

White Paper



Test Automation for Effective Post Deployment Testing

Nailesh Sampat, Anupama Jamwal

Abstract

Testing is critical to ensure that the existing functionality of software applications is not impaired by modifications or upgrades. As independent testing is still evolving, post-deployment testing is dynamic and challenging and requires a creative approach to be successful. The key area of concern in post-deployment testing is 'time-to-market', as frequent releases demand quick test turnaround.

This paper describes how [Test Automation](#) can address many of the challenges in post-deployment testing. The focused approach to automation defined here will enable testing organizations to cope with the release plans of applications

The Scenario

Organizations often use COTS (commercial off-the-shelf) products to implement business processes and they are often implemented 'As-Is' – as part of Release 0 (zero). Changes or enhancements are planned only in subsequent releases. However, users generally have specific requirements regarding the usability of applications, which may require enhancements or updates to the application. The release of a patch to enhance the application must be meticulously tested to ensure that it does not impact existing functionality. There can be two types of updates after an application goes live, namely planned updates and hot fixes. Patches released might cause additional defects, which need immediate fixes called hot fixes.

Only a test team that can churn out test deliverables in the shortest time can resolve post-release issues quickly. Post-deployment testing involves re-execution of previous tests as well as upgrading core functionality regression packs with new test cases, thus constantly expanding the scope of testing. The expansion of scope directly impacts testing timelines. The solution does not lie in increasing team strength. The test team must move toward automation of frequently used core test cases.

Problems faced while implementing COTS products:

1. Since the initial release is tested and delivered by a vendor, there may be a lack of application documentation (design, test artifacts, etc.) and test packs
2. Extent of testing is not known and must be inferred based on the stability of the application
3. Although standalone releases are usually stable, integration with other products often creates challenges
4. Enhancements to COTS products can raise design and scalability issues
5. Changes to any functionality may cause problems in other areas

Challenges faced in post-deployment testing:

1. Issues faced in production environments must be resolved quickly.
2. The scope of testing grows with the release of frequent patches, leaving very short timelines for testing. Manual testing increases the chances of human errors.
3. As the requirements may be around a new functionality or a defect fix, the focus of change may not be restricted to the extent of the change; rather it may extend to the end-to-end business flow.
4. The post-deployment test team, also called the live-support test team, gets little time to understand the application and analyze its requirements.

Test Automation

When to Automate

As applications grow and evolve, the size of regression test packs increases. The challenges in ensuring regression testing or executing test scenarios in the shortest possible time and the repetitive nature of the task make a case for Test Automation. However, there are many challenges in Test Automation. It is important to make sure that automation will bring the expected benefits. To start with, a Proof-Of-Concept (POC) must be developed on select applications to get a buy-in from all the stakeholders.

It is essential to understand the application life span plan and obtain a high-level overview of upcoming changes. It is imperative to determine the ratio between the number of test cases for existing functionality and those for new functionalities, for every test cycle. If a majority of the test cases are existing functionality test cases, it makes business sense to automate them for faster turnaround time, and to reduce regression costs and human error. This also helps in obtaining approvals from business sponsors of automation.

Test cases for new functionality or the test cases related to a hot fix should be executed manually for the first time, and may be subsequently added to the automation pack based on feasibility. Test Automation is an effective solution when the life of the application released is long or when the test cases need to be executed frequently.

When Not to Automate

It is better to run the test cases manually when the life of the solution is short, very few releases are planned, or when the test cases are not repetitive in nature.. This is because test automation comes at a cost — tool cost, cost of skilled resources and, most important, the time required to automate. If the costs cannot be recovered in the period in which the automated scripts are run, it is best not to automate.

Focused Approach to Test Automation

Infosys uses a well planned and focused approach to test automation. The feasibility of automation is studied by developing a Proof-Of-Concept (POC) on major applications and a common framework is designed to support multiple applications.

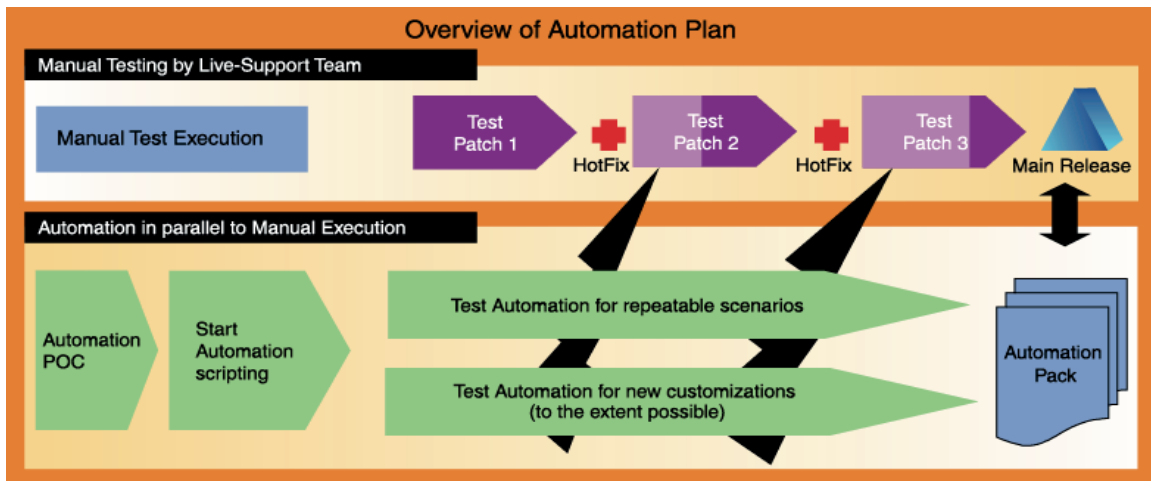


Figure 1: Automation Plan

Figure 1 shows a plan proposed and executed by Infosys that shows automation activities in parallel with manual testing. This is done to ensure that there were no delays for releases tested manually. It may be noted that the automation of new customizations and revision of the automation test pack is an iterative process.

Figure 2 depicts the five pillars of our framework.

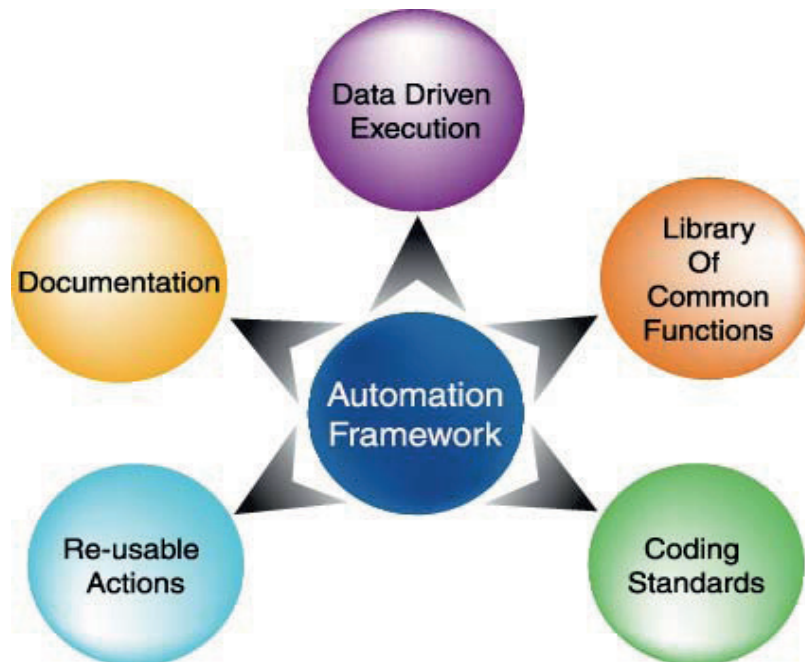


Figure 2: Pillars of the Automation Framework

Figure 3 illustrates the Automation Framework

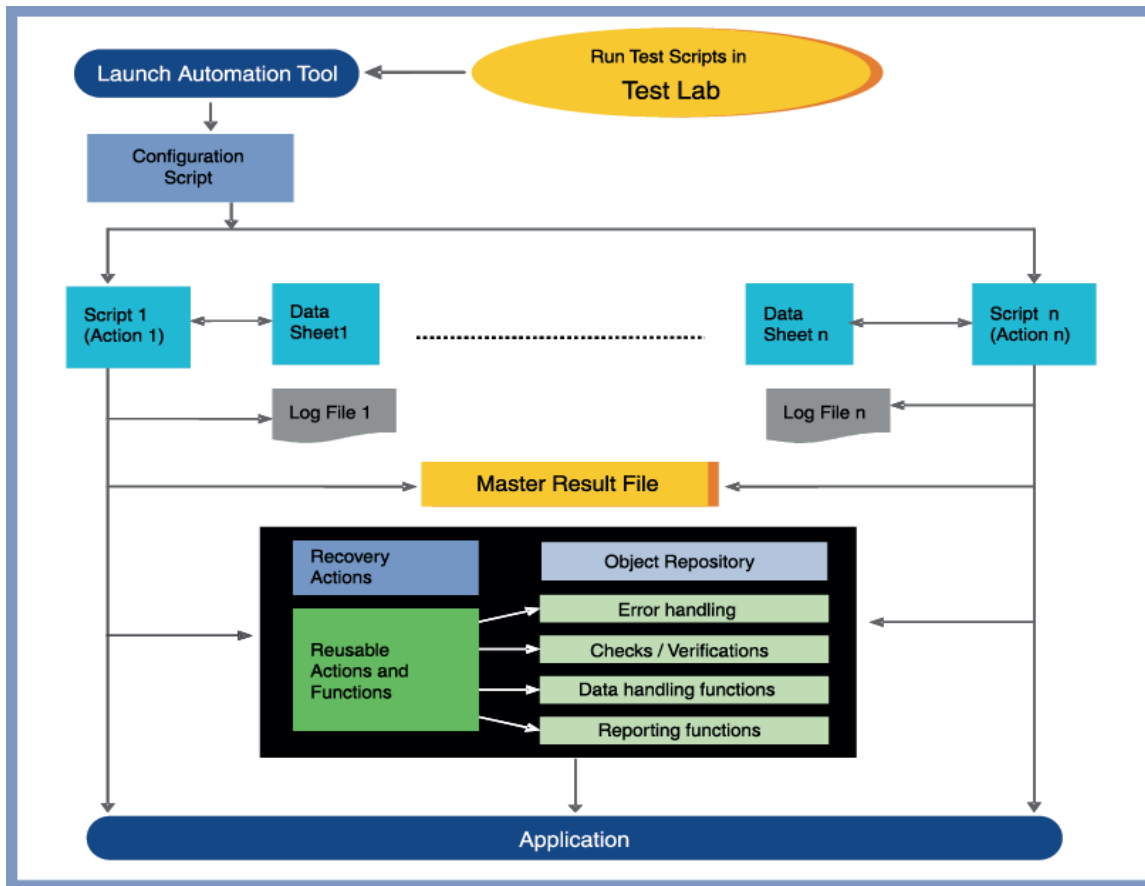


Figure 3: Automation Framework

Features of the Infosys Automation Framework

- **Data-Driven Execution**

The data-driven approach is applied to test automation as it separates data from the automation scripts by moving data to an input file called the data set. By parameterization of scripts, scripts can be run on multiple data sets. Also, this way, changes in data require updates to the data set only and not the scripts themselves.

The post-deployment team ensures that the data created can be used to test ‘end-to-end’ business scenarios.

- **Library of Common Functions**

A library of common functions is built. These functions can be used across different applications for test automation. Thus, rework is avoided and reusability is ensured.

- **Coding Standards and Naming Convention**

Coding standards are defined and documented to make the scripts easily readable and uniform. A naming convention is applied to Variables, Scripts, Functions, Log Files, Data Sheets, Object Repository, Reusable Actions, Script Actions, etc. Standards are set for commenting.

- **Reusable Actions**

All the test cases are analyzed before automation of individual test cases. Many actions are often found to be common and thus reusable. These reusable actions are automated and placed in a folder named “Reusable Actions” under the automation folder hierarchy and are accessed and used by multiple scripts. Examples of reusable actions include Login, Import, Registration, Logoff, etc.

Scripts are designed to achieve ‘end-to-end’ testing of business scenarios to the maximum possible extent.

- **Documentation**
Detailed documentation (Automation Help Manual) is provided, incorporating every aspect of automation including coverage of automation, library functions, etc. The documentation also gives technical details at the script level, covering initial setup, test execution flow, pre-requisites and test data requirements. This facilitates maintenance and enhancements of test scripts for the post-deployment test team. The document also helps other teams that execute the regression pack in their test environments.

Critical Success factors

- **Efficient Planning**
Automation coverage must incorporate script calls to the test management tool for result reporting, by providing standard automation reports at the script level.

The post-deployment team must estimate the impact of changes caused by a new patch when automation is in progress. Understanding the application release plan during the automation phase helps reduce maintenance work.
- **Application Setup**
Dedicated access to the application and database is required to ensure that no other team is impacted. If this is not possible, records created during automation must have unique naming conventions.
- **Automation Tool Setup**
It is important to understand the license type and the local desktop setup before starting an automation project.
- **Scripting/Coding Standards**
The naming convention must be in accordance with the test management tool's naming convention.
- **Issue Clarification**
It is important to understand the application architecture. Most of the issues in post-deployment can be overcome by self-reading or analyzing the functionality.

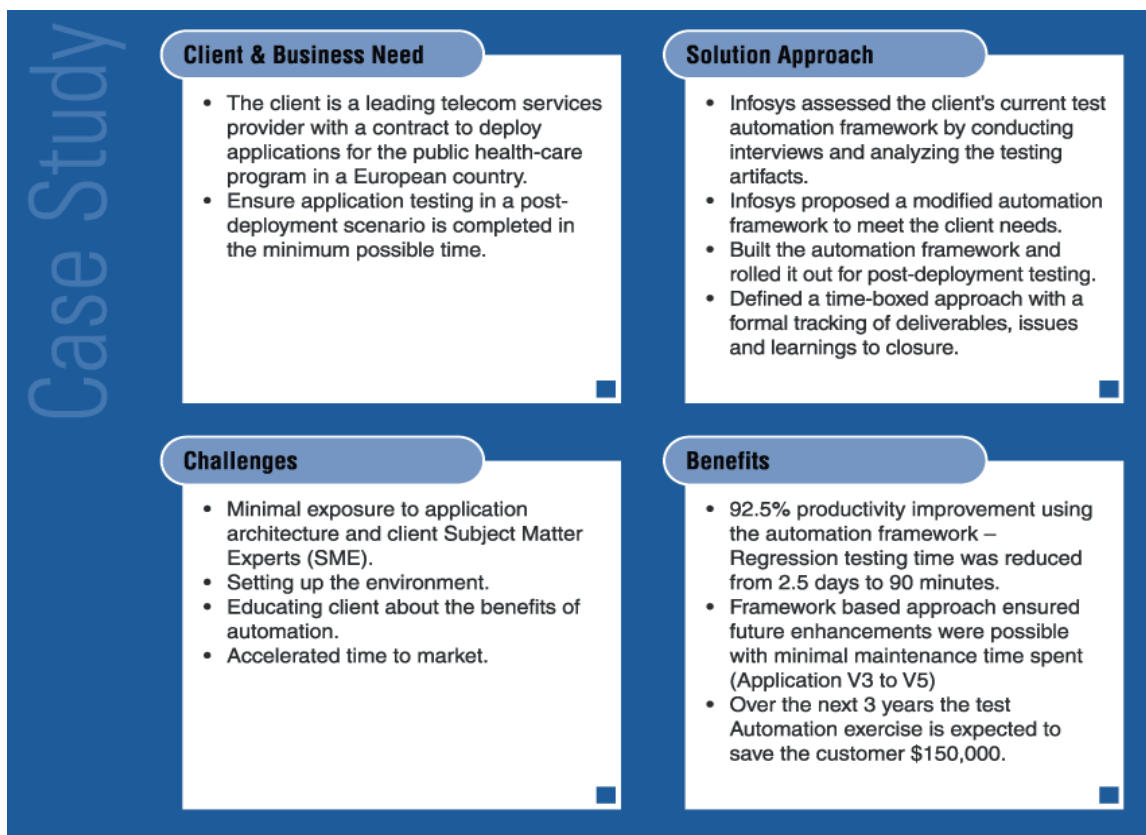


Figure 4: Project Case Study

Tangible benefits

Apart from planned and emergency patches, there often are other major application releases. Infosys' Automation Framework ensures that most of the test scripts can be "reused" with little tuning. However, at times a few scripts may have to be modified and new ones may have to be added to the core functionality regression pack.

Some of the benefits of this solution are:

1. Productivity can be improved by using the automation pack. Regression testing time is considerably reduced.
2. Test scripts are shared with the deployment team during application build to help them test the application in the production environment.
3. Test scripts are shared with the development or vendor team to ensure that the application meets quality expectations.

Conclusion

Test automation, when carried out in a planned manner, offers great benefits and is therefore worth considering. However, since every project is unique, the effort, time and cost to be invested in automation must be assessed to ensure adequate returns.

About the Authors

Nailesh Sampat is a Senior Test Manager with Infosys. He has over 12 years of professional experience in Information Technology, with 6 years of leadership experience in testing projects.

Anupama Jamwal is a Test Analyst with Infosys. She has about 8 years of experience in the IT industry. She has effectively executed testing projects in Healthcare, Banking and Finance domains, and has worked on manual as well as automation testing.



For more information, contact askus@infosys.com

About Infosys

Many of the world's most successful organizations rely on Infosys to deliver measurable business value. Infosys provides business consulting, technology, engineering and outsourcing services to help clients in over 30 countries build tomorrow's enterprise.

For more information about Infosys (NASDAQ:INFY), visit www.infosys.com.