Rational. software

Advancing Toward Test Automation through Effective Manual Testing

Bob Levy,
Lead Product Manager – Functional Test
Dennis Elenburg,
IT Specialist – Rational Products

Contents

2 *Introduction.*
3 *Framework Benefits Without Framework Cost.*
4 *Data-Driven Tests.*
4 *Extract User Interface Detail From Test Scripts.*
5 *Reuse of Repeated Flows.*
5 *Problem.*
6 *Solution.*
7 *Effective Manual Testing.*
8 *Advancing Toward Test Automation.*
11 *Conclusion.*
11 *About the Authors.*

Introduction

The fast pace of modern software development creates tremendous challenges for the test team. Online software updates and frequent release cycles generate a "moving target" for test automation efforts.

Test organizations that have not yet advanced beyond simple record/playback automation soon learn that recorded test scripts become obsolete within a few project iterations of the software being tested. Historically, sophisticated test organizations attempted to mitigate this problem by investing heavily in automation frameworks or keyword-driven approaches. They expected this approach would generate a return on investment over time. However, in today's marketplace few test organizations can afford to assign their most senior team members to framework building that yields little immediate payback.

This paper outlines IBM's recommended approach to the test automation challenge, knowing that most testing is first done manually. IBM Rational Manual Tester promptly increases the effectiveness of manual testing. It also encourages testers of all skill levels to build linked content using drag and drop and copy/paste when writing manual test scripts. This linked content in Manual Tester's reuse view allows a test team to focus automation efforts on the most frequently repeated flows first. IBM's approach yields earlier return on investment and accelerates reaching the benefits of keyword-driven testing while providing better test script documentation.

This paper will walk through a best practice scenario for using Manual Tester to more naturally organize test content. Once organized, the test content is ready to be automated using IBM Rational Functional Tester. This modular

approach beginning with Manual Tester reduces maintenance costs for both manual and automated testing. By adopting this incremental and iterative approach to test automation, you will gain immediate return on investment from the manual testing effort while simultaneously advancing your test team toward sound automation practices.

Framework Benefits Without Framework Cost

Test automation experts are often vocal about the benefits of frameworks for sustaining automation across multiple releases of the software being tested. However, these benefits come with the price of a large upfront investment in building the framework. Rather than revisiting the difficult business case for upfront investment, we will show how effective manual testing realizes immediate return on investment while moving the testing organization toward framework benefits without the upfront framework cost.

We can distill the essence of automation frameworks into a few core ideas. Here, we will cover the most important of these ideas: (1) data-driven testing, (2) extracting user interface detail from test scripts, and (3) reuse of repeated flows. Tool support for the first two ideas is not new, so we will briefly explain how the IBM Rational tooling implements these out of the box. However, lack of an approachable solution for the third idea has prevented most non-developer testers from achieving sustainable test automation. We will devote the rest of the paper to the important idea of "reuse of repeated flows." We will show how testing organizations can obtain framework benefits without framework cost by adopting IBM's approach of advancing test automation through effective manual testing.

Data-Driven Tests

Manual testing often requires the tester to enter data into the application under test and verify that the resulting information matches expected values. If the tester inadvertently enters incorrect data or overlooks a data mismatch, the test results become invalid. By automating this data entry and data validation process for the tester, Manual Tester reduces human error.

Testers often need to perform the same transactions many times with different data. Functional Tester's data-driven testing wizard makes it easy to automate this work. Using a spreadsheet-like data editor, testers create or import customized data sets to be inserted into the script during playback.

Manual Tester and Functional Tester deliver data-driven testing out of the box without the need to construct separate framework infrastructure.

Extract User Interface Detail From Test Scripts

Functional Tester automatically creates a map of user interface details while the tester records test scripts. This object map stores information needed during test execution in one convenient place. Change to existing user interface details no longer requires the tester to revise each test script. The tester can make revisions in one place since all automated scripts reference this centralized object map.

Functional Tester is further advanced by its patent-pending ScriptAssure™ technology. ScriptAssure™ is a matching system that finds the best reasonable match within customizable thresholds for changed user interface detail. This capability safeguards test scripts from modest user interface revisions typical from build to build.

Reuse of Repeated Flows

Testing can be tedious and repetitive. Test teams write and execute hundreds or even thousands of test script documents for a given software project. Testers must revise these documents as developers change the software being tested. These documents, and subsequent automated recordings, share many common flows including:

- *Setting up the application under test (e.g. launch, login)*
- *Filling out forms (e.g. create customer account)*
- *Navigating to a specific place in the application (e.g. navigate to order entry screen)*
- *Performing common verifications (e.g. does the database correctly reflect the transaction?)*
- *Exceeding expected values or boundaries of the application's function (e.g. does the right error handling get triggered when entering an invalid credit card number?)*
- *Data-driving a subflow of the test script (e.g. log in and perform the same transaction for each of the 50 different user accounts)*
- *Performing higher-order business flows, including those composed of other repeated flows (e.g. place order, sell stock)*

Problem

Most testers rewrite or copy/paste these repeated flows across their many test scripts. Record/playback misleads many testers into using automation to separately re-record each instance of a repeated flow. Both approaches result in content duplication.

Content duplication in test scripts is insidious because software change requires testers to revise every script in which the repeated flow occurs. This overhead discourages test teams from maintaining proper documentation to ensure a repeatable and consistent test effort. Maintaining this duplicated

content is tedious and increases the risk of human error. This root cause prevents test teams from sustaining automation across multiple software releases.

Solution

Imagine writing tests in such a way that you could update repeated flows in one place instead of everywhere it is used. If this new way of writing scripts accelerated the writing, then testers could easily surmount each release's overhead work. Testers will have time to build more creative tests and ship better software.

Manual Tester allows non-programmers to easily write test scripts with repeated flows reused as linked content (Figure 1). Simply drag and drop (reuse view) or copy and paste by link (CTRL-L) to create modular test scripts. IBM Rational's patent-pending user interface simplifies writing modular scripts regardless of skill level. Functional Tester makes it easy to record scripts that call other scripts. This retains the modularity discovered while writing and refining manual tests.

Reusing repeated flows as linked content is essential since traditional manual testing and record/playback automation approaches require you to duplicate the revision of each script performing the repeated flow.

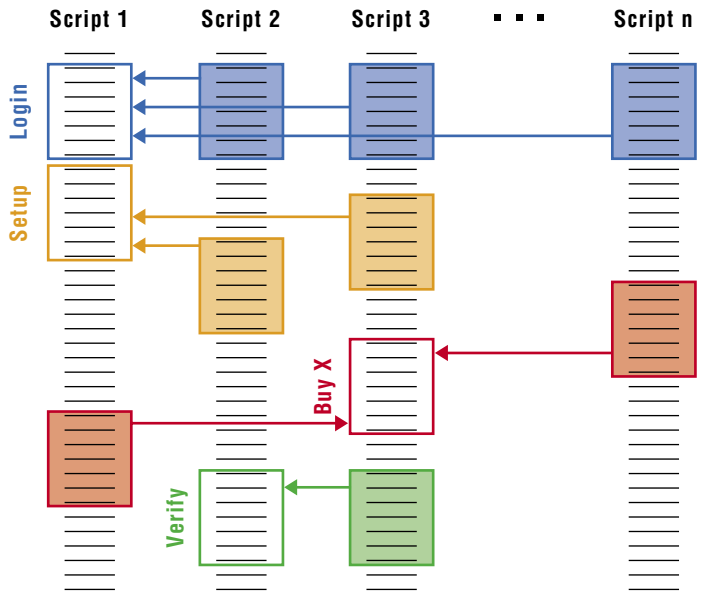


Figure 1: Reuse of Repeated Flows as Linked Content

Effective Manual Testing

Manual Tester is a manual test authoring and execution tool that encourages a modular, building-block approach to test authoring. Tools traditionally used to document manual tests such as MS Word and Excel fail to encourage or even enable this building-block approach. Manual Tester encourages modularity and reuse in its rich text editor. It also supports image and file attachments to improve test readability, and allows testing teams to import pre-existing Microsoft Word and Excel-based manual tests.

By encouraging modularity, Manual Tester allows the tester to assemble tests from reusable flows that document a set of steps for testing a small area of the application. Testers can then reuse these flows to assemble the many test scripts required to validate their application.

Manual Tester is more than just an authoring tool; it also improves productivity of manual test execution with assisted data entry and verification features. These features speed up manual test execution and improve results by reducing human error. Manual Tester also allows testers to import and export test results to comma-separated value files compatible with preferred third-party tools including spreadsheets, databases and other reporting and analysis tools.

You can find a more complete introduction to Manual Tester at:
http://www-106.ibm.com/developerworks/rational/library/content/RationalEdge/oct04/wilkey/index.html

Advancing Toward Test Automation

While writing or importing manual scripts in Manual Tester, anyone supporting the test effort can organize his or her manual test scripts into reusable modules as he or she works. These beginning steps lay the groundwork for keyword-driven testing, positioning the team for sustainable automation.

Manual Tester includes a reuse view where the test team shares reusable flows. This list helps testers quickly identify which reusable flows they should record during the first pass of automation with Functional Tester. Functional Tester records scripts in the tester's preference of Java or VB.NET to ensure alignment with available skills.

A test automation engineer would perform the following steps to record a repeated flow:

(1) *Launch Functional Tester and Manual Tester; open the manual test script of interest.*

(2) *Launch the application under test and perform any instructions in the manual test script that precede the module to automate. Follow the steps in the manual test script to this point. Testers may pause recording at any time to review the manual test script.*

(3) *Now that the application is in the correct state to begin automating the repeated flow, launch Functional Tester's recording toolbar.*

(4) *Record the steps outlined in the reusable module (repeated flow) chosen from Manual Tester.*

(5) *Stop recording and save the automated script module for reuse.*

Now, any time the test team encounters that module while testing manually, the team can simply invoke the automated script. They can even sequence the flow of manual and automated test pieces using the Rational test management feature set. This will ensure all testers maximize their use of available automation.

While automating the most commonly recurring modules, the testing team will also identify entire tests that they should automate. The test team can advance to the next level of automation when recording entire tests by reusing the previously recorded reuse modules.

A test automation engineer would perform the following steps to build an entire test script from reusable flows:

(1) *Launch Functional Tester.*

(2) *Use Functional Tester's "start application" feature to launch the application under test.*

(3) *Begin recording.*

(4) *Perform the manual steps against the application under test.*

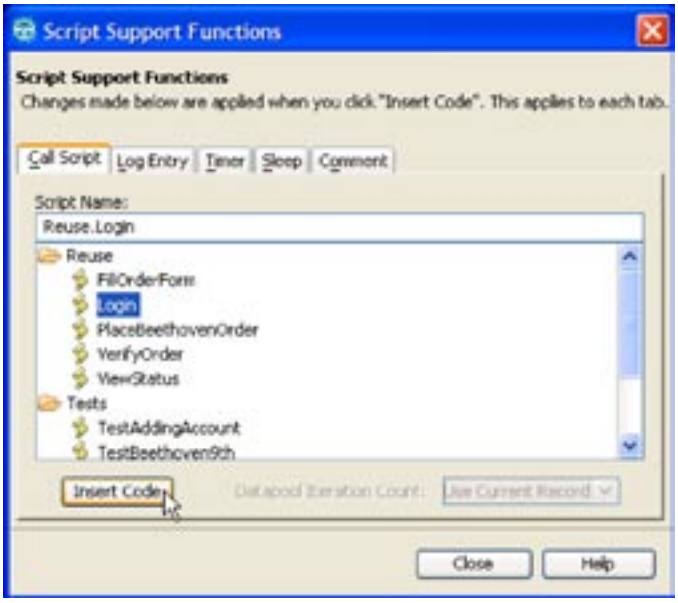
(5) *When you encounter a reuse module in your manual script, use Functional Tester's "Call Script" capability to call the previously recorded script (Figure2).*
This preserves modularity in your automated tests.
Briefly pause recording to ensure the software under test is in the correct state to continue.

(6) *Stop recording in Functional Tester when you complete the test.*
Your entire manual script is now automated.

Reuse is essential since traditional record and playback approaches require you to go back and update repeated flows everywhere that same task is performed. With modularity, required updates are centralized to the single core building-block, overcoming test script decay. This modular approach lowers total cost of ownership to a point where testers can sustain scripts by centralizing much of these required updates over the lifetime of your software.

Figure 2: Functional Tester "Call Script" Capability

then



Conclusion

IBM Rational's Functional Test tools include rich support for automation frameworks, even for teams who lack the time or technical skills to develop elaborate infrastructure. The manual test scripts serve as readable documentation to help team members quickly understand the intent of automation scripts and guide sustainable automation. Indeed, testers of all skill levels advance toward the benefits of keyword-driven testing as a free by-product of more effective manual testing using Manual Tester.

About the Authors

Bob Levy serves IBM Rational as a lead product manager, managing strategy and direction for the Functional Test line of business. His portfolio includes the award-winning IBM Rational Robot, Functional Tester, and Manual Tester. He has served as president of the Boston Product Management Association from its May 2001 founding to January 1, 2003. Mr. Levy has earned a bachelor's degree in computer science and an MBA from Dalhousie University in his birthplace of Halifax, Nova Scotia, Canada.

Dennis Elenburg is a certified Rational product specialist with 15 years of experience in software development and consulting. He currently works as an IT specialist on an IBM Rational sales team based in Dallas. He holds a physics degree from Austin College.

IBM Corporation
Software Group
Route 100
Somers, NY 10589

Produced in the United States of America
05-05

G507-0998-00