# A Structured Approach for Software Test Automation

**Shripad Hebbar**
**Adobe Systems India Pvt. Ltd.,**
**Bangalore.**

**December 25, 2008**

## Contents

## 1. Abstract

In today’s fast paced business environment, accelerating time to market and increasing cost efficiency are the competitive advantages to the business. Delivering of the products with Rapid Application Development has a key challenge of good software quality and reducing time-to-market in each release. A structured test automation approach is a crucial element in addressing these issues against traditional manual testing or record/playback automation and also achieving - Re-usability, Maintainability, Flexibility, Reliability, Robustness and Portability of the scripts.

This paper explains the application independent automation framework and emphasizes test automation should be approached as a software development.

## 2. Introduction

**Assumptions:**
This document begins with the assumption that the reader will have basic knowledge on the automated testing and / or programming.

Test automation is use of a program to control the execution of tests, comparison of actual outcomes to predicted outcomes, setting up of test pre-conditions and other test control and test reporting functions.

Automated testing can be a valuable tool to gauge and improve the quality of any software product. If it were as simple as recording and playing back test scripts, every company would have a vast array of test suites that covered the testing of their products, but they don't. This document will attempt to explore the criteria for automated testing as well as discuss an architectural framework that has produced proven results.

## 3. Benefits of Automation in RAD

In today’s world of Rapid Application Development, test automation has increasingly become a mission-critical activity. Test Automation can address delivery of high quality product with accelerating time to market and increasing cost efficiency. However, only the test automation alone cannot address these issues. A structured automation approach is required to achieve best out of it.

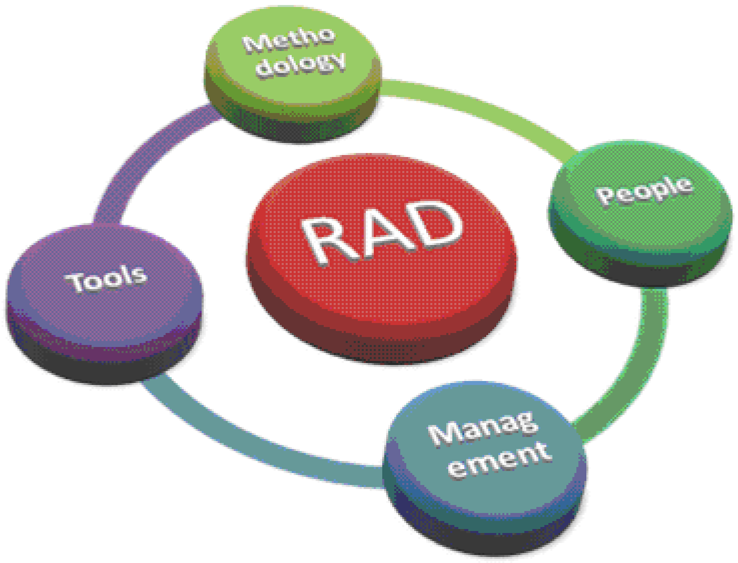


Figure 2. Four Essential Aspects of RAD

## 4. A Structured Approach for Successful Test Automation

### *4.1. Recognize that Test Automation Development as a Software Development.*

A Test Automation should be approached more as a software development in its own right without which it is destined to meet failure in the long term. Automation of software testing is just like all of the other automation efforts that software developers engage in — except that this time, the testers are writing the automation code.
- It is code, irrespective of the automation tool / script in use
- Within an application dedicated to testing a program, every test case is a feature.
- From the viewpoint of the automated test application, every aspect of the underlying application is data.

### *4.2. Methodology for scripting*

In object-oriented programming (OOP) people are familiar with encapsulation and abstraction so that the system stays resilient to changes. Test Automation should be viewed as Software Development, hence both Encapsulation and Abstraction be implemented while scripting.

**Encapsulation**

Encapsulation allows an object to separate its interface from its implementation. The data and the implementation code for the object are hidden behind its interface. Encapsulation hides internal implementation details from users.

**Using functions**
"Function" is a block of code, which may accept parameters, performs a specific task and returns a value (in some cases "void"). Since functions can call other functions, encapsulation on this level is very powerful in reducing maintenance.

**Abstraction Concept while scripting**

Abstraction is a way to remove the association of the behavior of an object with the actual details behind the scenes which implement that object's behavior. This 'abstraction' is usually accomplished through the use of base classes with virtual functions; each derived function provides the details that implement the behavior behind that abstraction.

## Building Blocks for Automation Solution

A structured test automation approach is a crucial element in addressing the reduce time to market with high quality and cost effective solution.

The major Building blocks for a successful Automation Solution are -

1. Automation Framework
2. Automation Tools
3. Processes / Infrastructure
4. Resources (People)

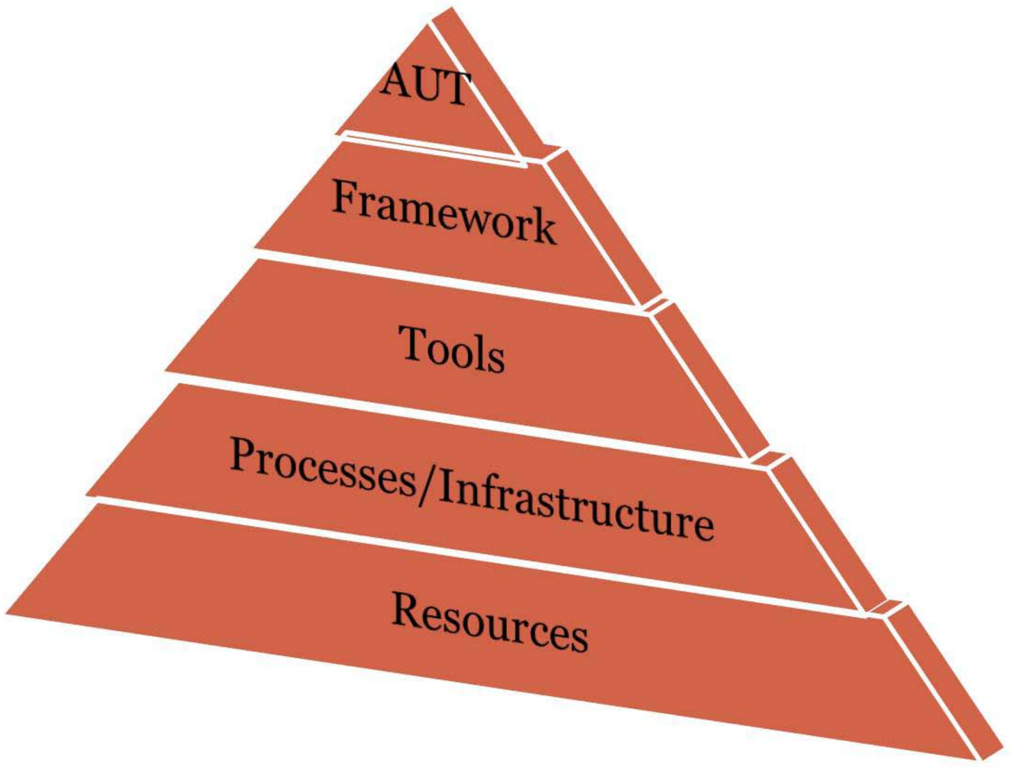


Figure 2. Building blocks for a successful Automation Solution

Test Automation Framework is a major block, success of the automation project enormously depends on it. Providing the right architectural framework for automation development means that the automation code can be used for longer periods of time with less maintenance than a simple record / playback solution. This translates to a significant savings over the course of longer projects, and the ability to more thoroughly test an application and easy to build and maintain scripts with less employee overhead.

## 4.3. Test Automation Framework / Architecture

A Test Automation Framework is a ***set of assumptions, concepts and best practices*** that provide support for automated software testing.

The Framework is an attempt to simplify and speed-up scripting by making use of reusability. The goal is to provide a complete end-to-end automation solution for testing the AUT. As we set out to develop a framework for automating the tests we need to keep the following points in mind -

a. The test framework should be application-independent
   Although applications are relatively unique, the components that comprise them, in general, are not. Thus, we should focus our automation framework to deal with the common components that make up our unique applications and reuse virtually everything we develop for every application that comes through the automated test process.

b. The test framework must be easy to expand, maintain and perpetuate
   The framework we develop should be highly modular and maintainable. Each module developed should be independent of the other modules and should not be affected by any change in other modules. This allows us to expand each module without affecting any other part of the system. The framework should be aimed at being simpler and perpetual. The framework isolates the application under test from the test scripts by providing a set of functions in a shared function library. The test script writers treat these functions as if they were basic commands of the test tool's programming language. They can thus program the scripts independently of the user interface of the software.

### Driving forces for an Automation project

There are many things that can derail an automation project. For instance, whenever there is a change in UI, most the scripts need changes in their code, hence the continuous maintenance. If the test automation to be stable and maintainable through the life of the product and not just something that you throw out when the current project is over, then automation must be approached with a broader view.

Following are the driving forces that distinguish the Automation approach –

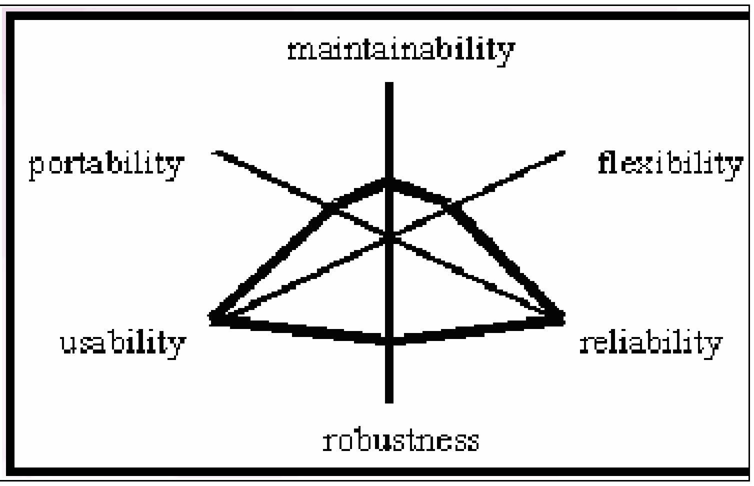
| | |
|---|---|
| - Re-usability<br>- Maintainability<br>- Flexibility<br>- Reliability<br>- Robustness<br>- Portability |  maintainability<br>portability<br>flexibility<br>usability<br>reliability<br>robustness |

Figure 1. Driving forces for a test automation

## 4.4. *Automation Framework Design and Basic Components*

One of the most widely used and proven automation frameworks incorporates the features of both Data driven and Keyword driven frameworks. The main components of this framework are -

**1. Driver scripts:**
These are the main scripts that invoke different controller scripts for running a particular test depending on the test run configuration.

**2. Controller Scripts:**
These scripts comprise of various utility and component function calls and logics to complete a particular test.

**3. Utility Function Library:**
This includes a set of generic functions that can be shared across various scripts and functions. For example, functions for DB Connections, Reporting and Logging etc.

**4. Component Function Library:**
This includes set of functions that are specific to functionalities of the application under test. Controller scripts make use of these functions to execute various test scenarios related to the application being tested. For example, functions for placing an order in an ecommerce application.

**5. Test data:**
Test data pool contains the entire set of application specific data that might be required by the component functions to do one or more iterations of test run.

**6. Object Property (OP)**
OPM provides the input for dynamic object creation. Properties and values of all the necessary objects are stored in OPM and is used for dynamic object creation.
Figure 1 is a diagram representing the design of our automation framework. It is followed by a description of each of the elements within the framework and how they interact.

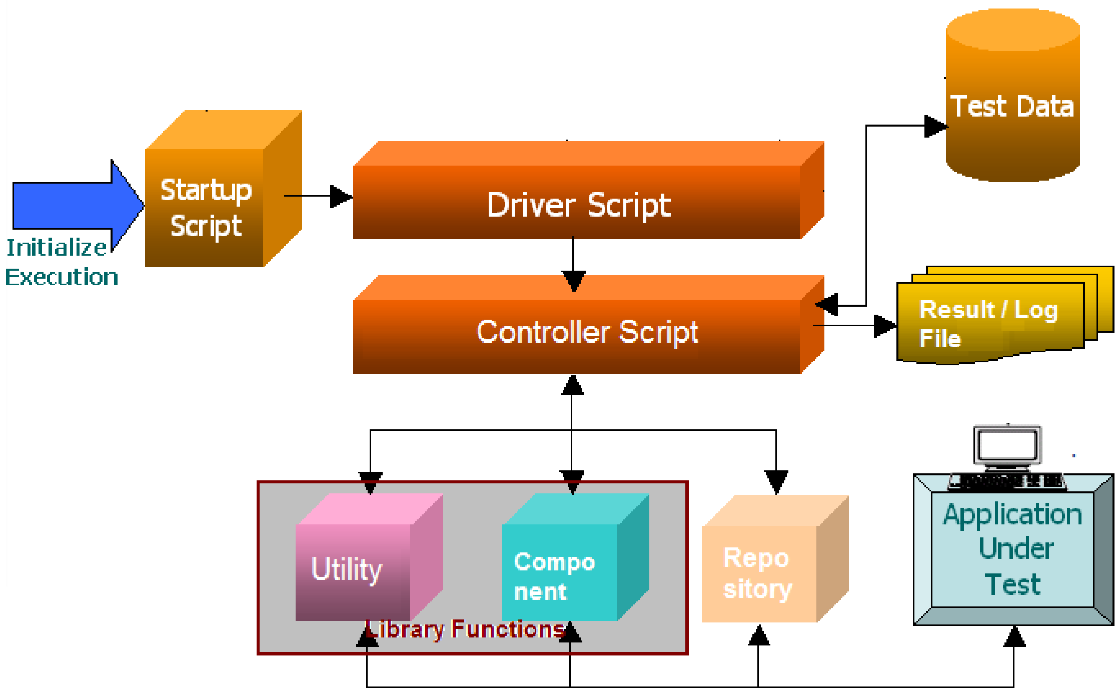


Figure 3: Automation Framework Architecture

### *4.5. Other Blocks of Automation Solution*

- Resources (People)
  - Availability of Skilled Resources
  - Training needs
- Process / Infrastructure
  - Testing Process, Defect management
  - Automation Strategy
  - Automation guideline
  - Test environment
- Automation Tools
  - Tool requirements
  - Open source tools v/s commercial tools
  - Compatibility with AUT
  - Availability of skill set

.

## 5. Conclusion

Properly planned and implemented automated testing can significantly lower project risk and cost. In the course of completing a project, there are many temptations to take shortcuts. While appearing to offer short-term savings, these shortcuts actually turn out to be quite expensive in the long-term.

Anyone in the software industry understands that solid, efficient automated testing can greatly reduce the risk of a project and a product. If that automation is well designed and implemented, it can not only reduce the risk of the current project, but can actually reduce the risk and cost of future projects.

## 6. References

1. Abstractions in Test Code - http://www.autotestguy.com/archives/design_patterns/
2. Choosing a test automation framework - http://www.ibm.com/developerworks/rational/library/591.html
3. Cem Kaner - Improving the Maintainability of Automated Test Suites
4. Dan Young TEST AUTOMATION: AN ARCHITECTED APPROACH

********************