Guide Test Automation

**Business needs, Business solutions**

Software Testing has found its place in the software industry, with more and more organizations understanding the crucial role that it plays in quality software production. As business requirements grow, so does the pressure on IT organizations to deliver more products with fewer resources, in reduced time and with high quality.

This scenario emphasizes the need of:

* Identification of a tool best suited for applications
* Test automation of multi-technology product suites (for ex: UI based application and SOA based web services)/li>
* Cost-effective approaches

A phased approach, keeping in mind different contributing factors, would provide a solution to this problem.

This approach would address:

* The need for a defined framework and challenges faced while creating it
* Different phases involved in framework development
* For tool recommendation: Types of tool and factors to be considered
* Factors to be considered, while creating a framework
* Components to be included in designing the framework
* How to design a prototype for input data files
* Key benefits of a defined framework

This approach does not attempt to replace any of the industry's standard test automation tools. It only suggests the process for effective test automation framework development using any tool, by overcoming tool limitations and accelerating its productivity.

**Introduction**

In today's business environment, project teams are expected to do more and deliver higher quality systems in lesser time with fewer resources. And when companies tighten their budgetary belt, software testing is often one of the first systems-development item to be done away with.

IT systems that don't solve real business problems or don't perform as promised, impose a similar economic toll on business costs and results. More than half of all software projects fail to meet objectives or suffer significant schedule and budget slippage because defects are discovered too late. All these factors combined result in a high percentage of "Defect Leakage" in the production line, resulting in poor customer satisfaction and less ROI from the product.

What needs to be done?

* Dedicated focus - Find a solution to the testing problems
* Find a long-term and cost-effective solution
* Comprehensive coverage against requirements
* Follow a "common standard" across the organization/product team/project team

All the above listed elements can be addressed by a time-to-market solution.

"Test Automation - Build a Test Automation Framework"

This paper provides details on building a Test Automation Framework using a systematic approach, which walks through a 10 different process stages to be followed in order to reap key-benefits.

**Factors to be considered - Test Automation**

Key factors required for a Test Automation to be successful include:

* Committed Management
* Budgeted Cost
* Process
* Dedicated Resources
* Realistic Expectations

**Committed Management**

**Time Plan Approval**

* Investment of time needed, for delivering the framework
* Necessary steps to be taken to make the stakeholders, management and the customer understand the importance of this one-time investment
* Scheduled time-plan approval

**Commitment on Priority**

* Commitment from the management and senior managers on the priority assigned to this activity, till the completion of framework development

**Cost and Budget**

Dedicated Budget A dedicated budget to be allocated, which includes costs related to test tool, development, deployment, resource and training. In addition, the maintenance cost for automated tests and tools must be included.

**Process**

Well-defined Testing Process

* Well-defined quality control procedures and test execution standards.
* No Ad-hoc testing
* Define the tests
* Define the test-coverage
* Define test criteria at each stage

**Resource Related**

Dedicated Resources A dedicated team is needed for effective test automation. Non-dedicated team will execute the test automation with their own limitations, which will lead to:

* Focus of activities on a specific part of a project, such as a subsystem, without concern for reuse in the future
* Less sharing of tools and information between project teams
* Automated tests that are poorly maintained, reused, and integrated due to the lack of efficient collaboration and co-ordination among different teams

**Realistic Expectations**

Management and the project team must be keep realistic expectations and should keep them in mind during the entire test automation life-cycle.

* Achieving 100% automatic tests is an unreachable goal
* All tests cannot be automated
* Benefits of automation is reaped only after several cycles of test execution
* No immediate payback for the investment
* Ramp-up time will be required for tool selection, framework creation
* Record and Playback helps minimally in test automation
* No available tool in the market supports all the systems and GUI objects
* Not all the testers can write scripts. Availability of specialized resources is a must.

Why - Framework

A framework defines the organization's way of doing things - a 'Single Standard'. Following this standard would result in the project team achieving:

**Test Library - Process Definition**

* Test Library creation follows a standard design and development process with proper documentation
* Well-defined process should be established for Team communication, Library versioning and Artifacts creation

**Standard Scripting and Team Consistency**

* Scripting standard should be maintained across the framework library creation, which includes business components, system communications, data check points, loggers, reporters etc.
* Project team should follow the defined scripting standards
* Published standards across the project team pre-empt the effort involved in duplicate coding, that is prevent individuals from following their own coding standards

**Encapsulation from Complexities**

* Test engineers are encapsulated from the complexities and critical aspects of the code
* Engineers are exposed only to the implemented libraries and tests are executed by just invoking the libraries

**Scripts and Data Separation**

* Automation test scripts separated from input data store (for example: XML, Excel files)
* No modification is required to the test scripts
* Only input data gets manipulated for testing with multiple input values

**Implement and Maximize Re-Usability**

* Establish the developed libraries across the organization/project team/product team, i.e. publish the library and provide access rights
* Utilities/components shared across the team
* Usage of available libraries
* Minimized effort for repeated regression cycles

**Extensibility and Maintenance**

* Complete support for application's new enhancements and existing features modification. For example, re-usable library could be created only for the enhancement features with a minimal effort
* Standard process for script versioning
* Role based access rights. For example access rights such as addition, modification and deletion of scripts
* Project based - Utility/Component access

Test Automation Framework Development Challenges

Test automation framework development is a multi-stage process. And passing through each stage involves multiple challenges to be addressed. Key challenges to be addressed are detailed below:

**Clear vision**

Clear vision of what needs to be achieved out of this automation must be defined and documented. To formulate the clear vision, the following needs to be identified:

* Testing Model - To be adapted (Enterprise, Product , Project)
* Types of testing under the scope based on the testing model
* Prioritizing the automation activity based on product/application/module (for example: Conducting functional testing and then performance testing)
* Identify the areas that need to be automated (for example: Registration, Order processing)
* Challenging validations that need to be taken into account (for example: Communicating from windows to Linux machine and executing the tests)
* Critical and mandatory functionalities

**Tool Identification and Recommendation**

Tool identification process is a crucial one, as it involves critical factors to be considered, which include:

* Creating a standard tool evaluation checklist which needs to be created by considering types of testing, teams involved, licensing cost of the tool, maintenance cost, training and support, tool's extensibility, tool's performance & stability etc.
* Testing requirements which may include types of testing such as Functional, Performance, and Web Service etc.
* We may need to acquire multiple tools to perform different types of testing on the product/project line.

**Framework Design - Appropriately Pick and Choose**

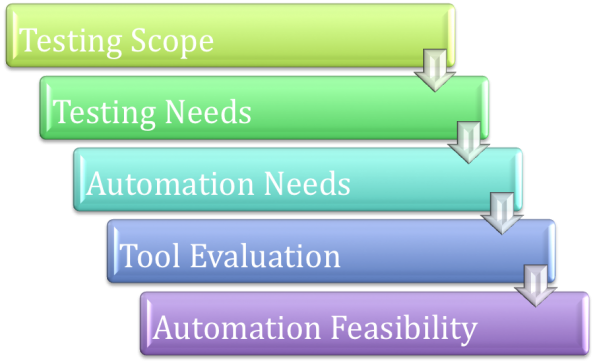
Framework design involves identifying requirements from multiple areas. At a high level, this includes (not limited to):

* Identification of necessary utility/components related to application functionalities
* Types of input data store to be communicated for data flow
* Communication between the utilities/components (for example: data check-point components communicating to the logger etc)
* Communication between the systems and utility/component development related to the same. (for example: communicating from windows to Linux environment)
* Tool extending capabilities - Developing utilities/components for the validations not supported by the identified test automation tool, if any.

Approach to Framework Development

**Pre-requisites and Assumptions**

* User is aware of the basics of test automation
* User has planned the test automation activity, by considering the scope, objectives, requirements, schedule and budget
* User has gone through the process of "Test Automation Tool Build or Buy" and has taken a decision of buying a tool or getting a open-source tool



### Identify Testing Scope

Each organization believes in its own requirements in software test automation. Considering the organization's requirements, test automation activities can be performed with three different scopes:

* Enterprise-oriented - Test automation to support different product lines and projects in the organization.
* Product-oriented - Test automation activities focused towards specific product line of applications
* Project-oriented - Test automation effort focused towards specific project and its test process.

### Identify Testing Types

Subsequent to the testing scope identification, product/application/modules under the testing scope need to be identified. Based on the product/application/module requirement, types of testing that need to be performed are identified. For example, em Scenario: For an 'Enterprise-oriented' testing scope, product A would require a functional testing, product B would require a web-service testing, a product C would require a performance testing and also complete project management etc.......

Priority must be assigned to each type of testing, based on the schedule for product release.

### Identify Requirements to be automated

Testing requirements and their nature is studied for the product/application/modules. Each requirement has its own actions, validations for testing. For example,

Scenario 1: For an application, form validation functionalities, database validation and accessibility functionalities needs to be validated.

Scenario 2:For an application, all the web-service methods needs to be validated. This would also include the delay time for the request's reponse from third party systems

All the identified requirements are assigned priority. This would help in identifying 'Build-Verification Test' (BVT) requirements that should never fail.

### Evaluate Test Automation Tool

Identified testing types and requirements, acts as a base criterion for test automation tool evaluation.

Checklist - An exhaustive evaluation checklist needs to be created which is in-line with the requirements and the tool is evaluated against this checklist for positive results. Checklist needs to cover (not limited to):

* Our requirements
* Types of testing
* Teams involved
* Licensing cost of the tool
* Maintenance cost
* Training and Support
* Tool's Extensibility
* Tool's Performance & Stability

Identify Tools - Next task would be identifying different industry-standard tools based on the requirements. This can also include tools to be purchased or open source tools. Identified tools must be evaluated against the checklist.

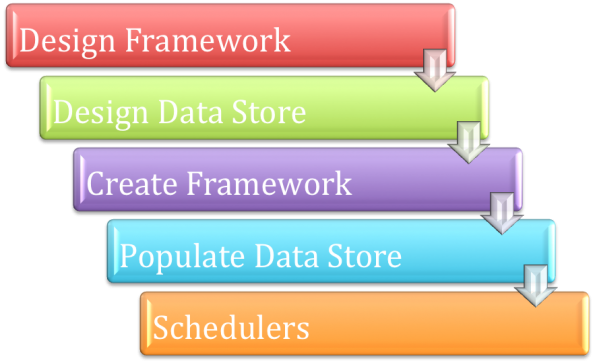
Sample Run - Tools claim that the tool supports specific requirements, but finally when we try creating our scripts it fails. Best way to evaluate is to create a sample run, this includes different types of actions and requirements we need to perform. Create sample scripts and execute the same for results.

Rate and Select Tools - Based on the sample run, supportive tools could be identified and rated. Also there may be scenarios, where multiple tools satisfy the requirements. In the said scenario, we may need to choose more than one tool, for test automation.

Implementation and Training - High rated tools will be procured/open-source licensed. Training needs to be conducted for the project team, on how to use these tools.

### Identifying Requirements can be Automated

Every tool has its own limitations. A feasibility study needs to be conducted for the requirements against the tools. This study would result in listing requirements that can be automated. Also based on the nature of the requirements, automation feasibility needs to be identified.



### Design Test Automation Framework

Generally, testers start creating test scripts based on the scenarios. This includes multiple actions to be performed against each objects. This approach leads to an ad-hoc test script creation and duplicate testing effort, i.e. testers, would create test scripts for a single action in different scenarios.

Our approach takes a different path as explained below. For designing a framework, various elements need to be taken into consideration. Utilities/Components (re-usable) would be designed for the following elements that include (not limited to):

* Actions to be performed - Identification of actions to be automated for each object of the application
* Communicating Systems - Study of different internal systems, third-party systems and their communication methodology
* Business Rules - List of business layers and any specific algorithm has to be studied. A separate function needs to be created for each specific algorithm.
* Database Communication - Database validation and check point validations
* Communication with additional automation tools - In the scenario, where we would require communicating with different automation tool. All the communication requirements needs to be identified and designed
* Data retrieval - Retrieval of data from multiple input data stores
* Schedulers - Functionalities related to invoking of relevant scripts based on scheduler configuration
* Tool Extensibility - Overcoming tool limitations. Components for actions/validations for which the tool does not provide any support
* Device Communication - Device communication and data transfer related actions/validations
* Log - User-defined logs for analysis
* Error Handlers - Error handlers to handle known and unknown errors and log the information
* Custom Messages - Display of relevant defined messages
* Result Presentation - Customized and presentable reports on completion of test execution

Test automation framework would be designed based on the listed factors, using the following guidelines.

* Application-independent.
* Easy to expand, maintain, and perpetuate.
* Encapsulate the testers from the complexities of the test framework
* Identify and abstract common functions used across multiple test scripts
* Decouple complex business function testing from navigation, limit-testing, and other simple verification and validation activities.
* Decouple test data from the test scripts
* Structure scripts with minimal dependencies - Ensuring scripts executing unattended even on failures

### Design Data Input Store

Types of input data files supported by the tools, needs to be identified. Based on the requirements, input files can be categorized as (not limited to),

* Objects Identifier - Object identification syntax respective to the tool, mapped to the logical object name. For example,"A username textbox in the registration page mapped to logical object name - regUName"
* Scenarios/Workflows/Transactions based input - Complete set of input data for different scenarios/workflows/transaction. Each scenario/workflow/transaction translates to "n" number of test cases. This test case based user input benefits the team during future enhancements, in a way that multiple input data can be added using the Test Case ID. For example, "A complete financial transaction order processing, which invokes web services methods for order processing. In this case, input data is created based on test case id". TestCase (TC) 1 would be entering account details, TC2 Order details etc ...
* Custom Message - This can contain custom messages to be displayed for known and unknown errors.
* Driver - File can contain list of file/transaction/workflow id's to be referred to, for a selected batch execution/group of test case executions

For all the files types, file format needs to be identified and prototyped based on the input data storage.

### Develop framework

Framework development is facilitated using the same set of identified tools. Scripting language supported by the test automation tool is used to create the components. Tool extensibility utility/component can be developed using a different language. Utility functions/components created based on framework design is explained in Step 7.

In addition to the re-usable components driver scripts and worker scripts needs to be created.

* Driver Scripts - Scripts that execute a set of transactions, by invoking relevant re-usable utilities/components for each test case. Driver scripts can be mapped to a group of test cases related to a scenario/transaction/screen/window.
* Worker Scripts - Scripts that execute the driver scripts. Worker scripts are group of driver/individual scripts to execute in a batch mode. Worker scripts produce the final results, for the executed batch.

Approaches for developing re-usable utilities/components include:

* Record/Replay - Used to identify the object recognition pattern, of the specific tool. Very minimal usage
* Screen/Window/Transaction - Used to identify different scenarios to execute scripts in batches.
* Action/Keyword - Invokes the relevant utility/components to perform actions on specific objects. Driver scripts are created based on this input
* Data Driven - Input to keyword driven approach, where validation needs to be performed using multiple input combinations.

### Populate Input Data Store

Input data store needs to be populated based on the file structure defined in Step 6. Data can be populated either manually or in an automated fashion from different data-sources. Test data would be populated based on parent-child hierarchy. For example,

"A transaction would be a parent hierarchy and Input to the test cases would be the child"

### Configure Schedulers

Scheduler requirement needs to be identified. Schedulers can be configured to run a worker script (batch script) on a specific time period. This approach benefits in a way that even a business user can configure the scheduler and make the test execution happen.

## Key Benefits of Framework

### Standard process in Production

- Test automation processes, a single standard is established across the organization. This helps the organization as they follow the standard processes as compared to pre-empted ad-hoc processes, which yield no results.

### Free from dependencies

- Complete coding and component usage standards are defined in production. Organization benefits include:

* Independency from the individual coding standards and the utilities/components created
* Complete documentation helps the organization in inducting the new members with minimal effort

### Complete Coverage

- Requirements are collected from an overall organization's perspective (for eg: Product suite on multiple technologies .net and java etc). Overall coverage of re-usable components which includes (data communications, system communications, schedulers, loggers, reporters etc)

This overall coverage minimizes the testing effort during the later stages of the releases, for the entire product suite across the organization.

### Future Enhancements Support

- Organizations need not worry about testing future enhancements. Only the validations related to the enhancements need to be added to the existing base framework, and that too with minimal effort.

### Cost Estimation

- At the end of Step 6, the complete cost for the framework development can be estimated. This cost includes,

* Acquisition cost - In the procurement process of the tool, following cost needs to be considered:
  + Tool Cost
  + Cost based on number of licenses, based on our requirements
  + Tool Support Cost (On-line, Telephone)
  + Version Upgrade Cost
* Training - Training cost incurred for training test engineers, business users, developers and creating supportive training documentations must be taken into consideration
* Environment - Cost involved in setting up the system environment (Hardware and Software) must be taken into consideration
* Development - Development cost can be calculated based on the components designed in the framework development - Step 6
* Maintenance - Each tool has its own maintenance requirements. Some tools may demand for part-time, some tools demand for a dedicated resource maintaining the tool. Based on the tool's requirements, this cost needs to be calculated

## Conclusion

Having seen the benefits, the importance of testing and test automation will be increasingly realized across domains and to accelerate the process. "Test Automation Framework" would find greater acceptance and importance in the industry.