White Paper

The Matlen Silver Group, Inc.
270 Davidson Avenue
Somerset, NJ 0873

Phone 732 469 2866
Fax 732 271 8443

Velocity Digital Toolset

Celerity Software Assurance

By Alex Azan and Michael Corley

Contents

INTRODUCTION

The Matlen Silver Group, Inc.'s Celerity Software Assurance (CSA) division was founded to help its customers address their Quality Assurance and Quality Control challenges. Many Information Technology departments acknowledge their deficiencies in software quality practices including processes, infrastructure, tools, environments and properly trained human resources.

While software testing tools have been existence since the creation of software itself, there is a new breed of sophisticated tools whose successful implementation is very often compromised. Perhaps the most common reason for the lack of success with these products is an absence of human capital that has chosen to truly master this technology. Today's IT implementations are very complex. While the end-user interacts with a friendly graphical user interface, there is a complex integration of various software and hardware components operating behind the scenes. Distributed systems have greatly increased the challenges of properly testing software systems. For these reasons, Matlen Silver has formed their CSA division and has specifically selected software testing tools as their primary proficiency.

REGRESSION TESTING TOOLS

There is a very large array of software tools available to support the testing mission and perhaps automated regression testing tool are the most misunderstood. Leading automated regression testing tools include Mercury Interactive's WinRunner®, Rational Software's Robot, Compuware's QARun and Segue Software's SilkTest. The primary concept of these tools is that they record an end-users keyboard and mouse interactions, and then play them back to regression test an application. By performing a workflow through an applications GUI while entering and verifying data, certain application logic can be validated. It is easy to understand the benefit of these tools. Some have been listed below:

- Automation can alleviate staff from much of their manual testing responsibilities, however it is generally not possible to automate all testing
- Automated tests are reliable and consistent, while manual test execution is prone to human error
- Automated tests execute much more rapidly that manual tests, decreasing the regression testing lifecycle
- Automated tests can be executed unattended to maximize the test execution workday
- Automation digitally captures your organizations testing knowledge, reducing your reliance on key personnel to participate in the testing effort
- Automation can provide a significant amount of reporting to verify test execution and the execution results
- Automated tests can perform functions that are not feasible by human resources

Another compelling reason to implement an automated regression testing tool is the overall improvement in Quality Control practices that will inevitably result from implementing automation. Matlen Silver CSA generally recommends a scaled approach to implementing automation; begin by selecting something

small that can succeed in order to gain experience and additional management support. The list below states many of the dependencies required for an organization to succeed in automation. Note that many of these items are required for effective software testing without automation.

- Test plans
- Test cases
- Baseline test data
- A process to refresh or rollback baseline test data
- A dedicated test environment (stable back-end and front-end)
- A dedicated test lab
- An integration group and process
- A Test case database (track and update both automated and manual test cases)
- A mechanism to rank/prioritize test cases per test cycle
- Coverage analysis metrics and process
- Defect tracking database
- Risk management metrics/process
- Version control system
- Configuration management process
- A method/process/tool for tracing requirements to test cases
- Metrics to measure improvement

Selection criterion is also a key factor in automation success. There are many facets to consider when choosing what to automate. Some of these factors have been listed below:

- The testing mission must be achievable with an automated solution
- Availability, completeness and quality of documentation (business requirements, functional specifications, use cases, test cases, test procedures, etc.)
- Stability and accessibility of the Application-Under-Test (AUT)
- Stability and accessibility of the test environment
- The test tools compatibility with the AUT
- Scope/complexity of the application and test procedures to be automated
- Implementation timeframe
- AUT life expectancy
- Frequency and magnitude of new software releases
- AUT automation testability
- Availability and skills of resources to participate in the automation effort

For additional information about implementing test automation, please read The Pitfalls of Test Automation located at http://www.matlensilver-csa.com/emedia.

IMPLEMENTATION METHODOLOGY

Introduction

All of the factors listed in the previous section are very important considerations when implementing automation. Perhaps the largest area of misunderstanding in test automation is the actual implementation methodology or approach used to construct your automated test script library. Automated regression testing tools are powerful programming tools and are most effective when implemented by applying sound software development principles. They should be implemented by programmers and a well designed architecture should be considered.

Matlen Silver CSA is has developed an implementation methodology named Enterprise Modular Test Methodology™ (EMTM) and a suite of software tools named Velocity Digital Toolset™. Velocity allows Matlen Silver CSA to build testing systems based on their EMTM architecture as well as fully exploit the testing tools powerful features with a common function library pre-built by Matlen Silver CSA.

Linear Scripting

Linear Scripting is the most common approach used in developing an automated test script library. It is also the approach which causes organizations the most amount of frustration when implementing these tools. There are three primary components that comprise an automated test script library developed using a Linear Scripting implementation. These components have been defined below:

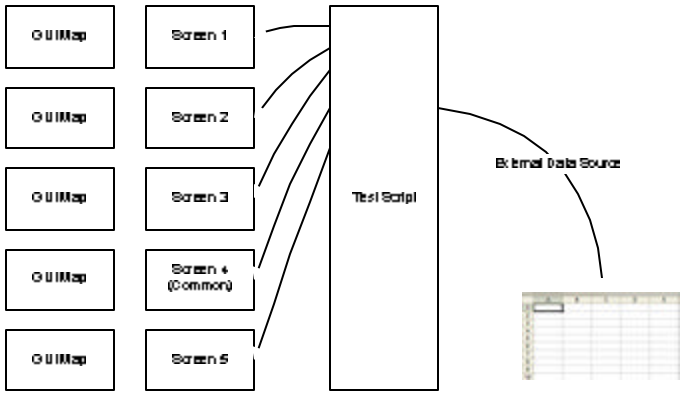
- GUI Mapping Files – used by the Test Script to locate objects/control on an application screen or page view (these files are not part of a Robot implementation)
- Automated Test Scripts – recorded instructions (in native test tool code) that perform or execute the intended test scenario
- Test Data Tables – contain test data values that are used by the automated test script to execute numerous data permutations (i.e. the test script accesses "Smith" from the test data table column for Last Name and places it in the "Last Name" edit box within the application)

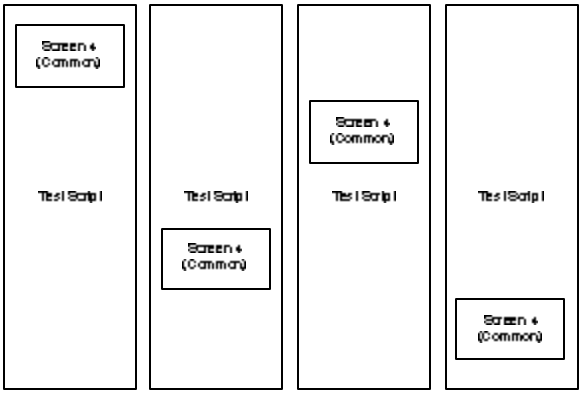
To create an Automated Test Script using a Linear Scripting approach a Test Engineer first creates GUI Mapping Files then steps through a manual test procedure on an end-user workstation while the testing tool is in record mode. The tool captures the end-users keystrokes and mouse movements. Test script code is dynamically generated by the test tool during the record process. This code comprises an automated test script. Quick results are seemingly produced with a Linear Script however the following issues exist:

- Test data values are hard-coded in the automated test script and the Test Engineer must create external Test Data Tables in order to execute numerous data permutations for the scenario recorded
- Additional logic must be coded into the script in order to verify data
- No reporting takes place
- Additional logic may need to be coded into the script in order to achieve other test objectives

One apparent drawback to a Linear Scripting implementation is standardization. Typically automated test scripts are developed and maintained by multiple Test Engineers. Programming standards must be put in place to ensure that scripts can be easily maintained. Furthermore, Test Engineer 1 may be validating data and reporting one way and Test Engineer 2 may be performing the same task using an entirely different programmatic approach.

One of the most profound drawbacks of a Linear Scripting approach which may not be immediately apparent is test script maintenance. The following graphic illustrates in Linear Scripting design denoting a system with five unique screens. GUI Map files are created for each screen. Note that the test script traverses through screens to execute the intended business scenario.





Engineers using this approach. As previously stated the code generated by the testing tool during a linear recording lacks many of the robust features and logic that a Test Engineer would want to place in their script to maximize the benefits of implementing the test tool. A good example is reporting to aid in defect analysis and resolution.

Enterprise Modular Test Methodology (EMTM)

Matlen Silver CSA implements a methodology named Enterprise Modular Test Methodology™ (EMTM) to address the script maintenance issue caused by a Liner Scripting implementation. This is accomplished by modularizing automated test script code. The components that comprise an EMTM implementation have been outlined below:

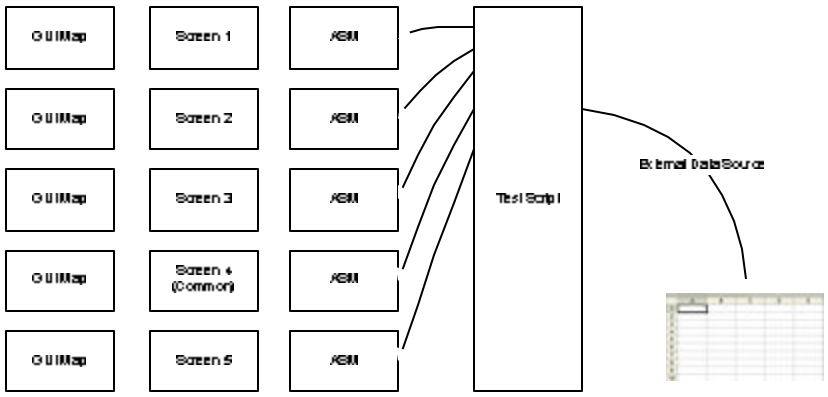
- GUI Mapping Files – these files are used by the test tool to locate controls or objects on a page view; one is typically created for each screen or page view within the application
- Application Screen Module (ASM) – ASMs are constructed for each unique screen or page view within the application and contain all of the structured code required for the ASM to interact with all of the controls on that particular screen
- Automated Test Scripts – in an EMTM implementation, the automated test script has become simplistic choreographed calls to ASMs
- Common Function Library – part of the Velocity Digital Toolset™, these two libraries (one in native test tool code and one as a standard Windows DLL file) contain functions that exploit and extend the capabilities of the chosen test tool
- Test Data Tables – contains test data values that are used by the automated test script to execute numerous data permutations; there are typically two columns for each control in the application, one for a test value and one containing a verification value

An EMTM implementation begins with the construction of GUI Mapping Files. These files are required as part of implementations with Mercury Interactive's WinRunner® and Segue Software's SilkTest, but not with Rational Robot. Robot embeds the logic used by the script to identify controls within the individual test script code. This practice is concerning from a maintenance perspective therefore EMTM requires the creation of GUI Mapping Files with a Rational Robot implementation.

The next step in an EMTM implementation is the construction of Application Screen Modules (ASM). An ASM is created for each unique page view within the AUT. An ASM contains all the logic required to interact with every control/object within that particular page view. ASMs make calls to the Velocity common function library to interact (perform test actions) with application controls and perform other tasks such as reporting. ASMs can be developed to accomplish various testing objectives such as business testing, screen data capture/verification, negative testing, and GUI/Controls verification testing.

Once GUI Mapping Files and ASMs have been created, Test Data Tables templates are created. Two columns are created for each control/object where test script interaction is required. One column exists for a test value and one column exists for a verification value.

Once the test automation framework has been created (GUI Mapping Files, ASMs and Test Data Tables) automated test scripts are constructed. In an EMTM implementation test scripts are simplistic choreographed calls to the ASM library. If there is data in the external data table, an action is performed on that particular control. If there is no data, then no action is taken. This is illustrated in the graphic below.



The Velocity Digital Toolset™ contains the following integrated components:

Console

One of the disadvantages of implementing a modular automated testing framework is implementation timeframe. Building the framework is time consuming. Matlen Silver CSA has developed a construction tool named "Console" that builds the EMTM framework very rapidly. By placing the Console in spy mode and simply navigating through the AUT, an inventory of the AUT is captured. This includes Window property and control property information. Once an inventory has been collected, the Console will automatically create the framework components based on pre-developed and tested code templates created by Matlen Silver CSA.

The Velocity Console is currently not a licensed software component, but used exclusively by Matlen Silver CSA to build test automation frameworks.

Common Function Library

GUI automated regression testing tools contain powerful programming capabilities through large function libraries. Test Engineers must have expertise in programming and a detailed knowledge of these libraries in order to exploit them. Matlen Silver CSA has developed a Common Function Library for performing business testing of applications using WinRunner and Robot test tools. ASMs call functions that live in this enterprise library. Suppose there is a new reporting requirement that must be implemented. Since test script code has been abstracted down to the function level, Test Engineers would only be required to change the one function that handles reporting and all test scripts in the enterprise that used that function would inherit the change. Clients who license this library from Matlen Silver CSA rapidly advance their utilization of the testing tool.

There are two libraries that comprise the Common Function Library. One created in native test tool code and one provided as a DLL which is loaded in the test tool and contains MFC style functions to fills gaps that exist in the test tools API.

Database Management System

The Velocity Database Management System provides two primary functions. The AUT inventory data captured by the console is stored in this database. In addition, during automated test script execution data is stored in this database which is accessed by the Portal. The database must be Microsoft Access 97 or above or Microsoft SQL Server v7 or above.

Portal

The Portal is used for comprehensive reporting including Object Property Reports, Test Run Log Reports, Manual Test Procedures, System Profile Information Reports and AUT Dependency Profile Information Reports. The Portal also provides the ability to delete records in the Database Management System.

Test Data Table File Management Tool

The Velocity Digital Toolset also contains a data file management tool which permits easy manipulation of test data table columns and rows.

Velocity System Features

Envision Velocity as a series of Digital Test Engineers that each has a series of specific functions.

Automation Engineer

Function:	Value:
■ Generates GUI Mapping Files ■ Generates Test Table Templates ■ Generates Business Testing Application Screen Modules (ASM) ■ Can generally support custom controls and easily adapt to client specific coding standards ■ Provides an automated approach for QA Analysts or End-Users to capture test scenarios	■ Automated systems are developed from proven architectural models and implementation strategies ■ The value of a modular, single point of maintenance testing system is realized without an extended development lifecycle ■ Adherence to the highest possible guidelines for script development is always maintained ■ System changes requiring modifications to automation components can be easily and quickly mitigated through the custom function library ■ All code is in the native test tool

	language

System Engineer

Function:	Value:
■ Captures test-lab workstation system/configuration information during each test execution ■ Captures application dependencies during each test execution	■ Silently monitors test lab activity on target applications and maintains historical data on the test mission ■ Data is readily available for troubleshooting ■ Comparisons can easily be made between workstation configurations ■ Comparisons can easily be made between application versions

Reporting Engineer

Function:	Value:
■ Test Run Log Report ■ Manual Test Procedures ■ Low Level System Design documentation ■ Access to the Velocity database management system to support custom queries and reporting requirements	■ Manual Test procedures can be easily and quickly developed via recorded information ■ System audits can be supported by printing low-level system design documentation

System Requirements

Console:	Windows 2000 and Internet Explorer 5.5
Non-Browser Based Applications:	Systems developed using Microsoft's Visual Basic and Visual C++ (32-bit)
Browser Based Applications	HTML, DHTML, Active Server Pages (ASP), Java Server Pages (JSP) Active X, VB Script, JavaScript & XML

Testing Tools:	Mercury Interactive's WinRunner® v7.01 and above Rational® Robot v2002 and above
Database Management System	Microsoft Access 97 or greater SQL Server 7.0 or greater
Portal:	Microsoft Internet Information Server (IIS)

COST BENEFIT ANALYSIS

Introduction

There are many approaches to determining the cost benefit or return-on-investment of implementing an automated regression testing solution. It is very important to note that there are many compelling benefits for implementing test automation and some of these benefits are soft and difficult to measure. Releasing poor quality software can bear negative financial and brand name consequences in addition to the political consequences to the departments deemed responsible within the organization. Several formulas have been provided to help you discover how to best propose or measure test automation within your organization.

Enterprise Modular Test Methodology

The value of a single-points-of-maintenance solution delivered with an EMTM implementation is very compelling. Based on Matlen Silver CSA benchmarks, the efficiency of maintaining an automated test library constructed with an EMTM model is 60 – 99% compared to a Linear Scripting implementation. Modular based testing implementations are recommended by the tool vendors and test automation subject-matter-experts. Organizations that engage Matlen Silver CSA inherit a world-class implementation model and avoid the costly mistakes of not adopting such a robust methodology early on.

Velocity Digital Toolset

An easy measure of Velocity's value is its custom function library. Matlen Silver CSA has invested over 2,000 hours into the design and construction of this library which will continue to evolve. Organizations who purchase Velocity obtain this library for a fraction of its construction cost. The Velocity Console allows Matlen Silver CSA to construct testing systems based on EMTM with substantial precision and speed. The Console can construct EMTM artifacts in a fraction of the time required to build them manually.

Manual vs. Automated Testing

Implementing automation frees staff to work on other activities or projects. Every script that is automated is one less script that must be manually tested, however both manual and automated test scripts must be maintained. Keep in mind that it is rare that all manual test scripts can be automated. Based on the example below, an initial investment of $200,000 in automation implementation reduces recurring

regression testing maintenance costs by 34%. A return-on-investment is achieved five months after automation as been implemented.

Recurring Cost of Manual Regression Test Script Maintenance:	$35,000.
Recurring Cost of Manual Regression Test Execution:	$90,000.
Cost of Automation Initial Implementation	$200,000.
Recurring Cost of Automated Regression Test Script Maintenance:	$15,000.
Recurring Cost of Automated Regression Test Script Execution:	$25,000.

Cost of Quality

Defects are least expensive to repair early in the software development lifecycle and become more expensive as the application nears production. Several studies have indicated that software defects released into production can be 80– 100 times more expensive to repair. The following table provides an example of how to determine the cost of quality.

Total Number of Defects	200.0
Percentage Identified Prior to Release	50.0%
Cost of Fixing Defects Prior to Release (Per Defect)	$ 100.00
Cost of Fixing Defects After Release (Per Defect)	$ 500.00
Number of Defects Found Prior to Release	100.00
Cost of Defect Removal Prior to Release	$ 10,000.00
Number of Defects Found After Release	100.00
Cost of Defect Removal After Release	$ 50,000.00

Test Cycle Workday

Automation tools can execute tests 24-hours per day, 7-days per week. Organizations that currently use a single 8-hour shift for their test cycles will immediately realizes a 67% reduction in elapsed time by adopting a 24-hour per day test cycle. If you factor in weekends and holidays, the benefits become even more compelling. As a general rule, once an organization has fully automated a given test procedure, the test cycle can easily be compressed by 50%, more frequently by 75%. Consider the following example:

Current Test Execution Workday (Hours)	6.00
Execution Workday with Automation (Hours)	18.00
Additional Test Execution Hours Per Annum	2,640.0
Value of Additional Test Execution Workdays	$ 198,000.00

Accelerated Time to Market

Since automated scripts can execute many times faster than manual tests, automation can reduce the testing lifecycle. Since test automation compresses the regression testing lifecycle, it aids in accelerating time to market. The following example can serve as a guide:

Time Savings (Days) With Automation	10.00
Units Produced Per Day	100.00
Margin Per Unit	$ 500.00
Value of Reduced Time to Market	$ 500,000.00

CONTACT

Michael R. Corley

Vice President

(732) 469-2866 Ext. 5132

The Matlen Silver Group, Inc.

Celerity Software Assurance Division

270 Davidson Avenue, 8th Floor

Somerset, NJ 08873

http://www.matlensilver-csa.com/