Test Automation for Higher Testing Productivity

Presenter : Nadeem S A
Designation : Test Manager
Email : nadeem@relq.com

RelQ Software Pvt. Ltd
Agenda

- Introduction
- Constraints in Manual Testing approach
- What is Test automation?
- Implementing Test automation for Higher productivity
  - Feasibility Analysis
  - Tools and their selection criteria
  - Test script development methodologies
  - Test Automation frameworks
- Case study
- Conclusion
- Q & A
Introduction
Software Testing

- Necessary to uncover hidden defects and improve the quality and reliability of software.
- Inadequate testing will be potentially disastrous to the business.
Software Testing

- Is a specialized task
  - To test applications of different domains
    - Require resources with Domain experience
  - To test systems built using heterogeneous environments
    - Require resources with different technology skills
  - To perform different Test types
    - Require resources with different Testing skills

- Consumes Time and resources to plan and execute tests
Challenges In Testing
Challenges

- How to perform the different types of Tests to get a quality product?
  - Consistently without cutting down the amount of Tests
  - Without impacting cost in a major way
  - Without impacting the release deadlines
Productivity Constraints of Manual Test Approach
Manual Test Approach - Constraints

- Time consuming and depends on the productivity of the Tester

- As functionality grows per release, the effort and the schedule required to test the application increases

- If functionality increases and the Test schedule and resources remain constant, then the Test coverage gets impacted.
Manual Test Approach - Constraints

- Difficult to execute Performance tests that require large loads using Manual Test approach

- Manual testing is taxing on the testers who repeat the same tests for an extended period of time
Why consider Test Automation?

- To consistently achieve good test coverage across all releases despite functionality increase
- To deliver more in less time without cutting down the tests
What is Test Automation?

The use of software to

- Set up preconditions for a test
- Initiate and control the execution of tests
- Compare actual and expected results
- Perform the required reporting functions
Implementing Automation for Higher Test Productivity
Test Automation Process

1. System study and confirm technical requirement
2. Tool evaluation and selection
3. Application navigation using the tool
4. Selection of test automation methodology

5. Application support /development
6. Application updates

7. Review results and report the defects
8. Perform tests/regression testing
9. Review and approve scripts
10. Build test scripts and test drivers
11. Enhancement of test scripts

Test summary report
Importance of Feasibility Analysis
Feasibility Analysis

- Automation feasibility is used to evaluate
  - Compatibility of Automation tool with application
    - Are all GUI objects getting identified?
    - Do I need to have additional plug-ins?
  - The need to use additional technologies
    - Need Perl or other scripting languages?
    - Need to store Test data in database?
  - The feasibility of automating the required test cases.
Feasibility Study

- Outcome of Automation feasibility study
  - How much complexity exists in implementing Automation for current and future needs?
  - What improvement in Test coverage does Automation provide?
  - How much Test effort reduction does Automation enable?
  - Whether the right tool has been selected for Automation.
Selecting the Automation Tool
Selection criteria for Automation tools

- **TYPE OF TEST**
  - Functional Testing
    - Silk test, Winrunner, QTP, Rational Robot, Test Complete, Test partner
  - Performance Testing
    - Load Runner, Silk Performer, QA Load
  - Unit Test
    - JUnit, NUnit
  - Test Management
    - Test Director, Silk Central, Quality Center, QADirector

- **USABILITY**
  - Ease of use of tool features
Selection criteria for Automation tools

- **COMPATIBILITY**
  - With the application for current and future needs
  - With required OS (Windows XP / NT / 2000 / Unix / Linux)
  - With required Browsers (IE, Netscape etc)

- **LEARNABILITY**
  - Adequate Training material and technical support
  - Online Community

- **CAPABILITY**
  - Support for Java / .NET / Oracle Tech.
  - Support for Client Server / Legacy Systems
Automation - Specialization in Testing
Setup best practices

- Capture the requirements by involving end users of the suite
- Design inline with the requirements
- Select teams with programming skills.
- Code as per identified standards / guidelines
- Develop re-usable scripts
- Maintain version control
Set up Best Practices

- Inadequate Testing of Automated scripts
  - Causes scripts to break easily

- Create repository of tested and re-usable scripts
  - Reduces time to develop automated scripts to test new functionality
Plan for maintenance and use

- Provide user manuals that contain information about
  - The features of the Automation suite
  - How to use the features of the suite
  - How to perform the Test and interpret results

- As functionality grows, scripts may have to be added or enhanced.
  - Train the end-users on how to maintain the suite
Test script development
Test script development Methods

- Record and playback
- Data driven
- Keyword driven
- Hybrid
User activates the “record” feature of the tool.

User performs the required actions for the test on version 1 of the application.

The automation tool generates scripts by recording user actions.

The generated scripts can be played back on version 2 to reproduce the exact user actions.
Record and Playback approach

Advantages:

- Less effort for automation and quick Returns
- Does not require expertise on the tool

Limitations:

- High dependency on the GUI of AUT
- Scripts contain hard coded data
- Not a recommended approach for developing scripts.
Data Driven approach

- Test scripts are freed from hard coded data.
- Test data is got from external files.
- Using the scripting language features of the tool, create a Library of functions to
  a) Perform the application specific tasks
  b) Read the test data files
  c) Log test results
- Group the functions to develop test cases
- Invoke the scripts using Test drivers
**Additional Info:**

- **Config files**: Allow the user to
  a) Select the test cases
  b) Select the type of test

- **Object Repository**: Can be an external file storing the required properties

- **Data Files**: Store the required data

- **Result files**: Store results in customized format
Advantages:

- Data is separated from scripts and stored in data files
- The volume of test data and its combination can be increased for exhaustive testing

Limitations

- Application must be available to start scripting
- Maintenance of the test script due to GUI changes is high
Keyword driven approach

- Identify the Keywords and related data for each task

- Create a Library of functions to the task specific to each keyword

- Group the keywords to create required Tests in Action files

- Using Test drivers, read the Action files to execute Test cases
### Keyword driven approach - Sample Action file

<table>
<thead>
<tr>
<th>Auto_ID</th>
<th>Command</th>
<th>Object</th>
<th>Data</th>
<th>TC_Execute?</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUT0 - 001</td>
<td>TCStart</td>
<td></td>
<td></td>
<td>YES</td>
</tr>
<tr>
<td>AUT0 - 001</td>
<td>AssignValue</td>
<td>%URL%:=<a href="https://www.company.com">https://www.company.com</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT0 - 001</td>
<td>LaunchApp</td>
<td>%URL%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT0 - 001</td>
<td>TCEnd</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT0 - 002</td>
<td>TCStart</td>
<td></td>
<td></td>
<td>YES</td>
</tr>
<tr>
<td>AUT0 - 002</td>
<td>AssignValue</td>
<td>%loginUserID%:=UserID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT0 - 002</td>
<td>AssignValue</td>
<td>%Password%:=Password</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT0 - 002</td>
<td>EnterData</td>
<td>loginUserID</td>
<td>%loginUserID%</td>
<td></td>
</tr>
<tr>
<td>AUT0 - 002</td>
<td>EnterData</td>
<td>loginPassword</td>
<td>%Password%</td>
<td></td>
</tr>
<tr>
<td>AUT0 - 002</td>
<td>ClickButton</td>
<td>Submit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT0 - 002</td>
<td>VerifyResult</td>
<td>LoginSuccessfulMsg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT0 - 002</td>
<td>TCEnd</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Additional Info:

- **Action files**: Contain test cases formed using keywords.
- **Config files**: Allow the user to select any Action files for execution.
- **Object Repository**: External file storing the required properties.
- **Result files**: Store results in customized format.
Keyword driven approach

Advantages:

• The User can easily create the test scenarios by grouping a set of Keywords in the Action files.

• The action files can also be a single point of reference for Manual and Automated Test cases.

• Maintenance is easy as changes in functionality require changes in Action Files only

Limitations:

• The number of action files increases with functionality.

• The initial effort to design is more compared to data driven approach

• Users will have to be trained on the use of Keywords
Hybrid

Scripts developed using a combination of Data Driven and Keyword Driven approach.

Advantages:

- Use the best features of different methodologies to suit the project requirements.

Limitations:

- Maintenance is an issue.
Test Automation Framework
What is a Framework?

- Framework is a logical support structure in which another software project can be grouped and tested.

- Framework comprises support programs, processes, code libraries etc to help develop and glue together the different components in a project.

- Can be implemented by leveraging proprietary automation frameworks and tool sets from leading vendors, supported by additional programming and processes to maximize the benefits of test automation.
Objectives of Automation Framework
Objectives of Frameworks

- To gain long term benefits by ensuring
  - Reusability of Automated scripts across projects
  - Scalable environment for tests
  - Effort for test environment setup is reduced
TYPES OF AUTOMATION FRAMEWORKS
Types of Frameworks

- Process framework
- Test Framework
- Hybrid framework
Process Framework

- Used to automate test environment setup process for multiple tests having similar Test environment.

End User

<table>
<thead>
<tr>
<th>Command for Test 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command for Test 2</td>
</tr>
<tr>
<td>Command for Test n</td>
</tr>
</tbody>
</table>

Automated Test Environment Setup

<table>
<thead>
<tr>
<th>Automated Test 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated Test 2</td>
</tr>
<tr>
<td>Automated Test n</td>
</tr>
</tbody>
</table>
Process Framework - Advantages

- Saves effort in test setup process
- Scalable for Multiple types of tests
- Provides an easy interface for end user to execute tests
Focus on creating reusable components that can be used for similar tests across projects.
Test Framework - Advantages

- Re-usability of components across projects reduces redundancy
- Re-usability of components also reduces the overall time to develop the Test suite
Hybrid Framework

Framework that combines the features of process and test frameworks
Case Study 1

A US based client, wanted to use Automation to resolve the following issues.

- Improve the Functionality Test coverage for Localized (BEFIGS) builds of the Desktop application
- Reduce the effort / time to perform the different tests

Additionally, a Test Automation framework that would enable end to end testing for localized Desktop application had to be developed.
Case Study 1

Develop test automation suites using Silk test for the following.

- Capturing UI differences in Localized (BEFIGS) builds
- Test installer functionality for localized (BEFIGS) builds
- Capturing Registry and File attribute differences against benchmark data after installation
- Testing functionality of all modules of localized (BEFIGS) builds
Case Study 1

- Test requirements
  - No of Products = 2
  - No of Test cases = 1400 test cases each for US and Localized builds (BEFIGS Languages) = 1400 x 7
  - Average No of builds per product for US = 10, BEFIGS = 10

- Challenge
  - Manual Execution of all test cases on all environment - Impossible.

- Solution
  - Framework based Automation
Case Study 1

- Automation Framework requirements
  - Should provide a common Command format for the user to execute any of the Installer or Functionality or Registry-file compare or UI Compare suite
  - Should allow for testing the suites on any of Win98 or Win 2k or Win XP OS
  - Should allow for executing any suite on any build on any OS for any of the BEFIGS languages
  - Should automate the process of test setup environment and execution
Case Study 1

Format of Command to execute test =
Go [Test] [Build] [OS] [Language]

<table>
<thead>
<tr>
<th>SL NO</th>
<th>COMMAND PARAMETERS</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Go</td>
<td>Command to initiate the execution of the test</td>
</tr>
</tbody>
</table>
| 2     | Test               | The first argument represents the following tests  
a) Installer b) Functionality c) UI capture d) Reg-file compare |
| 3     | Build              | The second argument represents the build that needs to be tested |
| 4     | OS                 | The third argument represents the operating system  
(Win2k, Win 98 and Win XP) on which the build to be tested |
| 6     | Language           | The fourth argument represents the language of the localized build that needs to be tested.  
a) Italian b) French c) Spanish d) German e) Brazilian Portuguese |
Case Study 1

Process framework implementation

Accept Command from End User for
- Type of test
- Build to test
- OS to test
- Language to test

Automatically select a free system for test

Automatically load OS image on VM ware

Automatically load the build for test

Automatically load and execute Automation suite for test

Log Results
Case Study 1

Components of the framework

- Automation suites (Developed using Silk test 6.5)
- Perl scripts
- DOS Batch files
- OS images
- VM Ware
Case study 1

Effort to develop framework = 12 Man months
- 3 resources
- 4 months

Effort to develop automation suites = 42 Man months
- 7 resources
- 6 months
Case Study 1 - Architecture

Test Environment
- Test Client Machine cluster
- System/application under test

Test Controller
- Test Controller Script
  - Check out scripts
  - Compile/build
  - Deployment scripts
  - Test Setup scripts
  - Test data generation-scripts
  - Test execution scripts
  - Report generation script
  - Report publishing script
  - Report mailing script

Test Repository
1. Test Scripts
2. User Defined Library
3. GUI files
4. Utility Scripts
5. Test data files
6. Tool Installer

Test Script Selection File/SCM

Web Server

Test database (optional)

System/application under test

1. Test Scripts
2. User Defined Library
3. GUI files
4. Utility Scripts
5. Test data files
6. Tool Installer
## Reduction in Effort to execute Test cases using Automation suite

<table>
<thead>
<tr>
<th>Task</th>
<th>Manual Effort (Man days)</th>
<th>Automated Effort (Man Days)</th>
<th>Saving in Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional Test</td>
<td>14</td>
<td>4</td>
<td>71 %</td>
</tr>
</tbody>
</table>

### Test Effort reduction

![Bar chart showing reduction in test effort from manual to automated testing]

**Type of Testing**
- Manual
- Automated

**Effort in days**
- 0
- 5
- 10
- 15
### Improvement in Test coverage using Automation

<table>
<thead>
<tr>
<th>Task</th>
<th>Manual approach</th>
<th>Automated approach</th>
<th>% Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional Test Coverage</td>
<td>20 %</td>
<td>80 %</td>
<td>60 %</td>
</tr>
</tbody>
</table>

**Case Study 1**

![Test coverage improvement graph](image)
Case Study 1

Reduction in Effort to Setup Test environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Manual Effort (Minutes)</th>
<th>Automated Effort (Minutes)</th>
<th>Saving in Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Environment Setup</td>
<td>120</td>
<td>20</td>
<td>83 %</td>
</tr>
</tbody>
</table>

Test environment setup effort reduction

![Chart showing reduction in effort between manual and automated testing]
Case Study 2

- **The Client**
  - A leading Bank having operations across the globe

- **The System**
  - Online Cash Management System, with support for multiple branches

- **Issue faced by client**
  - Functionality increases per release, need to ensure adequate test coverage in the fixed Test schedule (4 weeks)
  - Need to reduce the Test effort
Case Study 2

- **Tool**
  - QTP 8.2

- **Methodology**
  - Keyword-Driven

- **No. of Test Cases**
  - 3,500 (x 5 branches)

- **Scripting languages**
  - Perl, VB Script
Case Study 2

- Automation Suite requirements
  - Should be possible to execute the test cases for any of modules for any of the five branches
  - It should be possible to configure and selectively execute any/all testcase(s) in any of the Action files
  - Summary and detailed results of Test case execution should be available for all Test case executions
  - Should be portable on Win2k / XP English and Japanese OS.
  - Results should be emailed to selected recipients
Case study 2

- Effort to develop automation suite = 16 Man months
  - 4 resources
  - 4 months
Case Study 2 - Architecture

Diagram:
- SCRIPTS
  - Driver Script
- CONFIG
  - Configuration Files
- LIBRARY
  - Utility Functions
  - Business Functions
- OBJECT PROPERTIES
  - Object Repository
- ACTION FILES
  - Automated Test Cases
    - TC 1
    - TC n
- RESULTS
  - TC 1 Res
  - TC n Res
- KEYWORD TABLES
# Case Study 2

## Reduction in Effort to Execute Test Cases using Automation suite

<table>
<thead>
<tr>
<th>Task</th>
<th>Manual Effort</th>
<th>Automated suite Effort</th>
<th>Saving in Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executing 10,000 Test cases</td>
<td>117 Man days</td>
<td>20 Man days</td>
<td>83 %</td>
</tr>
</tbody>
</table>

### Graph: Test Effort reduction

- **Manual**
- **Automation**

---

*Effort in Person Days*

*Type of Testing*
Case Study 2

Improvement in Test Cases execution coverage

<table>
<thead>
<tr>
<th>Task</th>
<th>Initial Coverage</th>
<th>Current Coverage</th>
<th>Improvement in coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executing test cases</td>
<td>50 %</td>
<td>90 %</td>
<td>40 %</td>
</tr>
</tbody>
</table>

Improvement in Coverage

- Manual
- Automation

Type of Testing
Conclusion

- Test Automation does not eliminate Manual testing but supplements it.

- Test Automation requires skilled resources to be implemented properly.

- A framework based approach for test automation supports maintainability, extensibility, scalability, ..

- Test Automation Framework significantly improves the productivity of testing.

- Very useful in tests that are to be repeated in different platforms.
Conclusion

- Using Automation, significant improvement in coverage of tests during regression cycles is achieved.

- Integrated with a test management solution, provides complete solution for automated testing, reporting, test coverage and defect logging.

- Testing team knowledgeable of the application and test cases is preferred for test automation.
Thank You