Automation Test Frameworks

By Sidartha Gracias

Automation Testing

- Traditionally performed with tools that mimic manual test flows using a record and play-back system similar to marco recording in excel

<u>How does this work</u>

- Capture manual test flow, using record capability.
- While recording, captures object on which actions are performed and stores them in an object repository.
- On playback objects on page checked against OR for match.
- Tools then generates user actions on objects (e.g. buttonclick) to replicate test flow.
- Test scripts can be customized to use multiple data sets for the same test flow or to modify test flow.

The Problem?

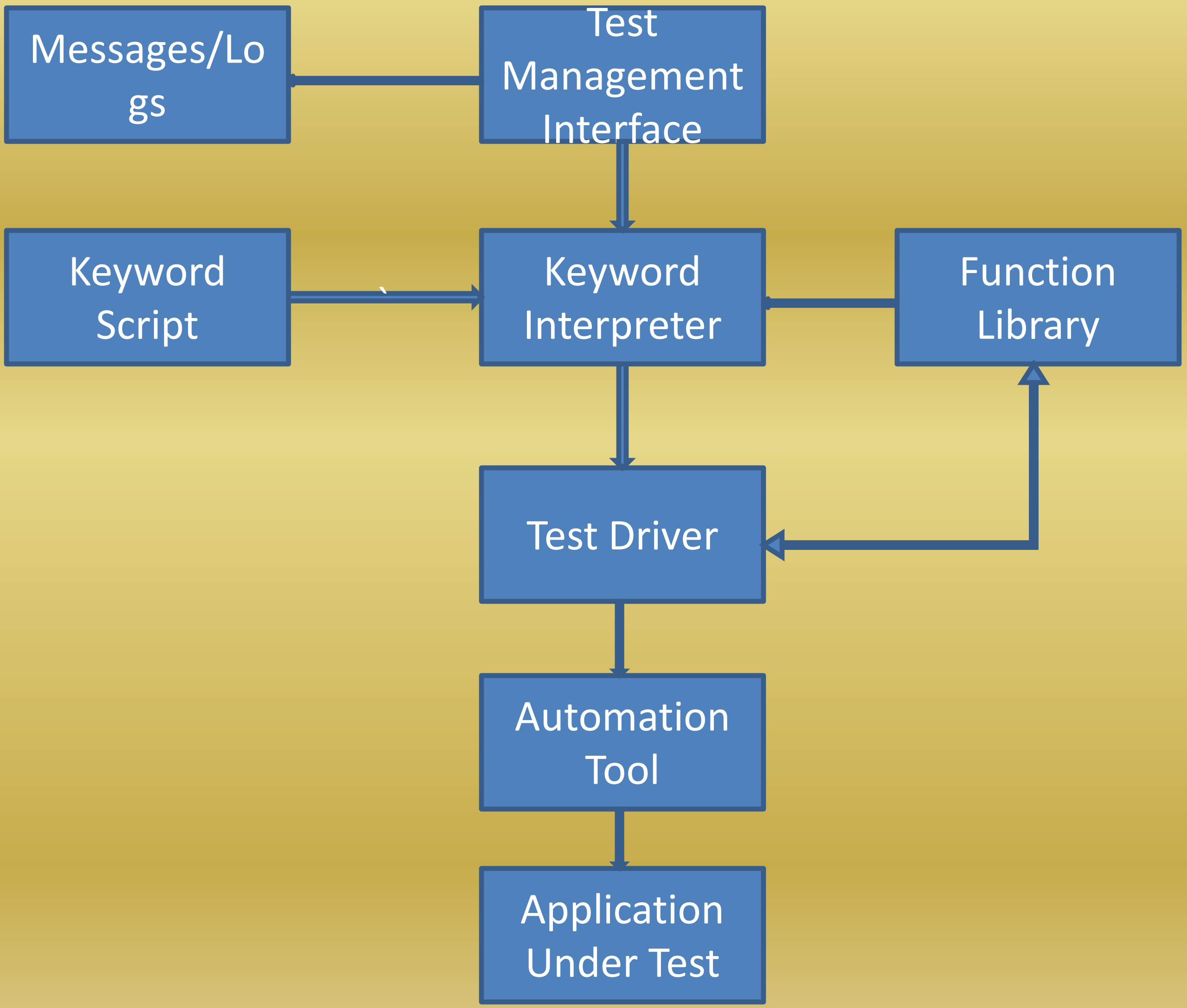
- Test Automation has for the most part been a failure. Why?
- Automation tools have been oversold. They have been given capabilities that they do not necessarily posses. E.g. demonstrate capabilities on one system and believe that would extend to all systems.
- Record and Playback looks great but performs poorly in practice.
- Work great for small projects but scales poorly.
- Do not port well from one project to another.
- Hard to customize

The Solution - Automation Frameworks

<u>The Idea</u>

build a layer over existing automation tools.

- Tools still used (in the general case) to capture object properties
- Framework Provides a library of functions that hides underlying complexity from users.
- Modular design makes it easy to extend
- User friendly interface and reporting options
- Uses Global parameters and variables to compensate for changes in application.


Messages/Logs
Test Management Interface
Keyword Script
Keyword Interpreter
Function Library
Test Driver
Automation Tool
Application Under Test

The Good

- Much easier to extend to larger projects.
- Designed to be highly modular, so changes in the application are easier to deal with.
- Highly customizable, easy to modify across projects.
- Reduced maintenance costs.
- Isolates technical nitty-gritty of test case scripting from test execution.

The Bad

- Adds additional overhead of testers having to understand the framework
- Performs poorly when application changes often as test scripts must be continually modified
- Often sold as a panacea, Automation testing however takes time and effort no matter how you do it.

Different Ways to Design A Framework

- Test Script Modularity Framework
- Data-Driven Automation Frameworks
- Keyword-Driven Automation Framework
- Hybrid Test Automation Framework

Test Script Modularity Framework

- Builds a abstraction layer in front of a component to hide the component from the rest of the application.
- Done by creating small, independent scripts.
- Each script represent modules, sections, and functions of the AUT.
- Small scripts can be combined to from larger tests
- Results in high degree of modularization and maintainability.

Data-Driven Automation Frameworks

- Test input and output values are read from data files (ODBC sources, CVS files, Excel files, DAO objects, ADO objects.
- These values loaded into corresponding variables in captured scripts.
- Test flow navigation coded into test script.
- Thus script is just a "driver," or delivery mechanism, for the data.

Keyword-Driven Automation Framework

- Requires the development of data tables and keywords, independent of the test automation tool.
- Essentially represents a manual test case as a series of keywords or actions.
- In a keyword-driven test, the functionality of the application-under-test is documented in a table as well as in step-by-step instructions for each test.

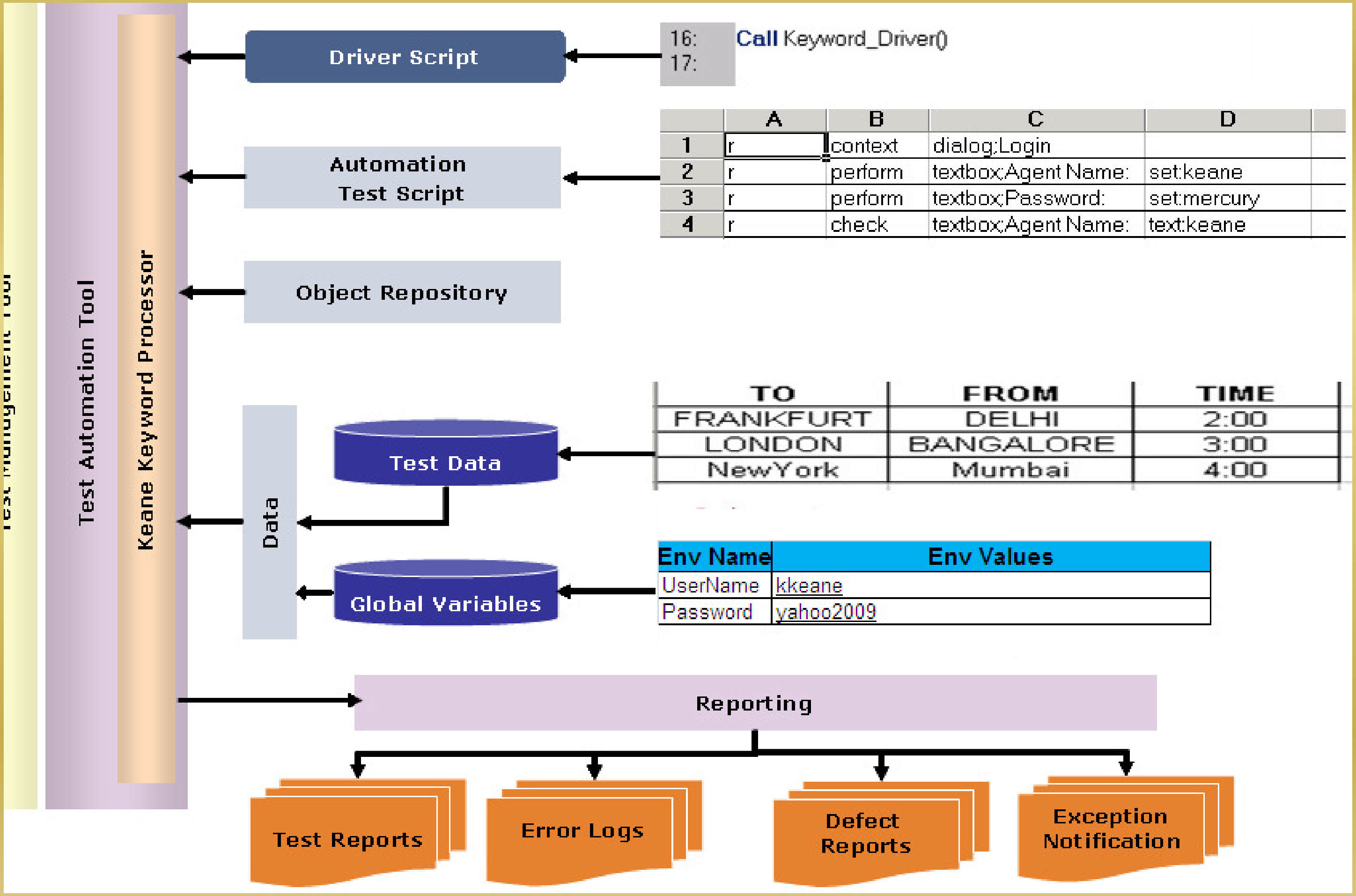
Hybrid Test Automation Framework

- Combination of all of the above techniques, pulling from their strengths and trying to mitigate their weaknesses
- Allows data driven scripts to take advantage of the powerful libraries and utilities in a keyword based approach
- The framework utilities can make the data driven scripts more compact and less prone to failure.
- The utilities can also facilitate conversion of existing scripts to keyword driven equivalents.
- On the other hand, the framework can use scripts to perform some tasks that might be too difficult to re-implement in a pure keyword driven approach.

EXAMPLE AUTOMATION FRAMEWORK

Selenium

FRAMEWORK ARCHITECTURE



```
16:  Call Keyword_Driver()
17:
```

	A	B	C	D
1	r	context	dialog:Login	
2	r	perform	textbox:Agent Name:	set:keane
3	r	perform	textbox:Password:	set:mercury
4	r	check	textbox:Agent Name:	text:keane

TO	FROM	TIME
FRANKFURT	DELHI	2:00
LONDON	BANGALORE	3:00
NewYork	Mumbai	4:00

Env Name	Env Values
UserName	kkeane
Password	yahoo2009

FRAMEWORK COMPONENTS

FRAMEWORK

- Main.rb
- Functionlibrary.rb
- Selenium.rb

ABSTRACT LAYER

- Object Repository
- Keywords

EXTERNAL DATA

- Data Sheets
- Global Variables

Main.rb

- Driver Script That invokes other components of the framework.
- Written in Ruby
- Reads in test scripts (which are in excel format)
- Main.rb invokes application under test.

Functionlibrary.rb

- Invokes functions corresponding to keywords in test-script file from the function library.
- Each Functions correspond to some actions that must be performed . E.g. Buttonclick, VerifyText.
- Function Library can be customized to include additional functions not present in the repository

Selenium.rb

- Selenium.rb holds all the built-in functions of Selenium tool.

Generating The OR

ObjectName	ObjectIdentification	ObjectType
Username	username	TextBox
Password	<password >	TextBox
Login	//input[@value='Login']	Button
Manage	link=Manage	link
Manage Projects	link=Manage Projects	link

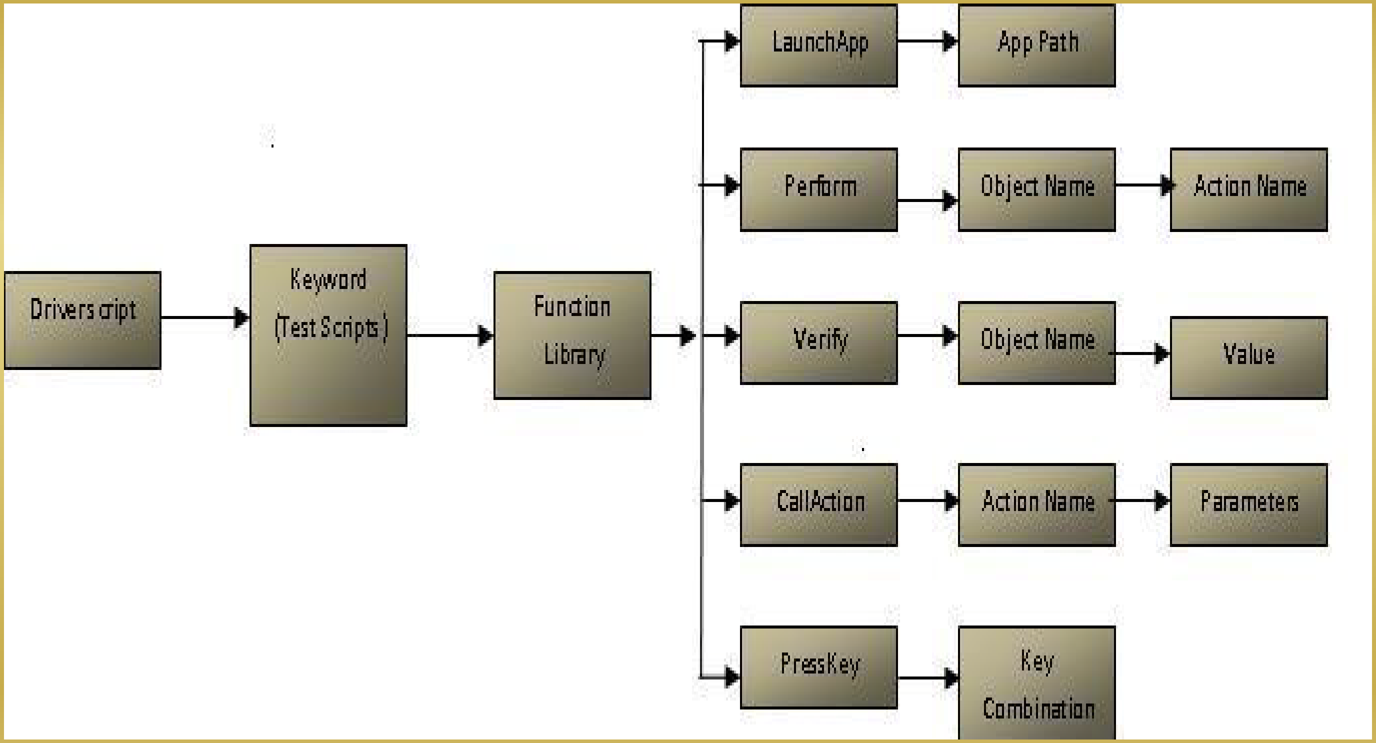
Example Test Script File

Step	Operation	Object	Action
1	Callaction	<Name of excel file that contains test script>	
2	Perform	link;Manage Projects	click
3	Wait	3	
4	Perform	ButtonCreate ;New Project	click
5	Perform	Textbox;Project Name	set:Selenium

Telling Selenium Were Everything Is

File\Folder Name	**Location**
Test Script	<Path to Test Script>
Object Repository	<Path to Object Repository>
Environment File	<Path to Environment File>
Summary Report	<Path to Summary Report Folder>
Screen Shot	<Path to Screen Shot Folder>
Detailed Report	<Path to Detailed Report Folder>

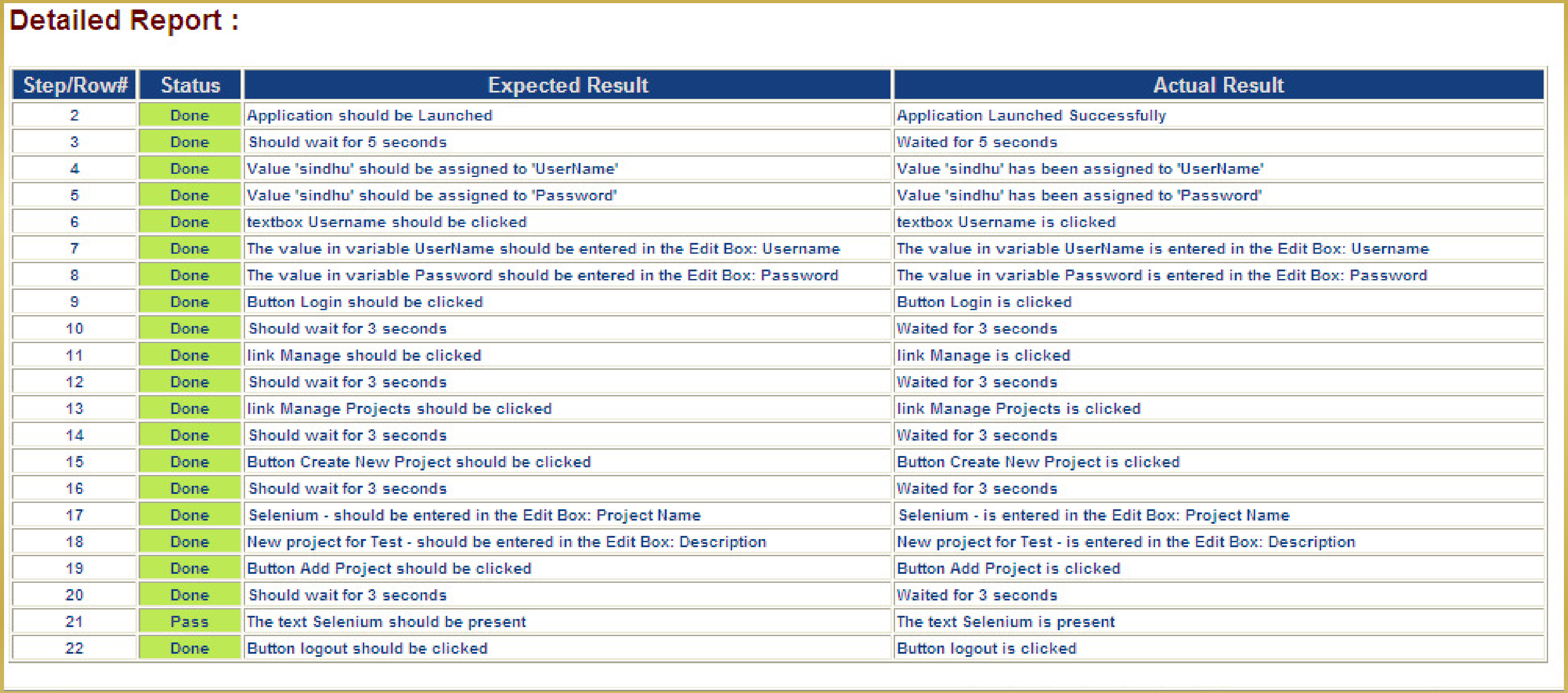
Control Flow



Reporting

- Reports output in Html Format
- Provides two types of reports summary and detailed report
- The summary report provides details of execution duration, test start time and end time
- The detailed reports describe exceptional cases handled, steps passed, and steps failed.

Example Detailed Report



Detailed Report :

Step/Row#	Status	Expected Result	Actual Result
2	Done	Application should be Launched	Application Launched Successfully
3	Done	Should wait for 5 seconds	Waited for 5 seconds
4	Done	Value 'sindhu' should be assigned to 'UserName'	Value 'sindhu' has been assigned to 'UserName'
5	Done	Value 'sindhu' should be assigned to 'Password'	Value 'sindhu' has been assigned to 'Password'
6	Done	textbox Username should be clicked	textbox Username is clicked
7	Done	The value in variable UserName should be entered in the Edit Box: Username	The value in variable UserName is entered in the Edit Box: Username
8	Done	The value in variable Password should be entered in the Edit Box: Password	The value in variable Password is entered in the Edit Box: Password
9	Done	Button Login should be clicked	Button Login is clicked
10	Done	Should wait for 3 seconds	Waited for 3 seconds
11	Done	link Manage should be clicked	link Manage is clicked
12	Done	Should wait for 3 seconds	Waited for 3 seconds
13	Done	link Manage Projects should be clicked	link Manage Projects is clicked
14	Done	Should wait for 3 seconds	Waited for 3 seconds
15	Done	Button Create New Project should be clicked	Button Create New Project is clicked
16	Done	Should wait for 3 seconds	Waited for 3 seconds
17	Done	Selenium - should be entered in the Edit Box: Project Name	Selenium - is entered in the Edit Box: Project Name
18	Done	New project for Test - should be entered in the Edit Box: Description	New project for Test - is entered in the Edit Box: Description
19	Done	Button Add Project should be clicked	Button Add Project is clicked
20	Done	Should wait for 3 seconds	Waited for 3 seconds
21	Pass	The text Selenium should be present	The text Selenium is present
22	Done	Button logout should be clicked	Button logout is clicked