WinRunner to QTP for GE Transport

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 transportation company GE- transport that specializes in providing transportation services across North America. The company's parent is a Fortune 50 company. They have a communication based transportation system product implemented in Java for management of trains, crews, etc. Quality is important to the customer because of the critical nature of their business domain.

Their quality assurance efforts resulted in a large WinRunner code base developed over a number of years. Their WinRunner test-scripts exploited almost every documented feature of WinRunner in the framework. The customer wanted to migrate this WinRunner code base to QTP. Client resorted to use Interoperate, since Interoperate has the best-in-class software product for automatically migrating WinRunner scripts to QTP. Interoperate's WR2QTP system provides upwards of 97% automation, while guaranteeing that the generated code will be correct, as well as readable and maintainable.

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 client's own tool to run test requests as single tests and Batch/Regression 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
- Customized translation and API implementation to support all of the client functional requirement and real time human behavior simulation of manual testing.

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 them.

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

Dynamic Dialogs Creation

The object repository for the project had Java Windows used as standalone windows and sometimes as part of the other windows which QTP identified as Java Dialogs. As the object repository only had them as standalone Java windows, we developed code to dynamically create Java Dialogs for the same window in case it's opened from another parent window. This completely removed the manual work involved to add these windows as dialogs in each and every window it's accessed. This also removed the manual work of finding such scenarios as the code took care of it depending on the scenario involved.

Complex Analog functionality

The client scripts used almost every analog function (e.g coordinate based mouse clicks, drag and drop etc) to simulate the real time manual testing efforts. Lots of these functions were not working in WinRunner and client had developed work around for that. We were able to provide the exact same implementation for all such scenarios in QTP though analog functionality in QTP is still in its immature state. We were able to get all complex analog functions working in QTP with workarounds to handle the cases where it was unstable. It was very difficult and challenging to implement the real time manual testing simulation through analog functions in QTP but we were able to provide very stable solution to client.

Complex Keyboard Typing

Client scripts also simulated keyboard-typing and button-press for real time manual simulation. (E.g. Ctrl+A, Shift+Delete). Such keyboard strings were complex and our API needed to handle all such events with correct context. We enhanced our library API to successfully handle all such keyboard typing strings with different functional buttons to provide very stable implementation in QTP.

Loops Statements

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 them 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 Java Table Implementations

The client GUI application was complex and it had implemented Java tables with drop-down lists to select column values. QTP did not identify such column types as direct java list. Client's click-and-type solution in WinRunner was very unstable. We wrote our custom APIs to handle such cases: this customization dynamically added Java list inside the Java Table and then selected value from the list with QTP native methods. After successful implementation of this scenario, we also implemented optional functions in client scripts to simulate the click-and-type method for stable real-time human behavior simulation.

Performance Optimization Solutions

Apart from several extensions 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 delivered faster APIs for file handling to reduce file read/write time in the client scripts. Apart from this, we also found unnecessary TSR files which were not used by client scripts and eliminated them to significantly improve script loading time. Our client also required us to implement error logging. Errors were logged in a file (for uninterrupted runs) and on the screen (for debugging runs) based on the value of a flag.

Overall, the project was successful, and Interoperate successfully helped its Client achieve its WinRunner to QTP migration goals.