KMG
APPLICATION TESTING AUTOMATION
AN APPROACH WHITE PAPER

KEY MANAGEMENT GROUP, INC.

The Competent People

www.kmgus.com

Proprietary Notice

Contents

1 OVERVIEW

1.1 ABOUT TESTING AUTOMATION

Software testing is an essential, requisite, costly, and time-consuming activity in the software development life cycle. As is true for software development in general, reuse of common artifacts can provide a significant gain in productivity.

Test Automation becomes increasingly critical and strategic necessity. Assuming the level of testing in past was sufficient, how do we possibly keep up with this new explosive pace of web enabled deployment while retaining satisfactory test coverage. The answer is more people or greater level of automation.

Automation plays a vital role for applications/products intended for customers who undergo continuous customization and number of releases. One can automate the regression freeze suite and run them on needy basis when new features are implemented. Moreover, as testing deals with testing the application or the system under a variety of platforms, configurations and circumstances, automation of execution-related activities offers another potential source of savings in the testing process.

A test automation framework is a set of assumptions, concepts, and practices that provide support for automated software testing. This document intends to provide a detailed insight on architecture and design of automation frame work suggested for web/window based application that KMG follows.

1.2 SCOPE OF THIS DOCUMENT

The objective of this document is to showcase KMG's automation testing approach and practice for all Development and Maintenance projects in KMG. This white paper also covers the complete details on architecture, design and implementation of automation framework.

1.3 ABOUT KMG

KMG is a global software development company, which provides premium IT solutions worldwide using Microsoft, IBM & Java Technologies. In a wide-open sea of countless software development companies, KMG distinguishes itself as a company driven by excellence.

KMG was established in 1990 and is among Top 10 fastest growing Indian-owned companies in the US. It is also rated among top 50 software companies in India. It has a Dun & Bradstreet rating of "Good- 2A1".

KMG's onsite-offshore model and industry expertise enables the company to enter into long-term, mutually beneficial strategic partnerships with many Fortune 500 companies.

KMG has its headquarters in NY with 4 Offshore Development Centers in India (Bangalore, Delhi, Chandigarh and Kolkata) and a sales and development center in New York.

KMG has around 40 professionals in the US supported by another 250 in India with expertise that covers Microsoft.NET technologies, J2EE, Mainframe, IBM iSeries (AS/400) and Software Testing.

KMG provides software maintenance, development and testing solutions to large and medium sized insurance, banking, financial service, healthcare and government organizations throughout the world. Indeed, the company has aggressively captured a niche market in property and casualty insurance sector in the USA.

2 PRE-REQUISITE OF TESTING AUTOMATION

In order to achieve successful automation, it is very important to determine when to start automating the application.

- **Stability of Product/Application** – First thing that needs to be ensured is that product/application should be fairly stable in terms of functionality. Even if it is slated to incorporate new features, the new features should not disturb existing functionality. Little changes are acceptable but frequent changes to UI or functionality will lead to extra effort in overall automation and that does not make any sense.

- **Scope of Automation has been defined** – Before setting out to automate the testing of your application/product, it is essential to define the scope/intended coverage of the automation tool. The scope may encompass functionality testing, regression testing or simply acceptance testing. You can even select to automate the testing of certain particular features or certain selective test cases of different features.

- **Individual test cases to be automated have been identified** – Automation suite should be looked upon as a base line test suite to be used in conjunction with manual testing, rather than as a replacement for it. It should aim at reducing the manual testing effort gradually, but not doing away from manual testing altogether. Setting realistic goals in early stage of test automation for achieving long-term success. So, even if after defining the scope of automation in terms of acceptance/regression testing, it needs to be made sure that following kind of test cases are eliminated from the scope of automation.

    - Testcases that are long and complicated and require manual intervention in between.
    - Tescases that take tremendous amount of time in automation and it is difficult to ensure re-usability even if they are automated.
    - Testcases pertaining to Non-functional testing (Security Testing, Load Testing, Accessibility Testing etc.)

- **Testcases have been fine tuned** – The testcases need to be fine-tuned for automation. The expectation level from the test cases for automating is widely different from expectation from manual testing point-of-view. All test cases need to be equipped with proper test data. E.g. – if there is a test case for uploading a file, then it should explicitly tell which file to upload.

- **The right tool has to be decided** – There are numerous tools available in the market. A careful effort has to be made into deciding which tool would be most suitable for automating the testing of your product/application. The following criteria can be useful in making the decision:

    - If testing to be automated is web based or Windows based. Some tools only support automation of web based applications.
    - If testing to be automated is GUI based, it might be preferable to use a tool like QTP, WinRunner, or UTP. Every tool will have its technical limitation that prevents efficient automation. So, it is necessary to evaluate testing tools for critical interfaces of the application.
    - Is the automation suite required to work on different platforms?

- **The right mode (script recording/script development) has been decided** – Most of the GUI automation tools have a feature called 'record and playback'. Using this feature, you execute the test manually while the test tool records the testing steps. It then generates a script that you can run to re-execute the test. Script development, on the other hand, implies writing the scripts for running the test cases in the language used by the tool.

3 AUTOMATION TESTING METHODOLOGIES

There are various methods or frameworks to automate an application testing which can be employed as per the need of the application. Every framework has its advantages and disadvantages. We can choose the Record & Play Back method or descriptive programming after analyzing the feature provided by any automation tool and the limitation of the product/application. Following basic methodologies are used for automating an application:

3.1 RECORD & PLAYBACK

- Tester manually records each step, and then plays back the Recorded script. During recording, automation tool reads the user input and generates the scripts.

3.2 ACTION DRIVEN FRAMEWORK

- All user actions converted to reusable actions
- Test case logic defined in test scripts
- Any new scripts make use of these actions

3.3 DATA DRIVEN FRAMEWORK

- Pass test data using test spreadsheets
- Test case logic resides in test scripts
- Test scripts same as record / playback

3.4 KEYWORD DRIVEN FRAMEWORK

- All business actions converted to reusable function libraries
- Test logic driven though external files
- Test data passed through external files
- Flexible to accommodate any changes with least amount of work

4 ADVANTAGES AND DISADVANTAGES

S.No.	Automation Methodologies	Advantages	Disadvantages
1	Record & Playback	1. Fastest way to generate script 2. Automation expertise not required	1. Re-record required on UI Change. 2. Lacks programming logic 3. Number of lines of code is highest. 4. Test data is hard coded into the script 5. Test resources are heavy
2	Action Driven Framework	1. Code is re-used in scripts 2. Easy implementation	1. Difficult to change script. Re-mapping of actions required on UI change 2. Lacks programming logic 3. Test data is hard coded into the scripts Usage of Active Screens makes test resources heavy
3	Data Driven Framework	1. Faster way to generate script 2. Test Data separated from script 3. Knowledge in scripting language required to access data from script	1. Difficult to change script in case of any Change 2. Lacks programming logic 3. Number of lines of code is same like record & playback.
4	Keyword Driven Framework	1. Provides high code re-usability 2. Execution flow controlled using external files 3. test data change doesn't trigger change in scripts 4. Customizable Test Results Logging possible using XLS/CSV/XML/TXT file formats	1. Automation expertise required to write drivers and function libraries

5 AUTOMATION GUIDELINES

5.1 REQUIREMENTS

The following are some of the basic requirements that are expected from an effective Test automation:

1. It should be easy to maintain and update the scripts with newer releases or changes done to the application.
2. There should be an easy way to select test scenarios at a more granular level in order set up more targeted regression
3. Recovery should be built into the test scenarios. In an event the machine on which the automated test suite is running crashes, it should be possible to restart testing from the last executed test scenario
4. It should be possible to execute tests from different machines at the same time in order to gain time efficiencies
5. The scripts should be completely parameterized.
6. The scripts should work on all environments with little change. There should be an interface to select which environment is being targeted

5.2 DESIGN PRINCIPLES

As with all designs, it is essential to decide some key design principles that guide the rest of the design and helps make other design decisions quicker and in the right framework. Following principles may be used to guide the design

5.2.1 MAINTAINABILITY AND EXTENDIBILITY

The automation suite should be easily maintainable by the ongoing testing team. This would imply that the design of individual elements be kept simple and intuitive. Nomenclature of the scripts and functions be kept simple and the code be well commented. The test suite also needs to be extendible. This would imply building a common framework and standards on which more applications or test scenarios can be easily added. All applications and test scripts should strictly adhere to the framework and standards.

5.2.2 EASE OF CUSTOMIZATION

All data should be parameterized into constants or data files. A tester should not have to modify a test script in order to setup a test run. The only things that could need changes during a setup of a test run should be a few constants and/or the data tables to setup the scripts with the right data. Even with the constants and data tables, the changes should be easy and well defined.

5.2.3 SCALABILITY

The test suite should scale in the sense that it should be possible to execute multiple scripts from multiples machines. There should not be a conflict in resources or design that will prohibit running tests in parallel. This is essential to get the real gains in automation by shrinking the total timeframe using multiple instances of the test suite.

5.2.4 REUSABILITY

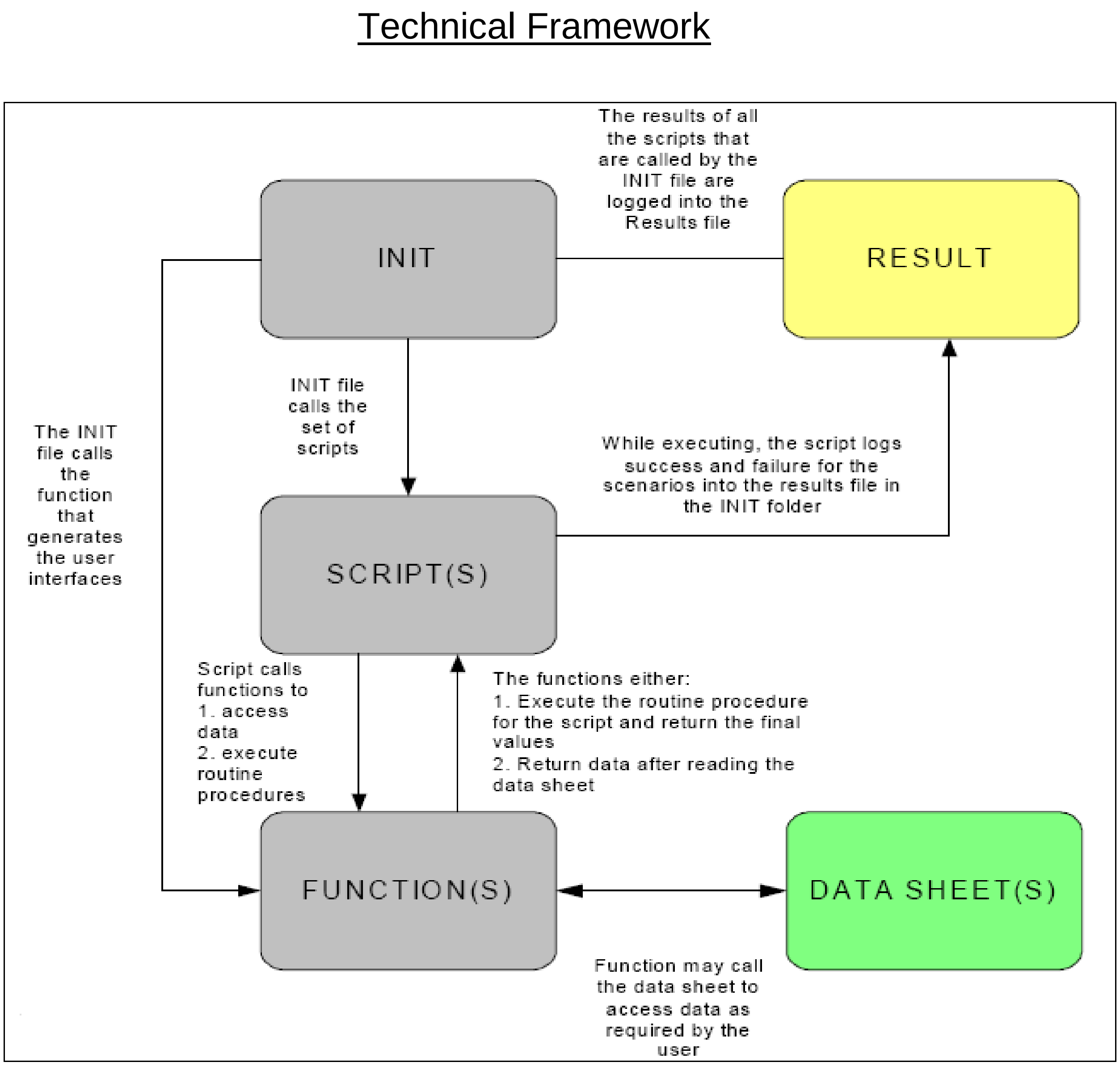
The automation scripts once created should be reusable across the different application. Code for common functionality can be created at one place and a tester should not have to re-write the code in case any common functionality is encountered in same or any other application.

6 AUTOMATION ARCHITECTURE

6.1 TECHNICAL FRAMEWORK

The **init** script initiates test execution for any application in ART (Automated Regression Test). Currently for all applications in ART, there is a corresponding **INIT** file. The INIT file calls the function for user interface and also calls different scripts in the application. The scripts while executing, call the functions to execute routine procedures and to obtain the data from the data files. As the scripts execute, they write the results into a log file that belongs to the **INIT** file. When the suite completes the testing, the results folder (containing the results log) presents in the **INIT** folder may be opened and the logs analyzed to obtain the test results.

Below snap shot describes the automation framework:


Technical Framework
The results of all the scripts that are called by the INIT file are logged into the Results file
INIT
RESULT
INIT file calls the set of scripts
The INIT file calls the function that generates the user interfaces
While executing, the script logs success and failure for the scenarios into the results file in the INIT folder
SCRIPT(S)
Script calls functions to 1. access data 2. execute routine procedures
The functions either: 1. Execute the routine procedure for the script and return the final values 2. Return data after reading the data sheet
FUNCTION(S)
DATA SHEET(S)
Function may call the data sheet to access data as required by the user

6.2 Design Elements

Following are the design elements used in Automation Framework:

6.2.1 Recovery

Under normal test conditions, the test suite might stop execution for a variety of reasons For example: the test suite not able to find a critical object like the login screen for further execution. Hence the suite is designed in such a way that in the event something causes the suite to stop, on restart it picks up from where the suite failed. This allows the tester to run the suite effectively with very little re-configuration.

6.2.2 File structure

The file structure for a module allows the future testers/developers easy and quick access to all the files that are required to be tested/modified. The file structure for a module contains an initialization (INIT) file (the file that contains the configuration settings), a constants file, a data file (to store the data required to run a script, data combinations, etc.), the code files (one file for each flow that is identified) and the functions file (that would contain the common pieces of code used by the scripts).

6.2.3 Coding standards

In order to make the suite files easy to maintain, more readable and allow interchangeability of code, it is important to have a standard set of instructions to be used across the projects while writing the code for automated scripts. These coding standards should be defined before the developers start writing code and after completion of development, each application code may be code reviewed by a group of people to ensure adherence to these standards.

6.2.4 Targeted regression

This is the ability/functionality provides the testers to test any specific flow(s) with a specific data combination(s) out of the whole suite. This may be done by using the data excel sheet.

6.2.5 Environment Independence

The suite can be configured to run on any environment by configuring the URL in the constants file.

7 CASE STUDY

KMG has successfully automated the application of one of its big clients. KMG's automation experts were involved in Automation planning, designing framework, creating scripts, executing scripts and report generation.

7.1 CUSTOMER DETAILS

The customer is a multi-line property/casualty insurance group. Customer offers a broad range of insurance products including commercial property insurance, worker's compensation insurance, surety bonds and automobile insurance. It is full service company, also providing claims and loss control services.
Customer uses an application to administer the policy life-cycle, from application intake to rating and underwriting, from policy issuance to renewal, and everything in between. The policies issued using this application is further uploaded in PMS, which essentially is the policy management system, based on **mainframes**.

7.2 BUSINESS PROBLEM

Challenge 1

The Customer and his team did not have any prior experience in the Testing automation. KMG therefore provided a demo on the existing application by automating few test scenarios and that demo helped the customer getting idea on how the business would get benefit after automating the application.

Challenge 2

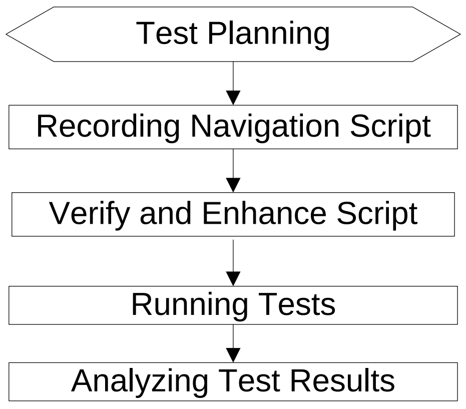
The customer's application had two parts; one is front end part made in .Net technology and other is mainframe part which is a command line interface (CLI) based application. Automating a mainframe application include various challenges. KMG provided the solution by automating both Mainframe and GUI based application.

7.3 KMG'S AUTOMATION APPROACH

KMG provided the automation testing solution in 2 phases; Creation and Verification of test scripts and Execution of test scripts:

7.3.1 APPROACH FOR SCRIPT CREATION

KMG performs the following steps for script creation:



1. **Test Planning:** KMG describes the test in detail prior to automation. It includes the exact steps to be followed, data to be input, and all items to be verified by the test. Test planning includes following activities:

   - Setting up Automation Test Plan and Strategy and estimating the time, effort and capacity needed for Automation
   - Senior Automation Experts start working on the Automation Framework that is to be designed for automation of the existing application Test cases.
   - Also started ramping up core team members on Automation tool and application. This immensely helps in scaling up the activity for Test Automation once framework is ready.
   - It is expected that questions, data issues, and application issues will be encountered and need to be addressed.
   - This phase in the automation approach is intended to clarify the test cases to be automated, familiarize the automation team with the application, and get potentially blocking issues to automation raised up and addressed as early in the project as possible.

2. **Recording Tests:** KMG graphically records each step it will perform as it navigates through the application. A step is any user action that causes or makes a change in the web site, such as clicking a link or image, or entering data in a form.
   When tests are recorded, the scripts are generated by automation tool

3. **Verify and Enhance:**

   - Inserting Verification Points into the test will enable the search for a specific value of a page, object or text string, which will help in identifying if the application is functioning correctly.

     A spreadsheet shall be created for each test wherein test data and verification points shall be entered. Verification points are nothing but the expected data which are required to be put in the spreadsheet. Below is the snapshot of the spreadsheet for reference:

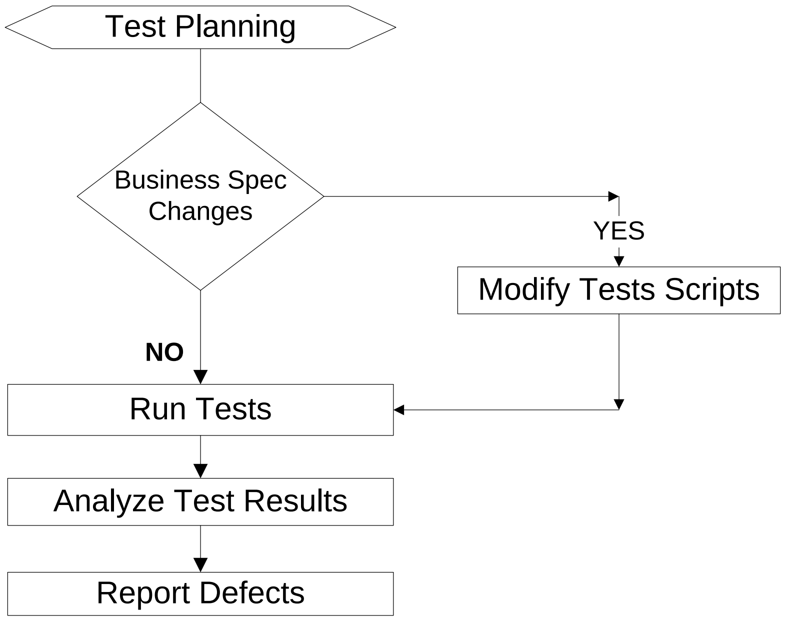
   - Broadening the scope of the test by replacing fixed values with parameters will allow checking the application performing the same operations with multiple sets of data.
   - Adding logic and conditional statements to the test enables addition of sophisticated checks to the tests. The recorded test script needs to be modified in order to replace the fixed value with the data from spreadsheet.

4. **Running Tests:** KMG run a test to check the behaviour of application and ensure one successful execution. While running, Automation Tool connects to the application and performs each step in the test as per the spreadsheet. After the successful execution, Automation test script shall be added to automation test pack.

5. **Analyzing Test Results:** KMG examines the test results to pinpoint defects identified in the script and modify the test scripts and then re-run the test script, if required.

7.3.2 Approach for Test Execution

KMG performs the following steps for test execution:



1. **Test Planning:** Prior to executing the test scripts, a detailed test execution plan is created which highlights the number of test scripts to be executed in that Test execution cycle for a particular test environment.

2. **Modifying Tests:** Modification in the automation tests scripts, if required, is done in order to incorporate the changes in the business specifications of the application.

3. **Running Tests:** As the test scripts and spreadsheet are ready, KMG reviews and update the data to be tested with regard to the spreadsheet and run the test.

   KMG executes all tests identified for that particular test execution cycle for any number of sets of data.

4. **Analyzing Test Results:** KMG examines the test results to pinpoint defects in the application that are identified by the automation scripts.

5. **Reporting Defects:** KMG raises the defects in the “Test Tracker Tool” currently being used by manual testing team as it encounters failures in the application when analyzing test results.

8 SUMMARY

In SDLC we encounter the situation where we need to repeat the sequence of actions many times during testing. For example - smoke testing, regression testing, acceptance testing, compatibility testing etc.

Test automation provides significant benefits in terms of saving the test execution cycle time. The saving can result from enhancements to the speed of test case execution as well as the ability to carry out testing operations beyond regular work shifts. The use of framework can further increase the saving by reducing the development and maintenance effort for automation scripts.

Automation testing is often the most cost effective method for software products that have a long maintenance life, because even minor patches over the lifetime of the application can cause features to break which were working at an earlier point in time.

Software testing using a test program will generally avoid the errors that humans make when they get tired after multiple repetitions. The test program won't skip any tests by mistake. The test program can also record the results of the test accurately. The results can be automatically fed into a database that may provide useful statistics on how well the software development process is going.

Automation software testing is the planned and systematic set of activities. The goal of automation testing is to make sure that same test steps and data are executed on current or any subsequent build. Automation testing is highly recommended for the projects which are wide in scope and stable in nature.