WinRunner to QTP for Neustar

Translation – Case Study

Version 1.0

PREPARED BY:

INTEROPERATE.BIZ, INC.

Table of Contents

Introduction

Interoperate undertook the task of migrating WinRunner scripts for a leading telecommunication company that specializes in creating IP solutions across North America. They specialize in providing solutions that enable switching and porting of telephone numbers in US and elsewhere. The scope of the first project was the "Number Portability Administration Center – Service Management System" that was provided by the company for 4000+ carriers all over North America. The customer thrives on Quality largely because of the critical nature of the business domain they specialize in. This resulted in a large WinRunner code base that they created, and optimized, implementing almost every documented feature of WinRunner in their framework.

Project Size and Schedule

The client had a WinRunner code base of 120,000 lines of TSL code, which are highly functional and reusable and 80,000 lines of GUI map that Interoperate was entrusted with the task of migrating to QTP's VBScripts and TSR files respectively. The project schedule was initially a period of 3 months, which was extended to 6 months as the acceptance schedule was hard to achieve considering the dynamic nature of scripts that were changed almost every other release and there were central core components that might have been already migrated to QTP. The client therefore split the WinRunner scripts into 14 groups which were as functionally independent as possible. Interoperate promised and delivered a minimum of one group per week translated and validated. This was possible due to a high percentage of automation, with several customizations programmed into our translator specific to this client. Such changes are explained in following sections of this document.

Highlights of the translation project

Interoperate was able to successfully translate and deliver QTP scripts that:

- Preserved Functional logic built into the client's compiled modules
- Preserved the reusability of each and every function
- Preserved the highly complex synchronization logic that the client had built-in to the scripts to enable WinRunner to successfully and efficiently synchronize with the client's applets
- Preserved the stringent coding standards set forth by the client
- Were able to successfully execute from HP Quality Center as single tests and Batch Runs yielding the same results with same performance as their WinRunner counterparts
- Preserved the Test Parameters that the clients had programmed into their WinRunner scripts and the variations in test flow based on those test parameters

We were also able to effectively stick to the schedule promised, while finding out problems in the client's scripts and recommending workarounds/fixes for the same.

Technical Challenges Faced

Listed in this section are some of the technical challenges that Interoperate faced in this project

Complex String Operations

The TSL scripts made use of a lot of dynamic strings built with several complex string handling functions. Interoperate's translator successfully identified patterns and handled all such string operations automatically which would otherwise be a time consuming and error-prone process, if corrected/modified manually

Remote Shell Execution Calls

The TSL scripts constructed several remote shell and putty calls dynamically and executed them. Interoperate was able to handle all such cases while preserving the functional interfaces, thus maintaining the reusability of such functions

Composite Arrays (Hash)

The TSL scripts made extensive use of composite arrays which were used to store test data. The Interoperate translator translates this to VBScript Dictionary objects which are reliable, flexible and faster to access than native arrays.

Multi-level Objects

The application under test was a complex Java Applet interface. This interface was configured to be recorded as a two-level application in WinRunner. However, in QTP, the hierarchy levels were dynamic, sometimes extending to 5-levels making it impossible to have a one-to-one mapping between the WinRunner objects and their QTP counterparts. Interoperate identified patterns and made sure that the translator translates the objects in its correct hierarchy. This saved a lot of time that would have gone into fixing these objects every time we encounter them, placing them in the correct position in the tree and adjusting their properties to make sure they are identified. Interoperate boasts of a 100% automatic translation of the client's GUI map.

Extensions Handling

The client's QA team had worked with their development team to add custom java objects and methods to the application and then using these objects and methods in WinRunner scripts for effective synchronization of the testing tool with the JavaApplet. This was a one-of-a-kind customization that Interoperate had seen and was a pretty challenging one to implement in QTP. However, Interoperate was able to implement the same logic in QTP, making use of the same controls and methods to implement synchronization with the AUT. This saved the client a lot of rework to adapt to the new paradigm that is QTP.

Loops

The client had extensively used a variety of loops and made extensive use of programming paradigms like "continue" that were not directly supported by QTP. Interoperate designed workarounds and incorporated these workarounds in the translator so that these changes are automatic and do not need manual intervention or editing. As a result of these factors, we were able to achieve greater than 98% of automation of the client's TSL scripts.

Customized Primitive Implementations

The client's GUI applets did not work as it must have with native QTP primitives. Workarounds had to be implemented for several WinRunner primitives. An example would be to make use of inbuilt methods of java edit objects for those objects where direct QTP primitives failed to set the correct text in the AUT.

Performance Optimization Solutions

Apart from several workarounds designed to meet the functional requirements of the project, Interoperate also designed and suggested several solutions for improving the performance of the scripts. We suggested reordering of test data and alterations of some of their functional algorithms that were taking a longer time. Also, we created a prototype of a different model of execution from QC that is 5x faster than the traditional method of script execution because of the tendency of QC to reload all libraries that are used for every script that is executed from QC.