advantages are as follows:

1) Keywords that reflect the business can be chosen
2) Keyword re-use across multiple test cases
3) Not dependent on Tool / Language
4) The keyword list is robust to minor changes in the software.
5) Division of Labor Test Case

## III. BASE REQUIREMENTS

There are various requirements that are known as "base requirements". These requirements must be fulfilled for success of keyword driven testing. These include:

1) The process of Test development and automation must be fully separated. It is very important to separate test development from test automation. Testers are not and should not be programmers. So, Testers must be adept at defining test cases independent of the underlying technology to implement them. Individuals who are skilled technically, the automation engineers, will implement the action words and then test them.
2) The test cases must have a clear and differentiated scope. It is important that test cases have a clearly differentiated scope and that they not deviate from that scope.
3) The tests must be written at the right level of abstraction such as the higher business level, lower user interface level, or both. It is also important that test tools provide this level of flexibility.

## IV. KEYWORDS

The Test Language is based on a dictionary, which is comprised of words (keywords) and parameters. A Test Case is a sequence of steps that tests the behavior of a given functionality or feature in an application. Unlike traditional test approaches, Test Language uses pre-defined keywords to describe the steps and expected results. Keywords are the basic functional sub-procedures for the test cases of the application under test. A test case is comprised of at least one keyword. [4]

There are three kinds of keywords. These are item or base level keywords, utility function and sequence or user keywords.

Item is an action that performs a specific operation on a given GUI component. For example set value “rashmi" in “User name" control, set value “12345" in “Password control result" field.

When performing an operation on a GUI item, the following parameters should be specified: Name of GUI item, what operation to perform and the values. Table 3 shows Item operations.

Utility Function is a script that executes a certain functional operation that is hard or ineffective to implement as a Sequence. For example: Runapp, closeapp. Table 4 shows Utility Functions.

Sequence is a set of keywords that produces a business process, such as “Login”. Sequence keyword is made by combining various items and utility function. It is recommended to collect frequently used functional processes such as login, addition of new records to the system as a sequence instead of implementing them as items in test cases. Table 4 shows Sequence keywords.

Parameters are additional information required in order to generate the test conditions. In most cases, parameters should be defined for the created keywords. For example: failed authentication by passing username with illegal password, number for mathematical calculation, etc. Table 6 shows Keywords & their associated parameters

Examples for sequence parameters: When the user wants to create a new user, the following syntax is used: create_user ( rashmi, 6/12/2000,rashmi.1306@yahoo.com)

Some of the keywords may contain dozens of parameters. In order to simplify the test creation, all parameters should contain default values. The tester should be able to change each one of the default parameters according to the context of the test. For example, if the tester would like to create new user that is older the 100 years, only the birth date will be changed and all other parameters will remain the same. Obviously, a specific change will not affect the default parameters being used for other tests.

TABLE III. ITEM OPERATION

| # | TYPE | KEYWORD | OPERATION | PARAMETER |
|---|---|---|---|---|
| 1 | Item | Username | SetValue | Username=rashmi |
| 2 | Item | Password | SetValue | Password=12345 |

TABLE IV. UTILITY FUNCTION

| # | TYPE | KEYWORD | PARAMETER |
|---|---|---|---|
| 1 | Function | Runapp | http://in.yahoo.com/?p=us |
| 2 | Function | Closeapp | http://in.yahoo.com/?p=us |

## V. OBJECT REPOSITORY

Object Repository is a centralized place for storing Properties of objects available in Application under Test (AUT). All software applications and websites are getting developed using many different components or small units like textbox control, input tag, web browser control etc. These components or small unit are known as Objects. [10]

Each object will be identified based on the object type. Each object will also have its own properties like name, title, caption, color, size. These properties help in the identification of these objects uniquely. There are also specified set of methods for each object. There are various properties that can be changed during run-time. These properties are known as Runtime Object (RO) properties. There are also some other properties that can't be changed. These properties are known as Test Object (TO) properties. [10]

There are some additional properties such as index, location which are known as ordinal identifiers. Actually these properties won't be available in the object of Application under the test. These are created in order to distinguish two objects which are having exactly same Test Object properties. For instance, some forms in the web pages will be have two submit buttons, one at top and another at bottom. These both can be identified separately on the basis of the location or index. As Test Object properties are also based on properties of object of Application under Test, there is no need for all the Test Object properties to be available in Runtime Object properties collection also. The object repository must support the editing of the properties of these Test Object and new properties can also be added to them. The value for the properties of the Test Objects in Object Repository need not be a constant. They can parameterize the values so that the Test Object property can be dynamically changed for each execution.

These properties are stored in the centralized place in object repository. This helps in the maintenance and updating of Test scripts can be easily done whenever there is a change in UI (User Interface) of the AUT. Assume that Login screen is used in around 20 Test scripts. If the Page name of login screen in changed, there is no need to make any change in all these 20 Test scripts. By just changing the property of Test Object in Object Repository is enough. A clear understanding of Object Repository is essential to carry out the operation of the Keyword driven testing successfully.

A framework for testing should be able to recognize any control or object in any webpage that needs to be testes. For recognizing the object, it should know the properties of those objects beforehand. During the execution of the test scripts, this identification is done. A framework has data tables for supporting the execution of multiple iterations of same step with different data.

There can be various methods to manipulate the test object properties. The Test Object properties of Test Objects can be accessed by implementing methods such as getTOproperty and getTOproperties.

TABLE V. SEQUENCE

| # | TYPE | KEYWORD | PARAMETER |
|---|---|---|---|
| 1 | Sequence | Login | Username=rashmi<br>Password=12345 |

TABLE VI. KEYWORDS & THEIR ASSOCIATED PARAMETERS

| # | Type of keyword | Parameter |
|---|---|---|
| 1 | Item | Value |
| 2 | Function | Value |
| 3 | Sequence | Item |

Even, Test Object property of Test Object can be changed using setTOproperty. It will be valid only till exiting the execution. After completing the execution it will resume the actual Test Object property stored in the Object Repository. During run-time we can get the property of the runtime object using getROproperty.

## VI. KEYWORD DRIVEN MODULE

A framework used for performing keyword driven testing will consists of various interrelated modules. These modules are namely core module, scripting language module, support library module and many more. [1] Figure 3 shows the Keyword driven module.

Core module takes a major role in analysis the keyword information of the script, and controls the implementation of the scripts. It is composed of four parts that are the data parser, the script parser, the script actuator and the middle layer. The data parser is responsible for analysis on the keyword, the script parser responsible for analysis the logic keyword in the script, the script actuator is responsible for the implementation of the script, and the middle layer is responsible for calling the test. [1]

Data Access module is responsible for data storage, including add scripts, modify scripts, read scripts, enquiry scripts, delete scripts, and other functions. The script has three levels, when the high level and low-level script bearing the script, the layer will maintain this relationship. [1]

Interface module is to enhance the framework's ease of use. It realize a GUI interface, the graphical interface allows users to edit, drag and drop the modalities script; provides a user friendly guide to understand and use; provides view and editor which make it easily for users to view, modify the existing test scripts. [1]

Support module consists of two parts: one is the libraries that all of the tests can be shared, including the log library and the test supporting library. The log library is responsible for providing the functions of log records to testers; the test supporting library provides the functions that all of the tests can be shared. The second is the testing library for GUI; this part provides the controls libraries. [1]

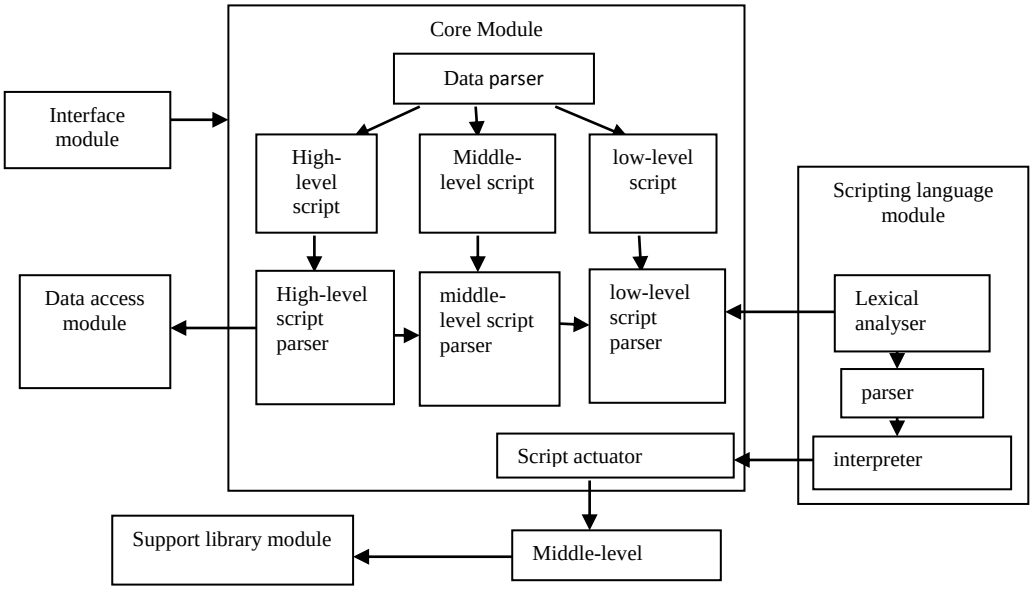


Figure 3. Keyword driven module [1]

Interpret scripting language module is consisted of three parts, such as lexical analyzer, parser, and interpreter. Lexical analyzer will be responsible for the output of the characters in a flow analysis as a word, parser will be responsible for the sequence of words with semantic analysis for the phrase and interpreter will be responsible for the semantic translation. [1]

## VII. PPOPOSED APPROACH

First, a web application is taken which is needed to be tested. A web application can be a login page, online shopping web application, online reservation web application etc. To start the process of recording, enter the URL of the desired web application. The process of authentication in Railway Reservation web application, in which user passes user ID and a password, is taken as an example to describe the proposed approach.

As user navigates the Web application, the Keyword driven testing framework records the steps. In the User Name and Password boxes, the name and password are entered and the Sign-In Button is clicked. The travel planning web page opens (If both correct username and password are entered). These operations form the basis of the test. The tool records all the operations performed in the Web browser until the recording is stopped.

When the recording is stopped, a test script file is generated. This generated test script file containing all user actions, is saved. The user actions will comprise of items clicked, items selected, value typed etc, during the recording procedure. The tool will also generate steps in the table format, representing each operation performed in the form of Keyword, Value and Operation. For example, User Name text field, Password text field and Sign-In button are the keywords. The Value represents values entered by the user in the items. The Operation includes the click, select, drag or drop, etc. The test Script will be useful in the play back of a test and reusability of the test script i.e. parameterization and multiple actions.

When the test is play backed, the tool runs the saved test script file. The recorded web application opens in the web browser and all steps are performed automatically, as it was originally recorded in the test. For example, the recorded test script for authentication process can be played backed. Parameterized tests and multiple action tests are also played back using the play back module.

When the test run is completed, it displays the results of the run (whether a test is passed or failed) in the test result page. The Test Results window opens, which contains the result summary of the test execution. The Test Results window displays the key elements of the test run for test analysis purpose. The key elements are composed of two parts. First element shows the steps (in the tree structure format) that were performed while the test was running. The second element is the test result details. The test result contains iteration and status summary. The iteration summary indicates which iterations passed and which failed. The status summary indicates the number of test or reports that passed, failed, and raised warnings during the test.

Object Repository is a centralized place for storing the properties of objects available in AUT (Application under Test). The keywords can be added in the object repository, either manually or at the time of recording. All software applications and websites are developed using many different components or small units (for example *textbox control in VB, input tag in HTML, web browser control in .net*) which are known as Objects. Each object is identified on the basis of the object type. Each object has properties (for example *name, title, caption, color and size*) and specific set of method, which help in identification of an object. The object repository will support the modification of Test Object's properties, as well as, new properties can also be added. The values of the properties stored in Object Repository need not be a constant. The values can be parameterized by making them variable.

The Test scripts can be added, deleted and modified. The test script modification helps in the parameterization of the test

as well as during the division of a test into multiple actions. While testing a web application, there may be needed to check how the web application performs the same operations with multiple sets of data. For example, how a Web application responds to twenty separate sets of data is to be checked. There are two ways to do this. Twenty separate tests are recorded, each with its own set of data. Here, no reusability of Test Script. Alternatively, A test is recorded, and the values of the objects in this test are made variable for parameterization. This saves the nineteen runs of the recording process. This single test with the help of parameterization can be played twenty times using a different set of data each time. Each test run is called iteration. All iterations are numbered. Later is the better approach and involves the reusability of Test Script. In the above example, the authentication page is signed in with 'rashmi.1306' as user ID and 'dracoXXXXXX' as password. The 'rashmi.1306' is a constant value, which means that 'rashmi.1306' is the user ID each time the test is run. Through the data table parameter, the user ID can be changed into a variable, so that a different user id can be used for each test run. In the parameterized Keyword user ID, two different user ids can be added for example 'ras_gupta' and 'prati_gupta'. The tool can be test the application with this different data without the need of recording with these data.

Actions divide the test into various logical sections. When a new test is created, it is represented as a single action. By dividing the tests into multiple actions, more modular and efficient tests can be designed. This is another feature of reusability of keywords that makes Keyword driven Testing more efficient and modular than the Data Driven Testing. To explain this we take the whole example of the above railway reservation web application that books a flight. This can be divided into several distinct processes or actions which are as follows:

- The Online railway reservation web application is logged in.
- The trains are booked.
- Another action for logged out from the web application.

The above test can also be parameterized for ten different train booking. This parameterized test now can be run ten times using ten different sets of data. With the help of Multiple Action, the test can also be organized so that only the second procedure runs ten times, simulating a single user logging in, booking ten trains, and logging out. This can be done by dividing the test into different actions. This saves the nine runs of Logged-in and Logged-out process.

The parameterized tests and tests divided into multiple actions are also play backed simply and their test results are analyzed. Figure 4 shows the above proposed approach in a flow chart.

## VIII. CONCLUSION

In this paper, the different types of keywords, base requirement, methodology, object repository and various keyword driven modules are investigated. These all are required to carry out a successful and efficient operation of Keyword driven testing. It is important to understand that keywords are not magic, but they can serve well. It is essential to do test design in a right and efficient way. The process of the test automation should be done but it should not dominate the process. It should flow from the overall strategy, methodology, and architecture. Moreover, the existing tools available for this approach make use of the HTML, Xml, spreadsheets to maintain test cases in object repository which are not very scalable.

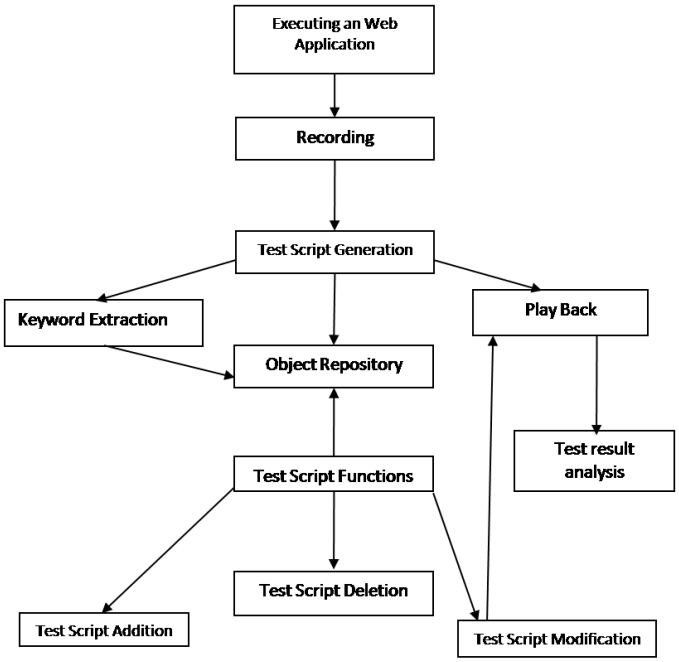


Figure 4. Proposed Approach

## REFERENCES

[1] Jie Hui, Lan Yuqing, Luo Pei, Gao Jing, Guo Shuhang, "LKDT: A Keyword-Driven Based Distributed Test Framework", International Conference on Computer Science and Software Engineering, 2008, pp. 719-722

[2] Pekka Laukkanen, "Data-Driven and Keyword-Driven Test Automation Frameworks", Helsinki University of Technology, Department of Computer Science and Engineering Software Business and Engineering Institute. 2007, pp. 1-102

[3] Juha Rantanen,"Acceptance Test-Driven Development with Keyword-Driven Test Automation Framework in an Agile Software Project" Helsinki University of Technology, Department of Computer Science and Engineering, Software Business and Engineering Institute. 2007, pp. 1-102

[4] Ayal Zylberman and Aviram Shotten, "Test Language: Introduction to Keyword Driven Testing". 2010, pp 1-7

[5] Tommi Takala, Mika Maunumaa, and Mika Katara. "An Adapter Framework for Keyword-Driven Testing", Department of Software Systems, Tampere University of Technology, Finland. Ninth International Conference on Quality Software. 2009, pp. 201-210

[6] http://en.wikipedia.org/wiki/Keyword-driven_testing

[7] Bharath Anand R., Harish Krishnankutty, kaushik Ramakrishnan, Venkatesh V.C.," Business Rules- Based Test Automation- A novel Approach for accelerated testing". 2007, pp. 1-12

[8] Liu Xing, Li Yan, Cai Mian, Guo Ying, "The Testing and Evaluation System for the Secure Operating System Based on the Mechanism of

keyword-driven". Ninth International Conference on Information Assurance and security. 2009, pp. 471-474

[9] http://en.wikipedia.org/wiki/Test_automation

[10] http://www.automatedqa.com/products/testcomplete/ manager-overview/

[11] Bennett, "J.P. Introduction to Compiling Techniques – A First Course Using ANSI C", Lex and Yacc [M]. McGraw Hill Book Co, 1990.

[12] Nancy S. Eickelmann, "An evaluation of software test environment architectures". International Conference on Software Engineering, 1996, pp. 353-364.

[13] Terence Parr," The Definitive ANTLR Reference: Building Domain-Specific Languages", Pragmatic Bookshelf, 2007, pp 14-85.

[14] Sheng Liang. Java(TM) Native Interface, "Programmer's Guide and Specification", Prentice Hall PTR, 1999, pp 1-35.

[15] Mercury QuickTest Professional Tutorial, Version 8.0

[16] Kaner, "Pitfalls and strategies in automated testing", IEEE Computer, 30(4): April 1997, pp 114–116,

[17] Kaner, J. Bach, and B. Pettichord, "Lessons Learned in Software Testing: A Content-Driven Approach", John Wiley & Sons, Inc., 2001.

[18] Kelly, " Choosing a test automation framework, July 2003",URLhttp://www106.ibm.com/developerworks/rational/library/591.html. April 30, 2005.

[19] Kit, "Integrated, effective test design and automation". Software Development, February 1999, pp 27–41.

AUTHORS PROFILE

**Rashmi**, done B.Tech (Computer Science) in 2010 with 79% from M.D.U. (Rohtak), Currently pursuing M.Tech (Computer Science) from Centre for Development of Advanced Computing, Noida, I.P. University and doing a project on "A Keyword driven framework for testing web applications".

**Mrs. NEHA BAJPAI** received M.Tech in Information Technology from the Vinayaka Mission University of Tamilnadu in the year 2005. She has ten years of teaching and one year of IT implementation experience. Presently, she is working as a Senior Faculty in School of IT at Centre for Development of Advanced Computing (CDAC), Noida. Her present interests are in the subjects related to Object Oriented Technologies, Oriented Analysis & Design, UML, Software Testing and Object Oriented Database Management System. She has over 15 research papers in various international and national Journals, Conferences & Seminars. She also served, coordinated and taught various International Training Programs under Indo-Vietnam bi-lateral Cooperation and ITEC/SCAAP Scheme of MEA.