
H. Ferhan Kilical, Ph.D.
Technical Fellow, Electronic Systems
(Test, Test Automation, SOA, Performances Test and Agile Methodologies)

Presented at the 22nd Systems and Software Technology Conference (SSTC), 26-29 April 2010, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License

Do any of these problems sound familiar? Too many failed scripts? Slipping schedules? Automation tools that never get off the shelf? Incomplete test coverage? Do more with less, and be creative in performing the business. Then, come and listen to the story of Warfighter’s mission critical application testing. This case study reveals how testers used automated functional, performance, and Service Test scripts for a major mission-critical application that had to meet the most rigorous quality standards. Also, working in an agile environment, the team managed end-to-end requirements and defects and performed functional, SOA and performance testing. With risk based and automated test strategies, the team was able to do automated regression and smoke tests in a fast-paced development? agile environment and managed test results, test sets, and test-related artifacts. It was a very successful project, completed on time with very limited resources.
Abstract

Warfighter’s mission-critical system: Agile automated testing and test management

Do any of these problems sound familiar?
- Too many failed scripts
- Slipping schedules
- Automation tools that never get off the shelf
- Incomplete test coverage
- Do more with less, and
- Be creative in performing the business

Then, come and listen to the story of Warfighter’s mission critical application testing.

This case study reveals how testers used automated functional, performance, and Service Test scripts for a major mission-critical application that had to meet the most rigorous quality standards.

Also, working in an agile environment, the team managed end-to-end requirements and defects and performed functional, SOA and performance testing.

With risk based and automated test strategies, the team was able to do automated regression and smoke tests in a fast-paced development – agile environment and managed test results, test sets, and test-related artifacts. It was a very successful project, completed on time with very limited resources.
• Special thanks to the project team at Mission Systems, NG, Herndon.
Complex, Mission Critical Applications & Testing

One NET

Joint Health Service

Decision Tools

Joint Logistics Service

Plan, Prioritize Data

Use Data, See Data, Capture Data

Application

Battle Command

Common Services

Joint Operations

Force Preparedness

Anywhere, Any User

Location, Capture Data

WarFighter’s Needs
General Info for Testing & Integration in Government Contracting and Test Automation

- **Keeping Track of:**
  - Requirements
  - Defects
  - Test cases, test processes and test plan

- **Managing Testing Cycles**
  - Problems in infrastructure and scalability
  - Ongoing implementations and significant development initiatives
  - Significant degree of customization and integration
  - Limited resource availability
  - Integration with Complex Portal and Identity Management Solutions

- **Weak Testing Methodology**
  - Manually intensive
  - People - Not process driven
  - No automated testing capabilities
  - Not trusting anyone else to test
• When a major government contractor delivers software, that software must comply with the most rigorous quality standards.

• By enabling both automation and proper test management, we benefit from critical advantages not offered by manual testing.

• If in “agile” development model: Development cycles are short and tests are conducted at the same time as the coding.

• Things can get particularly complicated when the team is testing SOA applications and performance-scalability testing.
Why Testing is Very Crucial in Agile & Waterfall Development?

Waterfall Model

Agile Model

<table>
<thead>
<tr>
<th>Sprint 1</th>
<th>Sprint 2</th>
<th>Sprint 3</th>
<th>Sprint 4</th>
<th>Sprint 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>Analysis</td>
<td>Analysis</td>
<td>Analysis</td>
<td>Analysis</td>
</tr>
<tr>
<td>Coding</td>
<td>Coding</td>
<td>Coding</td>
<td>Coding</td>
<td>Coding</td>
</tr>
<tr>
<td>Testing</td>
<td>Testing</td>
<td>Testing</td>
<td>Testing</td>
<td>Testing</td>
</tr>
</tbody>
</table>
We Did Not Have Processes and Tools in Place

- The client had **Rational tools** (Req Pro, ClearQuest and Test Manager) but they were not properly used.

- And a lot of **Excel sheets and Word docs**. Never ending story of not being able to control requirements and defects ...

- We bought **Quality Center**, and we made Rational tools and Quality Center tools talk to one another. We faced challenges with Firewalls and security.

- Meanwhile, we started working on the **processes** with the client and our NG internal processes for test.

- We started putting together IMS, built partnership with the client.
We had to make Rational Test manager and Quality Center Test Plan talk to one another.
Automating Test Scripts

- Functional Testing
- Performance, Load, Stress Testing
- Service Test
- Security and SA Type of Testing

Important Features in Relation to Automation:
- Test Case, Requirement correlation
- Defects, Defect Management and its correlation to requirements and test cases if possible
- Test Scheduling IMS and test schedules and customer testing
Purpose of Automated Testing

- Checks virtually any functionality in application.
- Provides consistently re-usable tests to be run each time a site or application changes.
- Shortens testing time especially regression testing.
- Tracks all test runs, logs, test results in one, shared repository.

**Major benefits are:**

- Reusability
- Consistency
- Productivity
- Team work environment
Good Automation Candidates

- Tests that need to be run for every build, sometimes referred to as Sanity tests (Smoke and Regression tests).

- Tests that use multiple data values for the same actions are known as Data-Driven tests (Equals to, =>, <=).

- Identical tests that need to be executed using different browsers (We are using IE6,7 and FF).

- Mission critical pages (Certain pages need to be checked all the time).
Agile Testing Cycles

• **Sprints Testing:** For every sprint, test team will have a baseline. The baseline consists of tests created as a result of sprint requirements that will be fulfilled.

• **Smoke Testing:** For every sprint, we review the new test cases and adjust standard smoke tests to reflect any needed changes.

• **Regression Testing:** For every sprint, we review the new test cases, and based on the requirements and development, we complete a regression test. That full regression test includes all the sprint baseline regression and smoke tests. The regression test is fully automated with **QTP**.

• **Load Testing:** During sprints at Herndon, based on the needs, we develop LoadRunner scripts for performance and tuning. At the end of each iteration, our goal is to have a set of LoadRunner scripts that will allow us to see the performance, load and scalability for major business rules and transactions or identify bottlenecks...

• **Service Testing:** During sprints at Herndon, based on the needs, we develop services tests scripts. Our goal is to run these scripts under load as well as security.
In One Iteration: Smoke and Regression Tests

- Total Number of Test Cases
  - Smoke – 87 per browser
  - Regression – 259 per browser
  - Patch – approx 15 per browser

- Total Number of Releases
  - 18 so far, with additional releases to some rounds

- Days and Resources to Test
  - Smoke – 3.5 hours per browser (uninterrupted), usually 2 people
  - Regression – 16 hours per browser (uninterrupted), usually 2 people
  - Patch – 4 hours per browser (uninterrupted), usually 1 person

- Number of cycles before ATRR
  - Herndon
  - Client Suite A
  - Client Suite B

- Hours of Smoke, Patch and Regression
  - Herndon – 4 days per release
  - Client - Suite A – 3 days per release

For each release, we have ~1000 test cases to be executed, with 5 to 6 resources over 200 hours each.

This does not include downtime or any technical problems.

With the following assumptions:
- The testers are very familiar with the system.
- Requirements lists are already in the system.
- Defects descriptions and defects are already in the system.
Testing Documents

• Test Plan
  - Living document updated throughout iteration
  - User stories augment the test plan
  - Delivered at the end of each iteration

• Test Cases
  - Written throughout the Agile process
  - Input to Rational prior to end of each sprint
  - Automate QTP, LR and ST

• Test Results
  - At Herndon with Agile Teams
  - Delivered at the end of each sprint

• System Test Report
  - Living document updated throughout iteration
  - Updated at the end of each sprint
  - Delivered at the end of each iteration
Script Development

- Developed global scripts that can be called from one script to the other
- Scripts were grouped into test sets for different purposes. Such as quick regression test sets, quick check for critical areas or known issues
- With one script we were able to test the system with different browsers at the same time. E.g., IE6, IE&, FF3, etc.
- The same scripts were used for executing tests at different suits. So with one script we were able to run several tests depending on the situation we were in for that particular day.
Calling scripts from other test sets, excluding log in and log out, preparing test sets and making sure users have the right privileges to perform certain business rules.
Quick Test Professional integrates with Quality Center. Quality Center opens the selected testing tool automatically, runs the test, and exports the results to Quality Center. When you run a manual test (or automatically), emails are sent to whoever needs to be notified to inform the status of the test or when a defect is written.

For running tests at night, we have a schedule feature where we connect QTP via Quality Center. The test sets will be set ahead of time based on execution flow and tests will be executed based on the schedule specified date and time.
Questions a Performance Test Should Answer:

- Does the application respond quickly enough for the intended users?
- Will the application handle the expected user load and beyond?
- Will the application handle the number of transactions required by the business?
- Is the application stable under expected and unexpected user loads?
- Are you sure that users will have a positive experience on go-live day?
Performance Test Processes at NG Herndon

Performance & Load Test Process in the Application and at Northrop - Herndon

Plan/Design
- Define performance test scenarios
  - Business process
  - Existing performance issue
  - SLA’s
  - Baseline

Build
- Application & System Performance Requirements
  - app usage profile
  - user profile
  - system component resource usage profile
  - historic and projected traffic data
  - Max concurrent users
  - Peak hour throughput
  - Typical user session duration

Execute
- Scenario Run-time Settings
  - Number of VUs
  - Test duration/ Test Iteration
  - Define rendezvous points
  - Transaction definition
  - Test data (user accord test parameters, test data)
  - Stand alone scenario for individual report & capabilities
  - Combination scenario with different reports and capabilities

Analyze/Diagnose/Tune
- Application & System Tuning
  - Application tuning (optimize queries, reduce DB IO...)
  - DB tuning (index, optimize static analysis, SGA size, redo log buffer cache size...)
  - System component configuration optimization (connection pool setting, thread setting...)
  - System architecture optimization (Load balancing scheme, FW rule and capabilities...)
  - Network optimization

Test Result Analysis
- Obtain baseline
- Merge analysis graph & capture correlation patterns
- Identify bottleneck transaction
  - Response time
  - Hit per second
  - Transaction per second
  - Identify bottlenecks system component
  - CPU usage
  - Memory usage
  - Throughput trend
- Review LR and server logs

Test Monitoring
- SiteScope
  - CPU usage & processor queue
  - Memory usage (paging swapping...)
  - Server average load
  - Hit per second
  - Throughput
  - Transaction response time
  - Connections (total & new closed per second)
  - Web server (requests, connections...
  - App server (queue length, queue time)
  - DB server (lock wait, SGA size...)
  - Running SQL script (SQL trace, buffer cache hit ratio...)
  - Network delay monitor

Develop LR Scripts
- Define LR architecture
- Set up performance test env.
- Transaction definition
- Test data (user accord test parameters, test data)
- Stand alone scenario for individual report & capabilities
- Combination scenario with different reports and capabilities

Develop LR Scenarios
- Define LR architecture
- Set up performance test env.
- Transaction definition
- Test data (user accord test parameters, test data)
- Stand alone scenario for individual report & capabilities
- Combination scenario with different reports and capabilities

To Suite
While running performance test scripts one can see the actual response time and monitor the servers involved in the architecture.
LoadRunner Controller Monitors and SiteScope

- **SiteScope Monitors:**
  - CPU Utilization on Portal
  - Memory on Portal
  - CPU Utilization on SIDE
  - Memory on SIDE
  - CPU Utilization on WMS/WFS
  - Memory on WMS/WFS

- **LoadRunner Monitors:**
  - Hits per Second
  - Throughput
  - Transaction Response Time, etc.

-**Back End Data Verification**
Challenges of Testing for Agile and SOA

• **A more versatile test-bed environments**
  - It may be difficult to model the whole set of end-to-end software that probably span many different servers
  - Ability to simulate unavailable components

• **Transition for testers - process-centric testing team**
  - Broad knowledge of business processes
  - Understanding the intricacies of domino effects on business transactions
  - Cross-functional teams environment
  - Understand and diagnose underlying technology and connectivity

• **Location and identification of web services** (Geographic locations...)

• **Availability of web services components:** Applications, Middleware, Supporting hardware, teams - development, system admin, network, etc.

• **Locating and isolating defects are difficult:**
  - Defects in service components would cause domino effects to applications that utilize those services
  - Capture and analysis of all SOAP messages that are passed from one component to another is overwhelming
  - Service components do not have GUI
**Positive Testing** - Generates a full positive test for the selected services. It tests each operation of the selected service.

**Standard Compliance** - Tests the service’s compliance with industry standards such as WS-I and SOAP.

**Service Interoperability** - Tests the interoperability of the service’s operations with all supported Web Services toolkits.
- .NET Framework with WSE 2 SP3 – Tests the interoperability of the service’s operations using .NET Framework with WSE 2 SP3.
- Axis 1.3 Web Services Framework – Tests the interoperability of the service’s operations using Axis 1.3 Web Services Framework.
- Generic Mercury Solution – Tests the interoperability of the service’s operations using Generic Mercury Solution.

**Security Testing** - Tests the service's security.
- SQL Injection – Attempts to hack the service by injecting SQL statements that will result in an unauthorized extraction of data.
- Cross-Site Scripting – Attempts to hack the service by injecting code into a Web site that will disturb the functionality of the site.

**Boundary Testing** - Tests the service to its limits using the negative testing technique.
- Extreme Values – Tests for extreme values of simple data types.
Map Service Interface Description

The application has three Mapping Interfaces:

• Map Server
• Reporting Detail Request from Palanterra
• Reporting Mapping Call
Report Mapping Call is the single internal Mapping interface in the application.

When a user clicks the “Map” button within the results page of a report, a call is made to the Mapping application.

Diagram:
- **Mapping**
- **User**
- **Reports**
- **Mapping related servlets**
- **ESB**
- **Application**

- **GetCapabilities**
- **CapabilitiesInfo**
- **GetMap**
- **MapInfo**
With Service Test We are Able to:

• **Develop scripts:**
  - without a GUI
  - using multiple protocols. In enterprise world we have to deal with a lot of multiple protocols. This feature is very helpful.
  - by WSDL, UDDI, File and URL. This is a very helpful feature, too.

• **These scripts can be executed in LoadRunner for performance**

• **We can analyze traffic over the network**

• **We can set Security Policies that includes tokens, SAML, and so on**
Practice for Successful SOA Testing Strategy

Start early in the life cycle:

• Testing client applications – Start the end-to-end testing and tuning 6 months before the deployment.

Create an assembly-oriented test plan:

Test the components of the application one by one, conducting testing into stages with incremental increase in the number of components.

• Choose an appropriate set of test cases to support end-to-end testing of the business process and end-user experience.
With the use of the tools we were able to:

- Prioritize testing priorities based on business risk
- Access testing assets anytime, anywhere via a browser interface
- Create an end-to-end quality management infrastructure
- Manage manual and automated tests.
- Accelerate testing cycles by scheduling and running tests automatically, unattended, 24x7
- Manage multiple versions of requirements, tests, test scripts and business components
- Enforce standardized processes and best practices
- Analyze application readiness at any point in the testing process with integrated graphs and reports
With the use of the tools we were able to:

- 50 to 70% decrease in actual testing time (efficient and faster)
- Able to cope with huge amount of testing and captured defects at early stages of development
- Able to produce contractual documents such as RTM – Requirement Traceability Matrix, defect reports, test reports, test plans and the like in a timely manner.
- Able to produce metrics for defects such as defect containment, defect density, defect aging and other metrics related to defects
- Were able to capture changes that were made by third party and service providers and sub contractors in Service Test
- Able to capture security related vulnerabilities
- Able to capture the bottlenecks
- Had a chance to work on tuning and optimization of performance bottlenecks in architecture database and overall performance of the system
- Most importantly customer decided to have the same set up that we have at Herndon. Instead of sending testers to classified lab we are planning to send scripts only. That is the plan...
What We Need For Contractor Integration Results

Purpose: integrate and test all system components prior to official delivery to the government.

- All testing related documentation has been completed and is up to date
- Successful completion of smoke, patch, and regression tests
- Performance-Load-Stress test baselines obtained
- SLAs are met
- All the test results were delivered into government CM
- All defects are documented in CM tool
- Final system test report submitted to the PMO
- Installation and build guide with all the updates completed and delivered

Results from test indicated that the software is ready for Acceptance test
Requirements Validation

Results we would like to see:

- **Release Requirements testing**
  - Completed 99.13% testing of all testable requirements
    - Executed 99.63% testing against IE 6.0
    - Executed 98.50% testing against IE 7.0
    - Remaining 0.93% of testing could not be functionally tested

- **Regression testing**
  - Completed 100% of planned regression testing
    - Executed 95.05% testing against IE 6.0
    - Executed 92.95% testing against IE 7.0