Capgemini

# Automation Test Framework for Testing Projects

# Contents

# Introduction

Testing is a quantitative measurement of the quality of a software been delivered to the customer. It can be either performed manually or automated using a tool in-order to reduce the manual effort. Software schedules are mostly doable but as they progress slippage typically occurs due to challenges like human errors, repeatability etc. and this is where automation plays a major role.

Automation can be effectively applied to nearly every aspect of the QA process:

- Unit tests within developers IDE
- Functional testing of software components
- Negative testing that attempts to force errors through invalid inputs
- Stress testing for determining functional capacity of the software and platform
- Performance tests to measure compliance with product requirements
- UI tests or instrumentation to determine usability of the software
- Regression tests to detect new defects in existing code at each new build
- Test case and test script generation to improve test coverage

Automation just as manual testing has its own advantages and disadvantages. There are various methodologies to implement automation from which user can use the best suited one for their application under test.

# A Quick Revisit To Automation

## Benefits of Test Automation

Automation has various benefits and few can be listed as below:

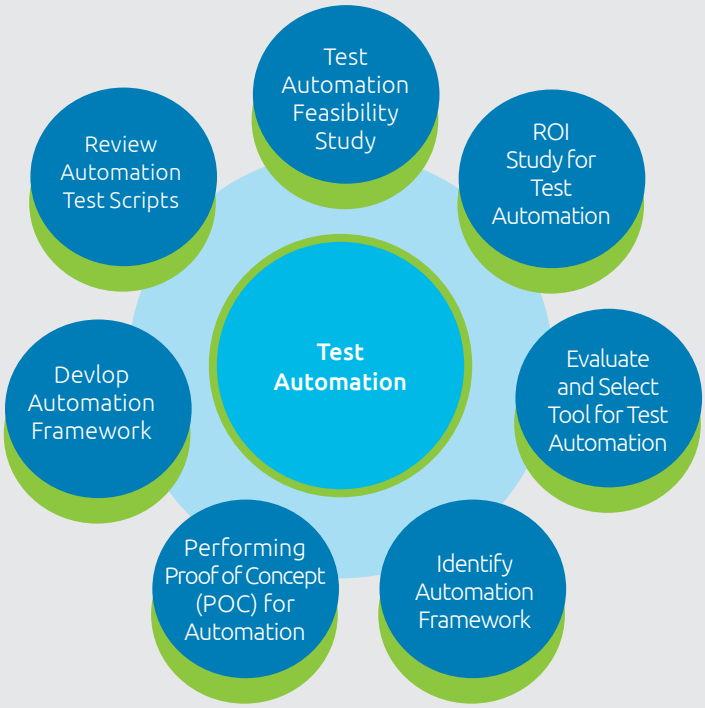
- Improved Reliability and Repeatability
- More efficient workflow
- Enabling Agile methodologies
- Saving time on next project

## Challenges of Test Automation

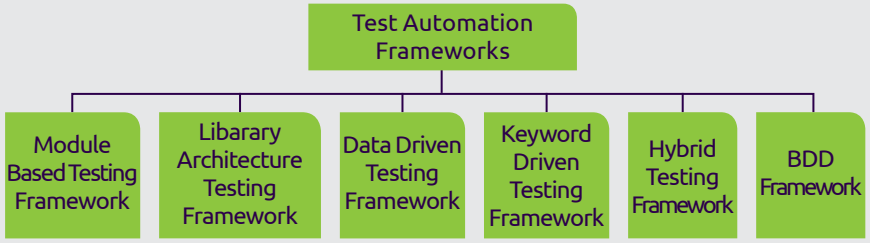
Automation is certainly not a substitute for poor testing practices within the organization it cannot alter the "physics" of testing. It cannot guarantee the absence of bugs nor is it a guarantee of a quality product release. Automation has its challenges as well and few can be listed as below:

- Unrealistic expectations are likely to exist regarding how much time will be saved in the testing schedule
- It costs time and money to analyse, acquire, learn, utilize and maintain automated test tools
- A new level of expertise may be required within the QA team for writing automation code
- Test software is like any software, it has bugs and these must be dealt with

**Test Automation Steps followed**
Before finalizing the test automation approach for any project, it undergoes the following steps:



**Test Automation Frameworks**
There are various automation frameworks available in market and we can categorize them as listed:




Each framework has its own advantages and disadvantages, based on the project requirements user can design or choose a suitable framework for the work item.

## Test Automation Tools

There are various test automation tools available in market for implementing automation. Few of the top rated tools are listed below:

1. SELENIUM
2. TestingWhiz
3. HPE Unified Functional Testing (HP UFT formerly QTP)
4. TestComplete
5. Ranorex
6. Sahi
7. Watir
8. Tosca TestSuite
9. Telerik TestStudio
10. WatiN

# Problem Statement

Inspro is an Insurance account which deals with Group insurance line of business for various clients. Multiple functionalities and products are in-scope for testing within the project; in which the Annuity product was tested for multiple data sets. The repeatability made this a good candidate for automation selection.

The test scope was to perform transactions in annuity polices. The various business transactions in annuity policies included withdrawals (secure/ non-secure/ partial/ full surrender), death processing etc. which involved processing of multiple new business transactions as pre-condition.

Total of 6 business transactions needs to be tested which involved 100 tests to be performed across 200 policies. The testing team was involved in issuing the policies, processing the batch run to turn the policies active post which the business transactions were processed in the active policies.

The entire set of activities was performed manually and the data collection for a single annuity policy consumed more effort in person hours. The effort along with the repeatable nature of work resulted in slippage as well as error in the data input resulting in rework.

**Problem statement**
The effort savings are as given in table:

| Functionality | #Tests | # Policies | Manual effort | Automation effort | Effort Savings |
|---|---|---|---|---|---|
| Annuity New Business | 100 | 200 | 13hrs | 5hrs | 8hrs |

# Proposed Solution and Technical Details

To Overcome The Challenges Of Existing Approach Such As The Schedule Slippage Due To Increased Manual Effort And Human Error Occurring Due To The Repetitive Nature Of Task, A Solution Was Proposed In Terms Of A Data Driven Test Automation Framework Using The Selenium Open Source Test Automation Tool.

The Proposed Test Automation Framework Helps In Achieving A Better-Quality Deliverable At Reduced Cost. The Solution Involved Automating The New Business For Annuity Policies On Which Further Business Transactions Needs To Be Performed As Part Of Test Scope.

The Test Automation Framework Uses The Data Driven Approach Since Testing Required The Similar Set Of Activities To Be Performed On Multiple Data Sets I.e. The Annuity New Business Transaction. By Using The Data Driven Approach We Overcome The Issue Of Having Different Test For Multiple Data Sets, The Approach Helps In Keeping The Data External To Functional Tests And Loading Them When There Is A Need To Extend The Automation Tests. Hence In The Proposed Framework The Test Script Reads The Test Data From An External Data Source Which Is An Excel File Over Here. The Web Driver On Receiving The Test Data From The Excel Sheet, Launches The Application. Once The Application Is Launched Successfully The Web Driver Executes Each Module Sequentially The Way They Appear In The Application.

The Tool And Related Details Are Summarized:

1. Tool Used And Version - Selenium Web Driver V2.53
2. Language Used - Java Se – Jre 1.8
3. Functionality Automated – New Business Functionality For Annuity Product
4. Effort Saving – 4 Minutes Of Manual Effort Over 1.52 Minutes Of Tool Execution Time
5. Framework In A Nut Shell - A Data Driven Framework Where The Business Logic Is Developed For Each User Screen In Separate Class Files And The Test Data Is Stored In Separate Excel File. Ui Objects Are Captured In Distinct Properties File Which Is Called In Separate Class File Corresponding To Each Page/ Module.

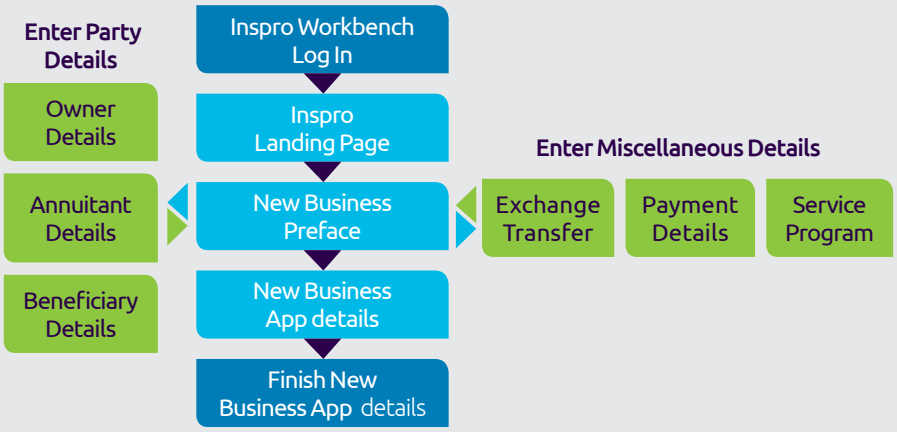
# Process Flow – Application Level

The process which is been automated with the proposed framework is the Annuity New Business transaction which involves the policy issuing for the Annuity products of Insurance. The policy issuing transaction involves multiple user entries in various screens and they are listed at a high level:

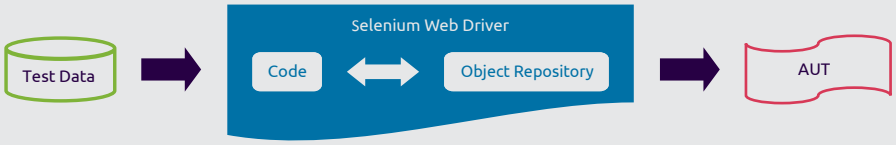
1. Product selection along with few basic details like the tax qualification type are entered at the landing page within the Inspro workbench (AUT).
2. Data entry in the New Business preface page which includes the entry of owner/ annuitant/ beneficiary details, policy effective date, application state etc. is performed.
3. Data entry in the New Business application page is performed which includes the entry of Payment details.
4. Processing the transaction at workbench with a button click which generates a policy number for reference.
5. Batch run for the database update.

The framework implemented takes care of all the processes involved in New Business except for the point # 5 which has to be triggered manually at the server side via a well-defined process.

**Process flow – Application Level**
Process flow is detailed in the diagram given:



# Architecture of The Proposed Solution




## Description of Architecture

The framework is designed using the Data Driven approach.

On a simple note, in this framework the user triggers the driver script which in-turn initialize the Web Driver. The Web Driver receives test data from the excel sheet which launches the application. Once application is launched successfully the web driver executes each module sequentially the way they appear in the application. Java files are maintained with business logic for each of the modules, a separate code maintained for handling the UI objects in another java file and the Object locators are kept in separate properties file in-order to keep the code simple and easy to house future application changes.

The Business Logic file is designed to handle the operations on the application under test UI. It collects the test data from the input excel sheet for corresponding UI module and places in an array of data. Further it shakes hand with dedicated object handling code module to get the UI objects specific to module under operation. Code is designed to synchronize the test data items and an array of UI objects received from dedicated object handling code module that seamlessly inputs test data.

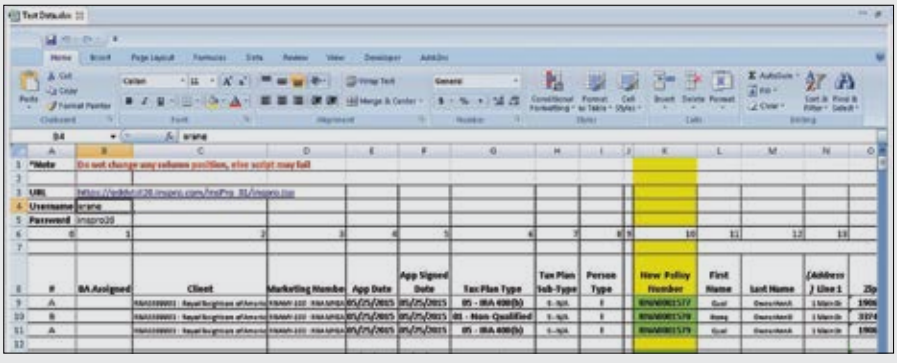
The dedicated object handling code module makes it easy for the business logic module to input the data in UI by procuring objects from Object Repository by decoding object locators.

Object Repository is a file maintained in a key-value pair format. Each key stands for UI object where value holds object locator. This makes it easy for an automation tester to accommodate future UI modifications. Object Repository is not only storehouse of all the UI objects but also helps in increasing maintainability of code.

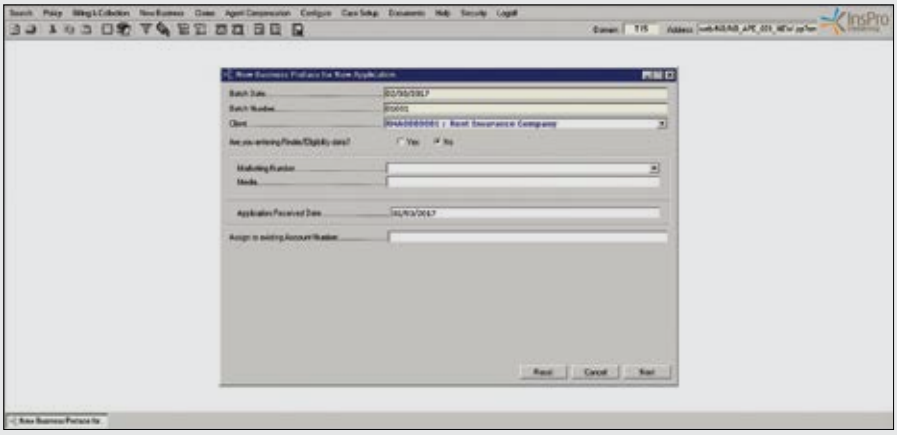
On successful policy creation the policy numbers newly generated are tracked in the test data excel sheet for further reference against each test data. This helps in a way as a reporting tool/ status tool to get the list at the end of each run.

## A detailing of the Architectural diagram

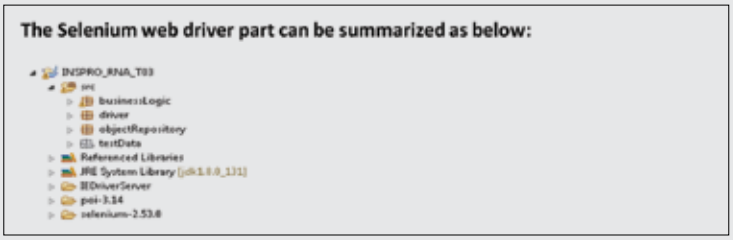
The test data input is in the form of an excel sheet and relates to the given excel:



The AUT has a landing page as below:



The Selenium web driver part can be summarized as below:




The selenium project folder structure in detail:

- src – This is a package that maintains major part of Selenium Framework. Being a root folder it cases the packages – 1. BusinessLogic, 2. Driver, 3. ObjectRepository, and 4. TestData
  - **Business Logic –** This package contains entire business logic that is required to work upon the Application UI. It contains java program that works upon the identified elements and perform the required action. Code is designed to input test data to UI elements, maintain driver session etc.
  - **Driver –** A Java program is cased in this package that binds all business logic modules together. It invokes Web Driver and thereby invoking each module sequentially. Each module is arranged in an order corresponding to application flow that are invoked by Driver Program.
  - **Object repository –** This package contains a properties file that maintains all UI object locators in a key value format. This makes the file readable and easy to understand thereby increasing its maintainability. This package also contains code that makes business logic easy to execute by reading each locator from properties file and fetching the actual object from UI and feeding as an input to Business Logic Code Modules to continue the test execution.
  - **Test Data –** Test Data provided by client is saved in an excel sheet format in a pre-defined template. This excel sheet must be placed in Test Data package. It maintains all details like Application URL, User Credentials and UI Test Data Sets
- Referenced Libraries – In order to use Selenium classes and methods, work with excel file using java classes and methods, it is mandatory to include all essential Selenium and Apache POI libraries, precompiled Java files which are nothing but JAR files. All these files needs to be imported to a project. Upon importing, they appear under this folder.

- JRE System Library – This contains JRE System Library that enables developer to code, compile and execute the framework. JRE is a Java Runtime Environment that combines Java Virtual Machine, a logical entity which provides platform to run java code and core classes including supporting libraries for easy implementation.
- IEDriverServer – It contains Selenium Web Driver executable for Internet Explorer. It is configured and invoked programmatically in src package. It enables users to code java programs that mimics manual actions programmatically to be performed on Internet Explorer Application UI.
- Poi-3.14 – it holds all JAR files that makes easy for developer to work with Excel Files. It provides pure Java libraries for reading and writing files in MS Office Format. These JARs should be imported to project before they can be used. Upon importing, they appear under Reference Libraries.
- Selenium-2.53.0 – It has all JAR files that enables user to accomplish manual actions performed on IE Window through automation by providing wider range of Java Libraries. They should be imported to project before they can be used. Once imported, they appear under Reference Libraries.

# Benefits Vs Enhancements of The Proposed Solution

## Benefits of the solution

Manual effort for the Annuity New Business / data collection process was 4 minutes per transaction.

On using this framework, the effort for the New Business process dropped to 1 minutes and 52 seconds. Thus, resulting in an effort saving which in turn resulted in a quicker TAT with respect to test data set up.

The saved effort was effectively put in for the review activities within the project which helped in ensuring a quality deliverable from a client perspective. A saving of 8 hrs per request (per request 100 tests) was achieved which is commendable saving since it is not only helped in effort reduction but also an error free test data set up.

## Enhancements to the existing framework

Listed enhancements can be made to the framework to make it more robust and increase the reusability factor across testing projects:

6. Handling the reporting functionality
7. Handling multiple products for new business creation
8. Implementation of Parallel run
9. Error handling for UI validation (it is not included as of now since this module doesn't encounter data error as data sheet is client provided and a clean data)

# About Author

**Keerthi Dilip** is a Consultant with Capgemini with the Insurance Business Unit in Capgemini Financial Services. In her 10 years of career in Insurance industry she has worked in various line of business starting from Life Insurance, Property and Causality Insurance  and Health Insurance. Her valuable contributions are in the field of manual and automation testing using Selenium open source tool. She can be reached at *Keerthi.dilip@capgemini.com.*

The author would like to thank **Akshay Rane** for his valuable contributions towards this publication.

Capgemini

## About Capgemini

With more than 190,000 people, Capgemini is present in over 40 countries and celebrates its 50th Anniversary year in 2017. A global leader in consulting, technology and outsourcing services, the Group reported 2016 global revenues of EUR 12.5 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Learn more about us at

**www.capgemini.com**

**People matter, results count.**



CSS_112017

## For more details contact:

**www.capgemini.com/insurance**
insurance@capgemini.com