Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Table - Driven Approach for Automated Testing of Web Applications

C.Bhuvana[1], K.Munidhanalakshmi[2], Dr.R.Mahammad Shafi[3]
[1]*Assistant Professor, Department of CSSE,*
Sree Vidyanikethan Engineering College, Affiliated to JNTUA, Tirupati, India
[2] *Assistant Professor, Department of M.C.A,*
Sree Vidyanikethan Engineering College, Affiliated to JNTUA, Tirupati, India
[3] *Head of the Department, Department of M.C.A,*
Sree Vidyanikethan Engineering College, Affiliated to JNTUA, Tirupati, India

Abstract- Table-Driven approach tends to be more application-independent than linear, data-driven, hybrid frameworks. The Objective of this paper is to explore the use of Table driven testing for automated testing of web application. In Table 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. Framework involves the creation of modular, reusable test components, Reusability, Script's easy maintenance, Readability of script components. All the components are assembled into test scripts. The components can also be parameterized to make them reusable across various test scripts. The test scripts can be divided into various reusable actions. This process can reduce recording procedure. The Existing tools for this testing uses HTML, XML, Spreadsheet, etc. to maintain the test steps. Finally, test results are analyzed to create test reports.

***Keywords**: Test automation, Keyword, Application Map, Component Function, Test script, Test results, Test reports, recording.*

1. Introduction:

The way software industry moved from manual testing to test automation. it is now moving towards Business Process Automation. Business Automation works towards making automation frameworks more and more meaningful to the Functional Subject matter Experts by involving them in test Automation as much as possible thereby making Test Automation business driven. To keep pace with the latest trends in the industry and to have an edge over its competitors, it is important for a testing organization to adapt itself to this new approach. One tool which is in line with this approach is Mercury's Business Process Testing (BPT). Being Industry's first web based test design solution; BPT is one step further to the Table Driven test Automation approach. For any Software Testing Organization, moving to a new tool like BPT is a big step. For it to justify this big step, it is important for the organization to have successfully implemented the Table Driven Test Automation Approach. This is because, it is easy for a project to implement BPT if it is already following Table driven approach than for a project which is not. One of the benefits of using this approach is the automation effort saved due to reusability of the keyword components across different applications. This benefit is realized when the extent of reusability is high. This is possible when the type (eg:Web,VB,SAP etc.) is common across applications to be automated, because kerword components built with a particular QTP(Quick Test Professional) Add-in can be reused only with applications that can work with that Add-in.
Today, most Organizations are developing web based application as against client server or mainframe applications. in some instances, organizations are converting their applications from legacy systems to web. Considering all this. it makes business sense for testing Organizations to go for keyword driven Test Automation Framework for Web-based applications[1].

2. Purpose:

This Paper is an attempt to build a Table Driven Test Automaton Framework which can be used across different web based applications. In this approach, the endeavor is to build a lot of application independent reusable keyword components so that they can directly used for another web application without spending any extra effort. With this framework in place, whenever we need to automate a web based application, we would not need to start from scratch, but use the application independent keyword components to the extent possible and create application specific components for the specific needs. Linear Approach version is the simplest form of creating a test. It is just writing one single program without modularity in sequential steps. In

Simple Business Process testing programs are broken down into business components and are used with one or the other of the above types of frameworks. Keyword driven

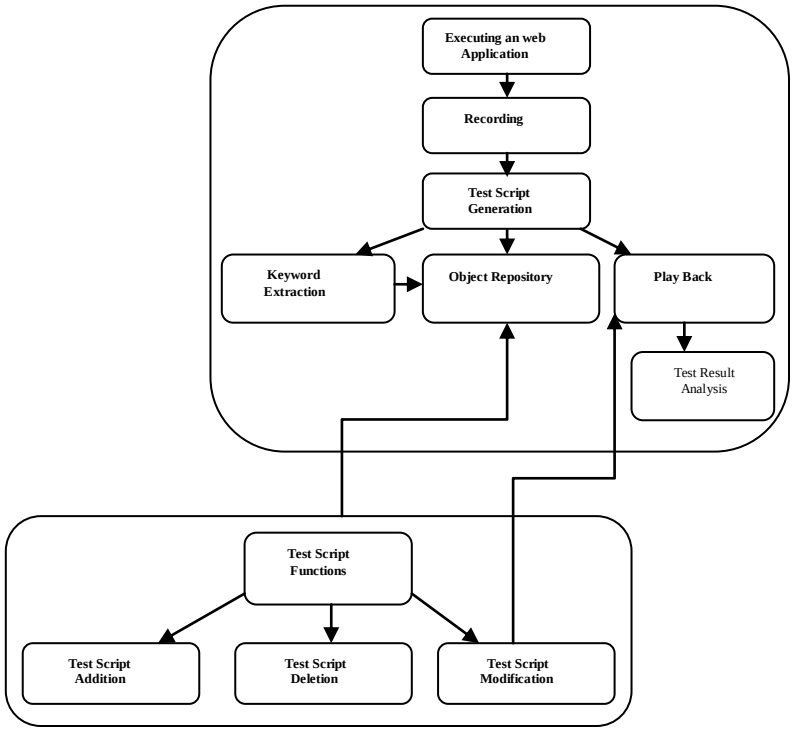


Fig 1: Proposed Technique

Version Create different keywords for different set of operations and in the main script we can just refer to these keywords.

3. Automation Testing:

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[2].

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.

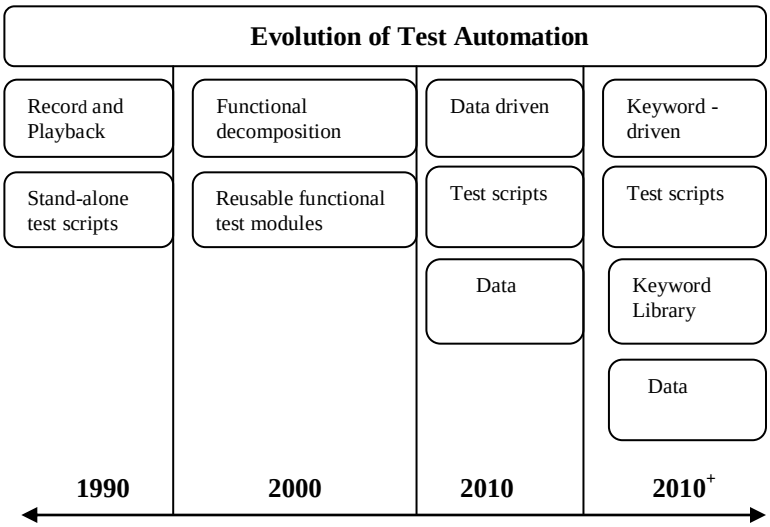


Fig 2: Evolution of Test Automation

Why Automation?

i) Reuse of Tests: When a certain set of test scripts require frequent execution it makes more sense to automate them and reap the benefits of unattended automated execution of test scripts. Another example will be in cases where a single script needs to be executed with multiple data sets. In such cases the effort to automate a single script and running it with multiple data sets is far less than the manual execution effort for all those data sets.*ii) Time Save:* Running unattended automated test scripts saves human time as well as machine time than executing scripts manually.*iii) Better use of people:* While automated scripts are running unattended on machines, testers can do more useful tasks.*iv) Cost Saving:* On test engagements requiring a lot of regression testing, usage of automated testing reduces the people count and time requirement to complete the engagement and helps reduce the costs.v) Machines are more reliable than humans: Great confidence will be gained when a system is released, when we use automated testing[8].

Keyword:

Keyword Driven Testing involves the creation of modular, reusable test components that are built by test architects and then assembled into test scripts by test designers. Keywords can be divided into **base and user keywords[1].** 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. 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.There are various kinds of keywords which are handled in this technique. These are basically **item or base level keywords, utility function and**

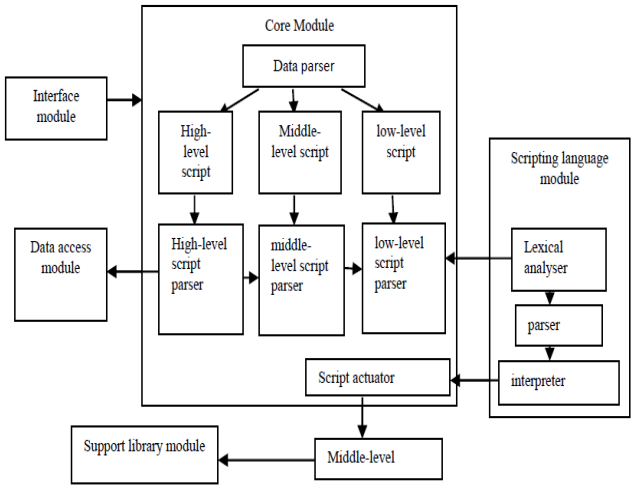


Fig 3: Table-Driven Module

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.

4. Framework Structure:

The framework consists of the following components:

4.1 Function Library
4.2 Application Map
4.3 Database
4.4 Application Scenario Files
4.5 Initialization Vb Script
4.6 Sequence File
4.7 Driver Script
4.8 Test Case List File

4.1 Function Library

In this approach, all the coding logic is in the form of user defined VB script functions. All of these functions are stored in function libraries (vbs file).

The directory structure in which these components are arranged is as shown below.

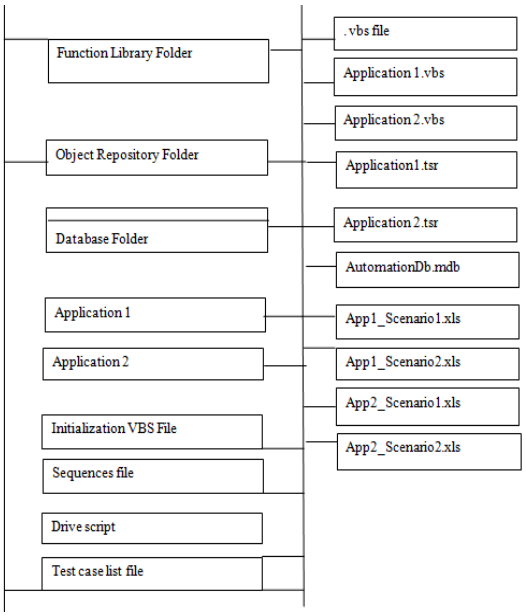


Fig 4: The Directory Structure

In function library there is absolutely no scripting done outside of the fuction library except for the Driver Script for each application, there are two function libraries,one which contains all the application independent functions(or common functions) and another that has application specific functions. While developing scripts for an application,the endeavor would be make use of the application independent functions as much as possible. In case, some functionalities require application application specific functions, create them and put them in the application specific function library. The common fuction library has a function (ExecuteScenarioFile) which reads the keywords, objects and other parameters form the Scenario File and makes calls to appropriate functions in the function library. This is the only function which is called from outside of the function library(this function called from the Driver Script).

Some of the common functions that can be created for web applications are as given below,
1. Close all browser windows and launch a new instance of browser.2. Click on a Web Button with all error handling.
3. Enter data in a Web Edit object with all error handling.4. Import data from MS Access to run time data table.5. Verify text on a Web page with all error handling.
All the function library files (.vbs files) are kept in the Function Library folder in the directory structure.

4.2 Application Map
An Application Map Provides Named References for GUI Components. Application Maps are nothing but "Object Repositry.Each application automated has a different object repository file and there is only one object repository for each application. All the object repository files (.tsr files) are kept in the object repository folder in the directory structure[3].
Component Function:
Component Functions are those functions that actively manipulate or interrogate GUI component. An example of a function would be click on web button with all error handling, enter data in a Web Edit with all error handling. Component functions could be application dependent or independent.

4.3 Database:
All the test data is stored in MS Access Database. the database structure is designed in such a way that there is more or less one table for each screen on the application. almost every table has test case Id as one of the columns and is either the only primary key or one of the primary key columns. The Test case Id represents a record or the data used for iteration.The database might have 100 sets of dat,but the user might want to execute only 10 of them. the Test Case Id helps in this situation by providing the user with the option to select only certain sets of data.The connection to this database is established from QTP by creating a System DSN for the database and using this DSN in the script.Ideally,we can also have one database for multiple applications to save disk space. All the database files (.mdb files) are kept in the database folder in the directory structure[3].
4.4 Application Scenario Files:
An application scenario file is a spreadsheet which contains keywords, objects and other parameters arranged in the desired order to form a test scenario. It is from this file that QTP reads information and performs actions on the application. It is this feature that makes the keyword driven frame so powerful. Each application has an application folder and all the application

scenario files (.xls files) belonging to that application are kept in the corresponding application folder.

4.5 Initialization VB Script:

The initialization VB Script is the starting of script execution. It launches QTP, sets the work area as development or production based on the user input and does the following settings to the driver script.

- Set work area based on user input.
- Set the application scenario file for the current run.
- Set object repository.
- Set function libraries.
- Set data source.
- Set the test case list.

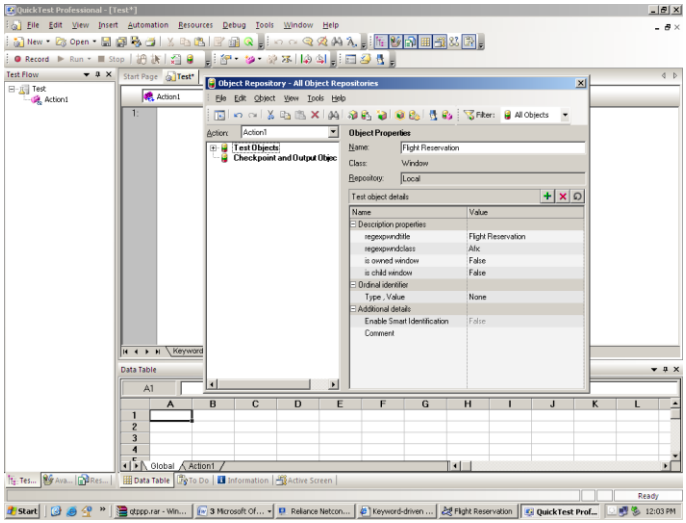
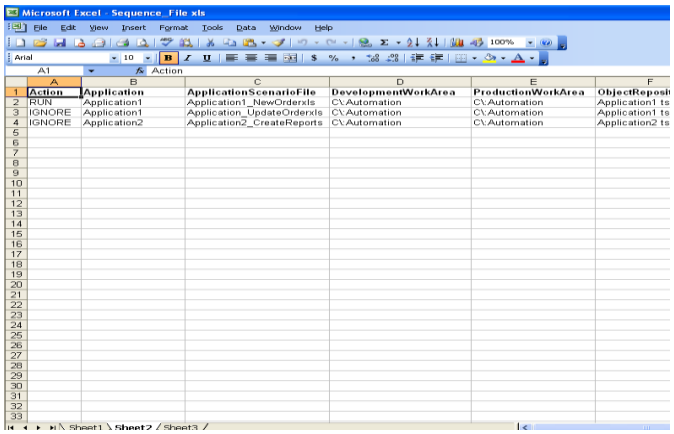


Fig 5: Database Structure

After doing the above settings, it opens the driver script in read only mode for the user to star execution. With this, the control is passed on to drive script. The initialization VB script (.VBS file) is kept directly under the root directory[6]. The significance of the initialization VB script is that it allows the user to run the same script to do the initialization settings for different web based applications.



	A	B	C	D	E	F
1	Action	Application	ApplicationScenarioFile	DevelopmentWorkArea	ProductionWorkArea	ObjectReposi
2	RUN	Application1	Application1_NewOrderxls	C:\Automation	C:\Automation	Application1 ts
3	IGNORE	Application1	Application_UpdateOrderxls	C:\Automation	C:\Automation	Application1 ts
4	IGNORE	Application2	Application2_CreateReports	C:\Automation	C:\Automation	Application2 ts

Fig 6: Application Scenario file

4.6 *Sequence file:* Sequence file is a like configuration file which contains information required to do the initialization settings to the Driver QTP Script for a particular application. The Sequence file is in the form of a spreadsheet which has an entry for each application and settings data such as application scenario file name, object repository name, function library name, test case list for execution, the data source ,script development work area and script execution work area. Each application has a "Run" or "Ignore" flag against it to say which application scenario is to be executed[7]. The Initialization VB Script uses the spreadsheet to decide the application & scenario to be executed, to get information about the Application Scenario file and do the initialization settings for an application.

4.7 Driver Script

Driver script is a QTP test script which drives the script execution after control from the initialization VB script. Strictly speaking; this is the only script which is present as a QTP test script and not in the function library. As mentioned in section 4.1, this test script calls the function "ExecutionScenarioFile" which reads the keywords, objects and other parameters from the scenario file and makes calls to appropriate functions in the function library. The beauty of this framework is that all the script execution is covered by a single test and single QTP action[4].

The drive script is kept directly under the root directory.

4.7 Test Case List File

Test case list file contains the list of test case Ids (explained in section 4.3 on database) to be executed in the current run. As mentioned in section 4.3, this provides the user with the option to select only a subset of the data in the database for execution of the current run.

Table 1: Comparision between Table-Driven Framework and other Frameworks [6]

		Record and Playback	**Functional decomposition**	**Data-Driven**	**Table-driven**
Feature : 1	**Work Status**	**Users' action are captured,then played back on the application**	**Repeatable functions are created**	**Input/ Output data is maintained in external files**	**Robust, application independent reusable keyword libraries are built**
Feature :2	**Benefits**	**Not much technical expertise required**	**Larger test cases can be built in hierarchical fashion**	**Size of the test pack is greatly reduced**	**Ease of maintenance and highly scalable reduced dependence on application availability**
Feature :3	**Shortcomings**	**Difficulty to maintain scripts, even small changes To the application require updates of scripts**	**Data exists within scripts, should contain technical expertise**	**Depends on technical expertise of test team, maintenance issues**	**Expertise in test tool scripting language required by Framework development**

5. Developing a strong & robust automation framework:

The key to develop a strong & robust keyword driven automation framework is a challenge for any testing organization. To make this happen and realize the benefits of using it, the following guidelines are to be considered while developing the framework using the approach mentioned in this paper[5].

- **Centralized automation team –** One of the factors that stand out as a clear benefit in using this approach is the ability to reuse keyword components across applications thereby saving a lot of time, effort and cost. However, this calls for a lot of coordination between people doing automation in different projects. Therefore, it makes sense to have a centralized automation team working for different projects.
- **Coordination between function and automation teams –** Though this approach provides the functional values with more flexibility to design the business flow to create automated tests, the automation teams still need to work closely with the functional Values to make them understand what each keyword does, how the object names map with the logical names in the object repository and the database structure.
- **Creating right functions -**The heart of the framework is the logic behind the functions in the function library. The way you design a function can make or break the whole framework. So, it is very important for the automation team to do proper analysis before creating functions. The endeavor should be to make the functions as generic as possible so that they can be reused across applications**.**
- **Script maintenance –** Even though changes in the application can be taken care of by updating the application scenario spreadsheets, it would be a good practice for the automation team to make sure that all of their functions are still good for use. Though this may not be a huge effort as in traditional automation framework, it might call for some work here and there[8].

6. 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.

References:

[1] Ayal Zylberman and Aviram Shotten, "Test Language: Introduction to Keyword Driven Testing".
[2] Bharath Anand R., Harish Krishnankutty, kaushik Ramakrishnan, Venkatesh V.C.," Business Rules- Based Test Automation- A novel Approach for accelerated testing".
[3] B Liskov, "A Framework for Automated System Testing", WebURL: "dspace.mit.edu/bitstream/handle/1721.1/40188/35334586.pdf", 1996.
[4] Jie Hui, Lan Yuqing, Luo Pei, Gao Jing, Guo Shuhang, "LKDT: A Keyword-Driven Based Distributed Test Framework".
[5] 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.
[6] Laukkanen, Pekka, "Data-Driven and Keyword-Driven Test Automation Frameworks", Master's Thesis, Software Business and Engineering Institute, Department of Computer Science and Engineering, Helsinki University of Technology, 2006.
[7] Pekka Laukkanen, "Data-Driven and Keyword-Driven Test Automation Frameworks", Helsinki University of Technology.

