ALLIANCE
GLOBAL SERVICES

Guidelines to create a
ROBUST TEST AUTOMATION FRAMEWORK

ABSTRACT

Mission critical software undergoes rigorous functional tests, especially supported by automated testing frameworks. Automating these frameworks and maintaining quality software releases are critical to business performance. Enterprises often face the dilemma of balancing costs and managing resources to ensure that automation frameworks cover all the business scenarios and the applications delivered are error-free.

By implementing the appropriate automated testing framework, enterprises can significantly increase the speed and accuracy of the testing process, provide a higher return on investment (ROI) from software projects and systematically minimize risk.

This White paper presents the benefits of different Test Automation Frameworks, their design components, governance models, and maintenance considerations. The paper concludes with a checklist of key elements to successfully maintain a Test Automation Framework.

EVALUATING TEST AUTOMATION FRAMEWORKS

Automation Frameworks are constantly evolving. This section describes the four key capabilities to look for in any test automation framework.

- **Script-less representation of automated tests**: The testing framework should offer point-and-click interface for accessing and interacting with the application components under test—as opposed to presenting line after line of scripting. Testers should be able to visualize each step of the business scenario, view and edit test cases intuitively. This will shorten the learning curve for testers and help QA teams meet deadlines.
- **Data driven tests**: A key benefit of automating functional testing is the ability to test large volumes of data on the system quickly. But you must be able to manipulate the data sets, perform calculations, and quickly create hundreds of test iterations and permutations with minimal effort. Test Automation Frameworks must have capability to integrate with spreadsheets and provide powerful calculation features.
- **Concise reporting**: The ability to run high volume of tests is of little benefit if the results of the tests are not easy to understand. The framework must automatically generate reports of the test run and show the results in an easy-to-read format. The reports should provide specifics

about where application failures occurred and what test data was used. Reports must present application screen shots for every step to highlight any discrepancies and provide detailed explanations of each checkpoint pass and failure. Reports must also be easily shared across the entire QA and development teams.

- **Coupling Test Automation Framework and the application-under-test (AUT)**: A key factor to consider while designing a Test Automation Framework is to identify if it is to be tightly integrated or loosely coupled with the application-under-test (AUT). There are merits and de-merits of each approach.
    - If the Test Automation Framework is intended for a defined set of applications and if applications-under-test are stable, then the Test Automation Framework should be tightly coupled with the application-under-test (AUT). In this case, UI validations and specific business processes are implemented within the framework, so that QA teams only have to configure tests based on their business scenarios.
    - If the Test Automation Framework is designed for applications that are in development or for future use, then the Test Automation Framework should be loosely coupled with the application-under-test (AUT) to provide flexibility for extension and derive ROI from the existing automation effort. In this case QA teams have to configure every validation or checkpoint which makes the test configuration complex and time consuming.

TYPES OF TEST AUTOMATION FRAMEWORKS

The size and complexity of the project determines your framework design. Five types of test automation frameworks are discussed here:

- The modular testing framework
- The test library testing framework
- The keyword-driven and table-driven testing framework
- The data-driven testing framework
- Hybrid testing framework

THE MODULAR TESTING FRAMEWORK

A modular Testing Framework involves creation of small, independent scripts that represent the modules, sections, and functions of the application-under-test (AUT). These scripts are then used to construct larger tests to realize specific business scenarios.

Modular Testing Framework is suitable for automation of large, stable applications.

THE TEST LIBRARY TESTING FRAMEWORK

The Test Library Testing Framework divides the application-under-test (AUT) into procedures and functions instead of scripts.

The Test Library Testing Framework is suitable for automation of small to medium, stable applications.

THE KEYWORD-DRIVEN AND TABLE-DRIVEN TESTING FRAMEWORK

Keyword-Driven Testing and Table-Driven Testing are interchangeable terms that refer to an application-independent automation framework. This framework requires the development of data tables and keywords, independent of the test automation tool used to execute them and the automation script that "drives" the application-under-test (AUT) and the data.

Keyword Driven Testing Framework is suitable for automation of small applications.

THE DATA-DRIVEN TESTING FRAMEWORK

Data-driven testing is a framework where test input and output values are read from data files (data pools, ODBC sources, CVS files, Excel files, DAO objects, ADO objects etc.) and are loaded into variables in the manually coded scripts. This framework requires variables to be used for both input values and output verification values.

Data Driven Testing Framework allows large number of tests to be executed quickly using the automation scripts instead of creating large number of scripts for testing individual test conditions. This framework is also suitable for applications that are under development.

HYBRID TESTING FRAMEWORK

The most commonly implemented framework is a best combination of all the techniques. Hybrid Framework combines the best of Keyword Driven and Data Driven frameworks.

Hybrid Testing Framework allows data driven scripts to take advantage of the powerful libraries and utilities that usually accompany a keyword driven architecture. The framework utilities can make the data driven scripts more compact and less prone to failure. Tests are fully scripted in a Hybrid Testing Framework thus increasing the automation effort. Record and play is not used to create test scripts.

Hybrid Testing Framework also implements extensive error and unexpected windows handling. Hybrid Framework is used for automation of medium to large applications with long shelf life.

DEFINING YOUR AUTOMATION OBJECTIVES

It is very important to define the objectives of test automation before designing the Test Automation Framework. This will help the team choose the right toolset, the right foundation framework for the current and future business needs. Some of the questions that must be answered are:

- What is the shelf life of the framework?
- What percent of application will be automated using the framework?
- How stable is the application?
- Should the Automation Framework be tightly or loosely coupled with the application-under-test (AUT)?
- Who is the target audience for using Test Automation Framework?
- What are testing skills of the target audience?
- How much is the enterprise willing to invest in automation?
- What is the expected ROI from the automation efforts?

Discussions of these automation objectives will govern the functional testing automation effort and ROI. Common automation considerations for a Test Automation Framework include:

- Maximizing test coverage
- Scale for future requirements
- Reliability and Consistency of results
- Duration of execution
- Identification of all regression testing defects
- Integration with nightly continuous builds
- End to end testing capability to cover business scenarios

AUTOMATION FRAMEWORK – TARGETED BENEFITS

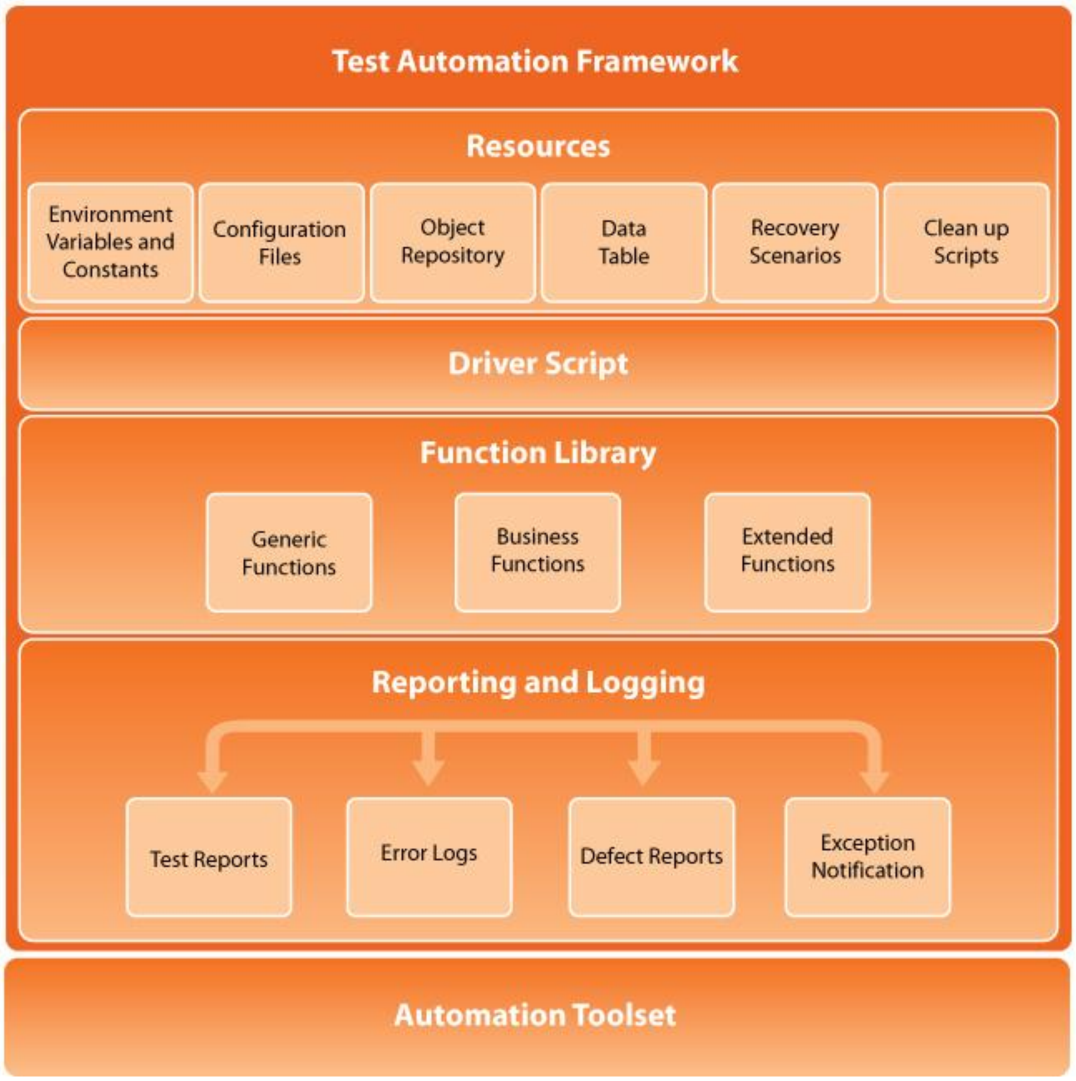
A framework is developed on top of a complex internal architecture of the automation tool and enforces the required standards for implementation. It ensures structured design and organization of automation code into components. Test Automation framework must target the following factors:

- Design that enhances the outcome of code scripted
- Faster test scripts generation
- Longer automation code life
- Ease of maintenance
- Reusability of test code
- Data driven test
- Software migration support

- Extended reporting capability
- Suitable coupling of the Test Automation Framework and the application-under-test (AUT)

COMPONENTS OF A WELL DESIGNED AUTOMATION FRAMEWORK

A Test Automation Framework design must be modular, easy to maintain, reusable and leveraged across multiple projects to maximize the value of the test automation suite. The best practices for a good automation framework must have the following core components:



Object Repository

The Object Repository contains the objects for the GUI based validation testing. Any change in GUI objects is modified at a central location, thereby avoiding rework of scripts.

Function Library

Function Libraries form the basic building blocks of an automation test suite and define the common set of reusable functions. A Function Library must contain the generic set of reusable functions that are usable across the applications as well as business functions specific to the application.

Global Variables/Constants

Global Variables and Constants define the variables and constants that will be used across the automation test suite. These entities are used to tune the automation test suite and provide the flexibility to modify entity references dynamically and ensure test suite maintainability.

Test Script

Test scripts are a critical component in the Test Automation suite. They contain the actual code logic to perform and validate a specific business or functional scenario.

Test Data

Test Data is application specific data used to test the application-under-test (AUT). As a best practice, common data (non-application specific data) should be dynamically generated within the script rather than referencing a Test Data sheet which in turn refers to a File or Content on some other file medium.

Recovery Scenarios

Recovery Scenarios define and guarantee the robustness of the scripts. They are defined and implemented to handle unexpected exceptions where the script execution may come to a halt. Recovery Scenarios help the automation script flow to follow a definite execution flow or path whenever unexpected exceptions arise during the script execution.

Cleanup Scripts

Cleanup Scripts play the critical role of releasing all the resources like object instances, file instances and return the automation suite back to a base state. Cleanup Scripts should be developed for both conditions – after successful execution of the script and at end of the recovery scenario invocation.

Configuration File

The global settings and application access parameters are maintained in a Configuration File. Any setting or parameter which changes with the test environment setup is part of the configuration file and any settings that remain constant irrespective of the test environment setup should be considered as part of Library Constants.

Reporting Mechanism

A Reporting or Logging Mechanism should be configured so that test results generated during the execution phase are easily accessible and understood.

Log generation is an important part of execution. It is very important to generate debug information at various points in a test case. This information can help find problem areas quickly and reduce the bug-fixing time.

Automation Governance

Automation Governance is a critical aspect of Test Automation Framework design. Automation governance will ensure that all areas of the testing methodology are kept consistent, regardless of the individual developing the script, or executing the test. Governance defines the process for development, testing, management and control of Test Automation Framework and makes functional automation successful. Key Automation Governance parameters are:

- Test Suite Structure
- Script Structure
- Script Versioning
- Coding Standards
- Script Naming Convention, Objects naming convention, identifying mandatory properties, basically OR should be under Governance for maintainability
- Error and Exception Handling
- Independent testing of automation framework
- Framework for Test Data creation and management

CHECKLIST FOR A SUCCESSFUL TEST AUTOMATION FRAMEWORK DESIGN

Even when there is clear evidence that automating a testing effort is economically justified, it can be difficult to determine how best to transition to an automated testing process. Here are five fundamental principles to consider while designing a Test Automation Framework:

1. **Complete a Test Plan document**. Understanding the goals of the application to be tested is critical to the success of any testing effort. This includes thorough up-front planning to verify that test requirements are implemented correctly.
2. **Define the business scenarios** that need to/can be automated. It is probably impossible to automate all aspects of a test plan. Automated testing should be focused around the complex and critical business processes. Many organizations find that they are automating nearly 60 percent of their total number of test cases, leaving 40 percent of tests to be conducted manually.
3. Create automated tests which require **minimal or no scripting by QA** teams. QA teams should be able to visualize each step of the business scenario for automation.
4. Do **data-driven tests to expand test coverage**. Test Automation Framework should provide the ability to create data-dependent tests that use specific keywords stored in Excel spreadsheets to populate fields in an application. This capability allows QA teams to drive massive volumes of test data through an application.
5. **Add verifications** into the Test Automation Framework. The criteria should include verifications of the front-end of the application, the middle tier, or the back-end database. Database verification confirms the values stored in the database and verifies transaction accuracy and the data integrity of records that have been updated, deleted, or added.

About Alliance Global Services

Founded in 1994, Alliance Global Services is a software solutions firm providing the full life cycle of capabilities from strategic guidance through product development, quality assurance - testing, and application maintenance outsourcing. With our proprietary RIGHTLINE methodology, we focus on cost effectively building and maintaining software products and applications for software, content and data intensive companies whose businesses depend on software innovation.

Corporate Headquarters
Six Tower Bridge
181 Washington Street, Suite 350
Conshohocken, PA 19428
Phone: 610 234 4301
Fax: 610 234 4302
Email: info@allianceglobalservices.com
Website: www.allianceglobalservices.com

Headquartered in suburban Philadelphia, the firm's worldwide offices include: Boston, Chicago, New Jersey and Hyderabad, India.