

STeP-IN SUMMIT 2007

International Conference On Software Testing

Test Automation in Post-Live Scenario

By

Nailesh Sampat and Anupama Jamwal
Infosys Technologies Limited

nailesh_sampat@Infosys.com and anupama_jamwal@Infosys.com

Copyright: STeP-IN Forum and Quality Solutions for Information Technology Pvt. Ltd.

Published with permission for restricted use in STeP-IN SUMMIT 2007 in agreement with full copyrights from owner(s) / author(s) of material. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior consent of the owner(s) / author(s). This edition is manufactured in India and is authorized for distribution only during STeP-IN SUMMIT 2007 as per the applicable conditions.

Practices Experience Knowledge Automation

Produced By

STeP-IN
Forum

www.stepinforum.org

Hosted By



www.qsitglobal.com

Abstract

Testing of an application does not end after go-live incase it has not been tested using structured testing methodology or there are further enhancements in the application. On the contrary it becomes all-the-more critical and challenging as the application evolves and undergoes changes after going live. The changes arise mainly due to enhancements, change requests or resolution of test issues that had managed to slip through earlier stages of testing. Independent Post-Live testing is introduced to manage quick time to market for releases that are in the form of hot-fixes or patches. Independent testing in this area is still evolving and is very dynamic and challenging and requires an equally dynamic and creative approach for its success.

Post-Live testing involves not only re-execution of previous tests, but also upgrading of core functionality regression packs with new test cases. Thus the scope of testing increases which directly impacts the testing timelines, as the luxury of time is almost absent. It is this very nature of Post-Live testing that makes it important to automate the regression test cases. This is to ensure quicker turnaround times, reducing cost of regression and human errors. The key differentiator in Post-Live testing is '*time-to-market*' as frequent post production releases need quick test turnaround time.

This paper presents the challenges faced in Post-Live testing and it then showcases Test Automation as a redeemer in this scenario. It presents how automation can redeem the test team from most of these problems with facts and figures and also defines a focused approach towards automation. It adds value to test management centers that are trying to cope-up with release plans for application in production and have no automation approach defined.

The scenario and the approach specified in this paper has been applied in London where the government runs the healthcare program and manages its IT applications. These applications are of critical nature since these are used by numerous end users and have frequent Post-Live patches.

Contents

INTRODUCTION.....	2
ASSUMPTIONS	2
OVERVIEW	2
THE SCENARIO	2
PECULIAR PROBLEMS FACED BY ORGANIZATIONS IMPLEMENTING COTS PRODUCTS.....	2
CHALLENGES IN POST-LIVE TESTING THAT THE TEAM FACED	3
TEST AUTOMATION – THE REDEEMER AND ITS BENEFITS	3
WHEN TO AUTOMATE	3
WHEN NOT-TO AUTOMATE.....	4
TANGIBLE BENEFITS	4
FOCUSED APPROACH TOWARDS TEST AUTOMATION.....	4
LESSONS LEARNT	8
CASE STUDY	11
CONCLUSION.....	11
REFERENCES	11
SPEAKER’S BIOGRAPHY	12

Introduction

Assumptions

This document begins with the assumption that the reader has some knowledge of Test automation. Some experience of working with automation testing or Post-Live testing would be beneficial (but not necessary) to relate to the contents of this paper in a better way.

Overview

This document attempts to discuss the problems faced in Post-Live testing and how test automation can be a redeemer in this scenario. The experience shared, is from a live project and benefits described are as experienced by the client.

The Scenario

The application at the client's end is a live application which is being used at a broad spectrum of public domain in London. It is a COTS (commercial off-the-shelf) product and is implemented 'As-Is' as part of release 0 (zero). The changes or enhancements were planned only in the subsequent releases. Owing to wide gamut of users in public domain - Healthcare, it is very likely that users have specific requirements pertaining to ease of use of the application. This invites the need for enhancements or educating users on how to use the application. A release of a patch to enhance the application needs to be meticulously tested before it makes its way to the real world. Patches released might cause additional defects which need immediate fixes called as Hot fixes. Thus there could be two types of patches after the application goes live, namely, Planned Patches and Hot-fixes. It has to be ensured that the new patch or hot fix not only achieves what it is designed for, but also does not disrupt the existing functionality in any way.

Earlier there was no formal testing established and issues found post-release needed a test team that could churn out test deliverables in shortest possible time. It was realized that Post-Live testing involves re-execution of previous tests as well as upgrading of core functionality regression packs with new test cases thus causing the scope of testing to keep expanding. The direct impact of this expansion of scope is on the testing timelines. Since the solution to this is 'not' to increase the resource strength, the challenge impelled the test team to move towards test automation of the core test cases that were frequently used.

Peculiar problems faced by organizations implementing COTS products

1. Initial release is tested and delivered by vendor thus there is a possibility of lack of application documentation (design, test artifacts etc) and test pack.
2. Extent of testing is not known and needs to be analyzed by studying the stability of the application.
3. Usually the standalone releases are stable but any integration with other existing products creates issues.

4. Enhancements to COTS products have their own issues on design and scalability side.
5. Changes to any single functionality might cause problems in other areas.

Challenges in Post-Live testing that the team faced

1. COTS products were implemented by System Integrators or Deployment Companies as a part of tactical solutions at various organizations. Issues were found in the production which needed quick turnaround time.
2. Due to frequent patches, the scope of testing continued to grow with very short timelines to test. Manually running the same set of test cases would have increased the chances of human errors.
3. Requirements were around new functionality or a defect fix. Thus the focus of change was restricted to the extent of the change and not the end-to-end business flow.
4. Release of an emergency patch required quick test turnaround along with ensuring that the business requirements were met.
5. Post-Live test team (also referred as Live-Support test team) had to understand the application in a short time and analyze the requirements.
6. Release timeline for a hot fix is very low, say a day or a few hours. It was more challenging to perform regression in such a scenario.

Test Automation – The Redeemer and its benefits

When to Automate

As the applications grow and evolve, the size of test regression packs also increases. The challenges of ensuring regression (or executing repeated test scenarios) in shortest possible time, makes way for Test Automation. Test Automation may seem to be a sure remedy for all problems, but given the challenge it cannot form the basis for automation. In practical terms, we first need to analyze whether automation can bring about the expected benefit to the client. To start with, we performed a Proof-Of-Concept (POC) on 2 applications to get a buy-in from the client and its business sponsors.

The question the client always had is how this test automation will prove beneficial. Based on our experience and discussions with the client, we understood the application life-span plan along with a high level overview of upcoming changes. From the information we gathered, it was understood that during every test cycle there was a 80:20 ratio (with 10% tolerance) between the number of existing functionality and new functionality test cases. It was analyzed and established if the existing functionality test cases were automated; it would result in a much faster turnaround time, reducing cost of regression and human errors. This helped the business sponsors to give us a go-ahead for automation.

We mutually agreed that the new functionality test cases or the test cases related to a hot fix will be executed manually for the first time and based on feasibility these should be added to the automation pack subsequently.

We conclude that in the Post-Live testing scenarios where we have tactical solutions with fairly long life of the application released and test cases are of repeatable nature, Test Automation can be considered as a redeemer. Needless to say that test automation of regression test cases brings about quicker turnaround times, reducing cost of regression and human errors.

When Not-to Automate

There were 4 tactical solutions that the client had implemented. After our discussion we agreed that when the life of any tactical solution is small, with very few releases planned or test cases are not of a repeatable nature, it is a better idea to run the test cases manually.

It is important to understand that test automation comes with a cost, which is in terms of the tool cost, cost of skilled resources and most importantly the time required for automation. If the costs can not be recovered over the period of time of running the automated scripts, it is best not to automate in such scenario.

Tangible benefits

Apart from the planned and emergency patches, there were also major application releases (V3 to V6). Due to a well defined framework model (discussed later), it was found that most of the test scripts could be "re-used" by a little tuning, with the exception of few scripts that needed to be modified and new ones that had to be added to the core functionality regression pack.

Benefits of automation are well known to test management gurus; however, below are few specific benefits of automation achieved by Post-Live test team:

1. More than 80% productivity improvement using automation pack.
2. Regression time was reduced from 2.5 man days to 90 minutes.
3. Regression pack is executed on every new build. It is executed in parallel to manual test cases related to hot-fix or a new functionality.
4. Test scripts were shared with deployment team to help them test the application on production environment.
5. Test scripts are planned to be shared with development or vendor team to ensure that the application has an expected level of quality.
6. Automation proved to be an additional check point to ensure that the product met the business requirements.

Facts: The application is in use since February 2006 and the automation pack has been executed since V3 i.e. from May 2006 to V6 in October 2006. We have had 3 main releases, approx 4 to 5 patches per release and 5 to 6 hot-fixes per release. The benefits will continue as the application's life span is for approx next 3 years.

Focused Approach towards Test Automation

This section is a short description of the plan and the focused approach the team followed for this project.

Feasibility of automation was studied by performing a Proof-Of-Concept (POC) on two applications. Mercury Quick Test Pro (QTP 8.2), the existing tool available at client's site was tried and was found suitable. A plan was proposed where a common framework was designed to support multiple applications.

Below is the plan that was proposed and executed in a time boxed approach.

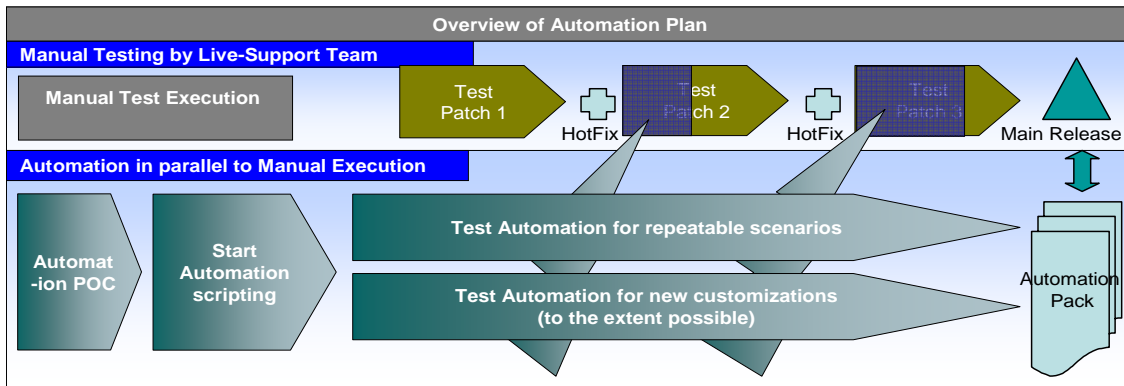


Figure 1: Automation Plan

The plan shows automation activities in parallel to manual testing. This is to ensure there are no delays on releases tested manually. It is implied that automation of new customizations and revision of automation test pack is an iterative process.

Below are 5 pillars of our framework model:

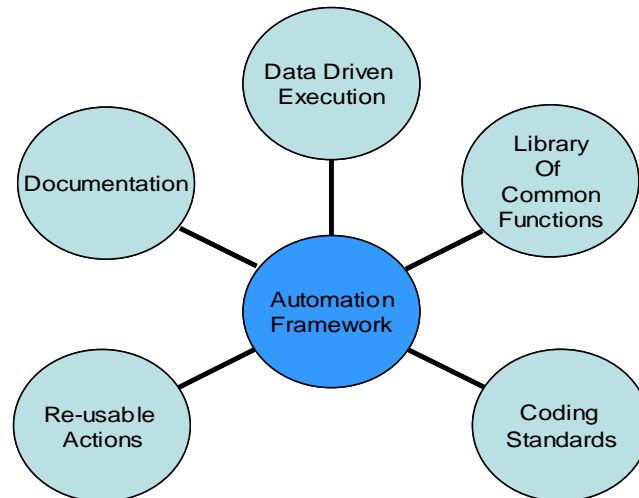


Figure 2: Automation Framework Model

The figure below illustrates the Framework Architecture design:

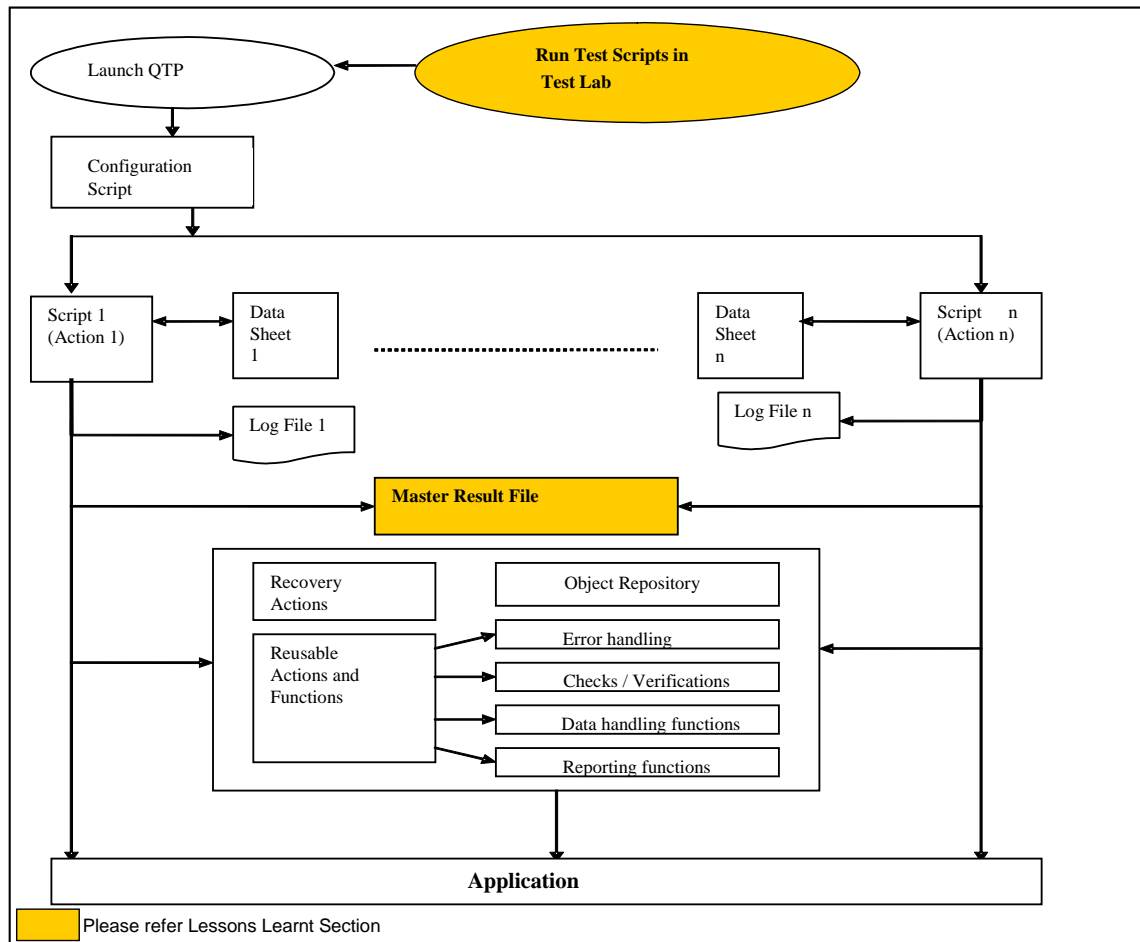


Figure 3: Automation Framework Architecture

- **Data Driven Execution**

Data driven approach for test automation was applied as it separates the data from the automation scripts by moving the data to an input file, called the data sheet. By parameterization of the scripts using the data sheets, scripts were made to run with multiple data sets. Also, changes in data required updating the data sheet and not the script themselves.

The team also ensured that the data created in the data sheet can be used to test 'end-to-end' business scenarios.

The directory structure for storage of all the files and scripts was also pre-defined and documented as part of the framework. Complete directory structure of the automation folder was decided, which included folder hierarchy for Scripts, Library functions, Datasheets, Results, log files and Object repository etc. A shared Object repository was used.

- **Library of Common Functions**

A library of common functions was built. These functions were used across different applications that were planned for test automation. Thus rework was avoided and reusability was exercised.

- **Coding Standards and Naming Convention**

Coding standards were defined and documented so as to make the scripts easily readable and uniform throughout. Naming convention was applied to variables, Scripts, Functions, Log files, Data Sheets, Object Repository, Reusable Actions, and Script Actions etc. Standards were set for the comments within the scripts, such as test case Header etc.

- **Re-usable Actions**

Before starting with the automation of individual test cases, all the test cases were analyzed. Many actions were found to be common and thus reusable. These reusable actions were automated and placed in a folder named "Reusable Actions" under the automation folder hierarchy and were accessed and used by multiple scripts. Few examples of re-usable actions include Login, Import, Registration, Logoff etc.

The test scripts were coded to execute in a functional flow by leveraging on QTP's strong recording and scripting capabilities. QTP capabilities to set recovery scenarios, error handling and result logging were also exercised. Elaborate discussion on QTP is outside the scope of this paper.

Scripts were designed to achieve the testing of an 'end-to-end' business scenario to the maximum extent possible.

- **Documentation**

Detailed documentation (Automation Help Manual) was provided detailing out every aspect of automation, such as coverage of automation, library functions etc. The documentation also gave technical details at script level, covering initial setup, test execution flow, pre-requisites and test data requirements.

This was provided to facilitate maintenance and enhancements of the test scripts for Post-Live test team. The document also proves helpful to other teams who want to execute the regression pack in their test environment.

Lessons Learnt

Infosys test automation team leveraged its past experience and knowledge base to ensure that there were lesser issues during this automation project. But as all projects are not the same, there are always learnings that the team carries as part of enhancing its experience.

There were no learning in areas of skill set, project completion and deliverables, people issues or framework adopted, but there were few in terms of planning coverage, estimating impact of change, limitation of tool and more important the need of having the client's vendor interaction.

The learnings are listed below:

- **Adequate Coverage/Efficient Planning**

Automation coverage missed out time allocation to incorporate script call from test management tool for result reporting. This had to be managed as the client required result reporting in test management tool but given the time constraint this was mitigated by providing standard automation report at script level.

The team failed to estimate any impact that would be due to any changes caused by a new patch while automation was in-progress. Understanding application release plan during automation phase would have helped reduce maintenance work. However a lot of effort was saved due to the robust framework design.

- **Application Setup at client's site**

Dedicated access to application and database is required to ensure no other team is impacted. However this luxury was not possible and thus records created during automation were planned to have unique naming conventions.

- **Automation Tool Setup at client's site**

Client had node license and not server / network license. Thus the team had to ensure adequate license per node was available before start of the project. It is important to understand license type and local desktop setup before start of an automation project.

- **Tool Limitation**

In QTP, where the license is node license, there are issues when multiple team members need to access same Object Repository during automation scripting. QTP restricts access to the shared Object Repository to one user at a time. Thus teams had to plan the Object Repository usage accordingly or use 'Object Repository Merge' feature to ensure there is a single Object Repository created.

- **Scripting / Coding Standards**

Naming convention planned was not in accordance with the client's test management tool's naming convention. This caused re-work on the script call process from test management tool.

Internal review process highlighted the necessity of detailed comments vs. short comments. This is to ensure that modifications, if any, can be carried out smoothly by other team members.

- **Issue Clarification**

As noted earlier, the application is a COTS product. Thus knowing the application architecture was equally important. A lot of issues were sorted out by self reading or analyzing the functionality. Internal review process was executed but review process with client's vendor was missing due to unavailability of required subject matter experts (SMEs).

Case Study

The case study is a snapshot of this project.

Client & Business Need	Solution Approach
<ul style="list-style-type: none"> Client Description – The customer is a leading telecom giant in Europe and has won contract to deploy applications in London's public sector – Health Care Business Need – Achieve testing in post-live scenario in minimum time and less resources 	<ul style="list-style-type: none"> Infosys proposed to define a framework based approach for automation using QTP and Test Director Assessment of the customer's as-is test automation framework by conducting interviews and analyzing the testing artifacts Evolve automation framework using the Infosys' automation testing framework and pilot the same to get approval for the overall program Define time-boxed approach with tracking deliverables, issues and learnings to closure
Engagement Overview	Value Adds
<ul style="list-style-type: none"> Infosys test team provides testing and test automation services A small team ensured automation is achieved as per plan without affecting existing release plan 	<ul style="list-style-type: none"> More than 80% productivity improvement using automation pack – Time reduction from 2.5 man days to 90 minutes Framework based approach ensured future enhancements were possible with minimal maintenance time spent (Application V3 to V6) Automation pack has been executed 6 times since creation and has proved beneficial to the client to ensure there is adequate regression of application done in a short time
Challenges	<ul style="list-style-type: none"> <i>Well done all – we have a happy customer</i> (E-mail dated Wed 26/07/2006) Live Support Test Manager at Client's site
<ul style="list-style-type: none"> Minimal exposure to application architecture and client SMEs Automation tool and environment setup Educating client about the benefits of automation and Infosys test automation framework Time-boxed approach 	
Client Says	

Figure 4: Project Case Study

Conclusion

Test automation in Post-Live testing is introduced to manage quick time to market for releases that are in the form of hot-fixes or patches. Independent testing in this scenario is still evolving and has its own challenges.

As it is said "Well begun is half done". In spite of initial hesitation to take the 'first' step towards test automation in a dynamic scenario such as Post-live, it was experienced that Test automation when carried out in a planned manner, does reap its benefits.

However every project being unique, it must be assessed whether the effort, time and cost to be invested in automation will be returned eventually.

Reference

Independent Validation Solutions (IVS) is an Enterprise Capability Unit (ECU) started in 2001 at Infosys Technologies Limited. IVS integrates processes with systems, tools, templates, guidelines and checklists to deliver better client value. The automation framework used is part of the key methodologies adopted by the IVS group. This can be accessed at its internal web portal.

Speaker's Biography

Nailesh Sampat



Nailesh Sampat is a Senior Test Manager working with Independent Validation Solution's (IVS) division of Infosys Technologies Limited. He has over 12 years of professional experience in Information Technology with 6 years of leadership experience in testing projects. His experience in test management and test consultancy spans across market verticals like Health Care, Retail, Telecom and Investment Bank.

Some of the key highlights of his work are:

- Strong knowledge of testing processes with good exposure to test management tools and QA methodologies.
- Responsible for the overall quality of deliverables, time to market and effective communication
- Manage stakeholder expectations and improved project performance against competing demands on scope, cost and schedule.

Anupama Jamwal



Anupama Jamwal is a Test Analyst working with Independent Validation Solution's (IVS) division of Infosys Technologies Limited. She has over 7.5 years of experience in the IT industry in Testing and Systems Administration. She has effectively executed testing projects in Healthcare, Banking and Finance domain and worked on manual as well as automation testing.

She innovated in her earlier profile by automating important systems administration tasks using Perl scripting. These tasks included automation of WAN link status monitoring, server monitoring and hardware inventory reporting to name a few.

She subsequently worked on automation and manual testing in diverse testing projects for a financial services giant in the US, a global leader in Systems Software, the IT & Multimedia planning department of an ASEAN giant, and for a National Health Service provider in a European country.

