LUXOFT
Expertise in Software Services

1st. Volokolamsky proezd, Bld.10
123060 Moscow • Russia
T +7 (495) 967-80-30 • F +7 (495) 967-80-32
www.luxoft.com • info@luxoft.com

Automated Testing Tools Research

65 pages

Date: 09.03.2007

Version: 1.0

Table of Contents

1. History

1.1 Revision History

Version	Date	Description of Changes	Reason	Made by
0.1	02.03.2007	Created	Customer request for research	E. Gomonova
0.2	02.03.2007	"Customer Needs" and "Introduction" sections were drafted. Descriptions of some products were drafted.		I. Anisimov I. Petrov D. Kondratiev
0.4	06.03.2007	All sections except "Summary" were drafted		I. Anisimov I. Petrov D. Kondratiev D.Zernov
0.6	09.03.2007	Document is ready for review		O.Moroz I. Anisimov I. Petrov D. Kondratiev D.Zernov
1.0	09.03.2007	Proposed		

1.2 Review History

Version	Date	Reviewer	Reference
0.3	02.03.2007	A. Ignatov	
0.5	06.03.2007	A. Ignatov	
0.7	09.03.2007	A. Ignatov	

1.3 Approval History

Version	Date	Approved by	Signature or reference
0.8	09.03.2007	E.Povalyaev	

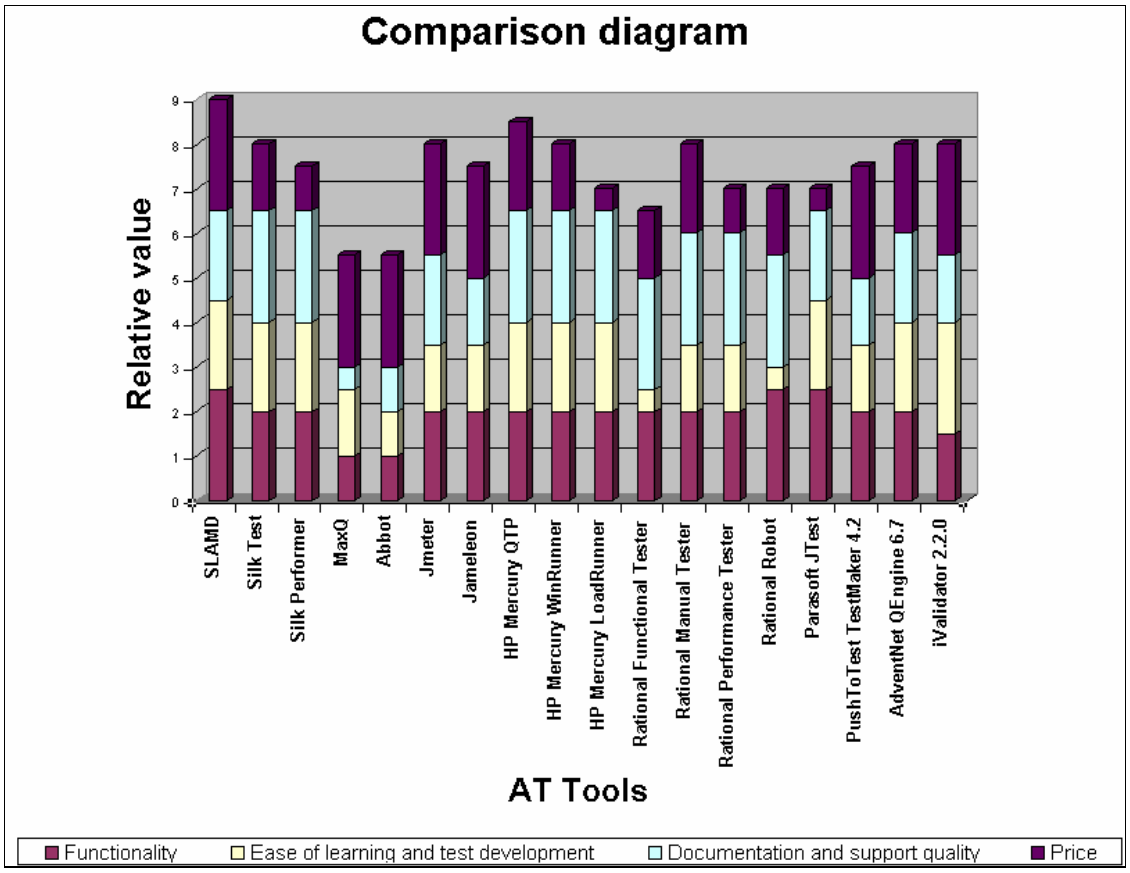
2. Introduction

2.1 Purpose

The main purpose of this document is to present the results of the research established by Luxoft to compare several Automated Testing tools and find out the most appropriate solution, which is eminently suitable to business and technical requirements ofCustomer.

2.2 Summary

We have evaluated and compared several commercial and free open source tools for automated testing (See the diagram below):



In our research we evaluated tools considering two major categories:

- Performance / load / stress testing tools
- Regression / Functional / Unit testing tools.

We also separated the results between commercial and free tools.

Performance / Load / Stress Testing Tools Results

Among commercial tools Mercury LoadRunner® is a leader. LoadRunner is the industry-standard load testing solution is the best tool to detect bottlenecks before a new system or upgrade is deployed. LoadRunner achieves this task with support of a wide range of protocols and thousands of virtual users executing different business processes to emulate the production conditions a deployed application will face. Supported protocols include Web, J2EE, .NET, XML, ERP/CRM, wireless, Citrix, and as well as client-server applications. Through **SilkPerformer** is also on a leading position, it has less functions (less supported protocols and environments) than Load Runner. The only disadvantage of Load Runner is high price.

The best free open source tool is SLAMD from Sun Microsystems. SLAMD is an open source Java-based application designed for stress testing and performance analysis of network-based applications. SLAMD provides an easy way to schedule a job distribute that job information to a number of client systems, and then executed concurrently on those clients to generate higher levels of load and more realistic usage patterns, monitoring resource usage in job progress. SLAMD also facilitates processing of test data with report generation and export of job data.

INTERESTING FINDS

Apache Jakarta Jmeter comes next after SLAMD. When comparing with Apache, SLAMD provides a much better support for distributed testing as well as a powerful scripting language to automate test case development.

Regression / Functional / Unit Testing Tools Results

Among commercial tools Mercury QuickTest Professional™ is an absolute winner. QuickTest is a next-generation automated testing solution that deploys the concept of Keyword-driven testing to radically simplify test creation and maintenance.

QTP's IDE provides user with some very advanced features, related to managing test assets (Objects Repository, Data pool, Checkpoints, Test script as a tree). QuickTest Professional supports functional testing of all popular environments, including Windows, Web, .Net, Java, ERP/CRM applications (SAP, Siebel), Oracle, etc.

The best free open source tool is iValidator. According to our information iValidator is the most sophisticated free unit testing tool. It provides complex test scenarios and extensive test control which includes mechanisms for running hierarchic and sequential test concurrently control their behavior depending on errors in individual tests or test suites. Unit Tests in iValidatore are reusable and flexible with parameters supplied through the XML description.

The test results are reported in xml. The structure of the report reflects exactly the hierarchic structure of the test configuration and thus provides a clear presentation even for complex, deeply layered scenarios.

The iValidator test framework is by no means limited to the test automation of Java based systems. By the use of so-called adapters any system can be tested. The test execution and the display of the results can be integrated seamlessly into Eclipse with iValidator providing Eclipse plug-In and support of JUnit.

INTERESTING FINDS

Also of interest in this category is IBM Rational Manual Tester which provides unique manual test authoring and execution tool. Promoting test step reuse and reduce the impact of software change on testers and business analysts.

2.3 Scope

The scope of this analysis is limited by the explicit list of <*product type name*> for evaluation, including the following products:

- Mercury testing tools
- Rational testing tools
- Segue (Borland) testing tools
- Parasoft JTest 8.0
- Push-to-test TestMaker 6.2
- AdventNet QEngine 6.7
- SLAMD 2.0.0-alpha1
- JMeter 2.2
- iValidator 2.2.0
- Jameleon

Solutions are evaluated basing on the list of requirement and criteria specified by the customer and listed in sections "Customer Needs" and "Estimation Criteria" correspondingly.

3. Customer Needs

Automated Testing - is automating the manual testing process currently in use. This requires that a formalized "manual testing process" currently exists in the company or organization. Minimally, such a process includes:

- Detailed test cases, including predictable "expected results", which have been developed from Business Functional Specifications and Design documentation
- A standalone Test Environment, including a Test Database that is restorable to a known constant, such that the test cases are able to be repeated each time there are modifications made to the application

The real use and purpose of automated test tools is to automate regression testing. To perform it there should be a database of detailed test cases that are repeatable, and this suite of tests must be run every time there is a change to the application to ensure that the change does not produce unintended consequences.

An **automated test script** is a program. Automated script development, to be effective, must be subject to the same rules and standards that are applied to software development.

According to Customer request, the needed automated testing tool should provide ability to perform the following types of testing:

- Regression Testing (for Web interfaces) - not just navigation and links, expected and actual results
- Load/Stress/Volume Testing
- Unit/Integration Testing

There are also some additional features of the automated testing tool that should be taken into consideration:

- Cost
- Inter-operability – ability to work together with other tools
- Capacity - particularly on load testing, how many users at once, virtual clients, multiple physical client spanning etc

4. Estimation Criteria

During tools evaluation comparison parameters were split into three major groups:

1) General Test Automation Comparison Parameters
2) Regression / Unit / Integration Testing Parameters
3) Load/Stress Testing Parameters

Parameters are described in the following sections.

4.1 General Test Automation Comparison Parameters:

4.1.1 Test Design tools for both Automated and Manual testing.

Automated testing does not replace the need for manual testing or allow to "down-size" testing department. Automated testing is an addition to a testing process. This testing process needs Test Methodology and Test Design tools to be used in manual and automated testing.

4.1.2 Methods used for test creation and test specification

Methods used for test creation and test specification are as follows:

- **The "Key-Word Driven" (or "Test Plan Driven") Method**. This method uses the actual Test Case document developed by the tester using a spreadsheet containing special "Key-Words". In this method, the entire process is data-driven, including functionality. The Key Words control the processing. Script development methodology in this case is based on "Functional Decomposition" methodology used to reduce all test cases to their most fundamental tasks implemented with scripts which perform these tasks independently of one another.
- **Record / Playback Method**. With this method non-technical user-type testers record their actions, and then play back the recorded scripts. The Record/Playback method has serious shortcomings when used in a real-world scenario:
  - The scripts resulting from this method contain hard-coded values which must change if anything at all changes in the application.
  - The costs associated with maintaining such scripts are astronomical, and unacceptable.
  - These scripts are not reliable, even if the application has not changed, and often fail on replay (pop-up windows, messages, and other things can happen that did not happen when the test was recorded).

- If the tester makes an error entering data, etc., the test must be re-recorded.
- If the application changes, the test must be re-recorded.
- All that is being tested are things that already work. Areas that have errors are encountered in the recording process (which is manual testing, after all). These bugs are reported, but a script cannot be recorded until the software is corrected. So what are you testing?

4.1.3 Functional Testing.

Testing in this case takes an external perspective of the test object to derive test cases. The test designer selects valid and invalid input and determines the correct output. There is no knowledge of the test object's internal structure. This method of test design is applicable to all levels of software testing: unit, integration, functional testing, system and acceptance. The higher the level, and hence the bigger and more complex the box, the more we're forced to use black box testing to simplify (for example, High-level Test Flows that mirror actual business process). While this method can uncover unimplemented parts of the specification, you can't be sure that all existent paths are tested.

4.1.4 Scripting.

Interpreted script languages are used by testing frameworks to automate creation and deployment of test cases.

- **Script language.** Being interpreted test scripts do not require recompilation and are easy to modify and adapt test scripts for different test cases.
- **Support for BSF (Bean Scripting Framework) or other extensions**. Java BSF allows test developers to choose a BSF-compatible scripting language that they prefer and feel comfortable to write tests.
- **Accesses to Test Repository.** Test scripts can access Test Repository directly to store test data or retrieve complete test context and test history.
- **Statistics and reporting functions.** Script statistics trackers can be used to measure the number of times a given event occurs during each interval (e.g., the number of SQL requests completed per second). Any statistics collected by script trackers will automatically be reported back to the framework when the test job is complete.

4.1.5 Test Documentation.

Test template may include documentation of the test case that test implements. Documentation improves maintainability of test cases in the system.

4.1.6 Test Plan Support.

Test Plan consists from sets of test cases together with schedule that determines times when tests run.

Parameterization and Reuse. Test Plan may have parameters that allow customizing its use. Parameters my be grouped by test cases as well as parameters relating to schedule of test runs.

4.1.7 Test Scheduling

Test Scheduling includes the following options:

- **Automated periodic runs**. Test Cases run periodically according with an associated schedule.
- **Every time application under test changes**. Regression and integration tests can run automatically when new build of the system is done.

4.1.8 Interface.

Test framework can provide the following forms of administration interface:

- WEB-based
- Command-line

4.1.9 Support of Test Repository.

Automated Testing requires a formalized "manual testing process" that includes Test Repository. Test Repository is built around test database that is restorable to a known constant, such that the test cases are able to be repeated each time there are modifications made to the application Types of objects stored in Test Repository.

- Integration with application build process (Test Repository update).
- Support of Version Control Systems

4.1.10 Supported protocols for remote testing.

Many complex systems can be tested only through remote interfaces. These interfaces can be accessed using protocols, such as:

- TCP/IP, UDP
- SSL
- HTTP/HTTPS
- SMTP, POP, IMAP
- LDAP
- etc

4.1.11 Supported Testing Environments.

Some frameworks are designed specifically to test specific systems through tight integration with run-time environments of these systems, such as: Java/J2EE, Web Services, AJAX, .NET, Web (Internet Explorer, Firefox, Netscape), WebLogic, JBoss Web Server Monitoring, SAP, Siebel, Oracle, PeopleSoft, Visual Basic, ActiveX, Windows applications, mainframe terminal emulators, and Macromedia Flex.

4.1.12 Statistics and Reporting Tools.

These tools collect statistics and generate reports of different kinds. The most commonly used report types are comparison reports and reports with graphical charts. Reporting tools can export test data to various data formats (text files: tab or comma delimited, XML files)

4.1.13 Integration with external systems

This parameter describes the ability of the tool to support QA tools such as Mercury Quality Management and CRM/ERP Systems.

4.1.14 Team work

Team work includes support of source code control systems such as CVS, Subversion, ClearCase, and StarTeam. To enchance collaboration among distributed team members, testing tool may also provide monitoring of commits and check-ins from defined team members.

4.1.15 Learning Effort s

This parameter describes the user effort that are required to learn how to use the tool.

4.1.16 Price

This parameter contains information about the price of a tool.

4.2 Regression / Unit / Integration Testing Parameters

The Regression / Unit / Integration testing parameters are as follows:

- Support of complex test scenarios.
  - Hierarchy of test suits and test parameters
  - Concurrent execution of hierarchic and sequential test structures.
  - Conditional test behavior depending on errors in individual tests or test suites.
  - Access to complete test context and test history.
- Reusability and parameterization
  - Flexible parameterization of test setups, checkups, and teardowns.
  - Parameters in XML
- Supported test levels
  - Developer tests
  - Integration tests
  - Acceptance tests
- JUnit Support
- IDE Plug-In (Eclipse)

4.3 Load/Stress Testing Parameters

The Load/Stress testing parameters are as follows:

- Flexible job scheduler
- Statistics
  - Trackers
  - In-progress results
- Distributed testing
  - Centralized control of Test Jobs
  - Multiple concurrent clients
  - Self-optimizing jobs
  - Security and access control
- System Resource monitoring (system meters)
- Extensibility

5. Automated Testing Tools Comparison

5.1 Mercury

5.1.1 Mercury LoadRunner®

http://www.mercury.com/us/products/performance-center/loadrunner/

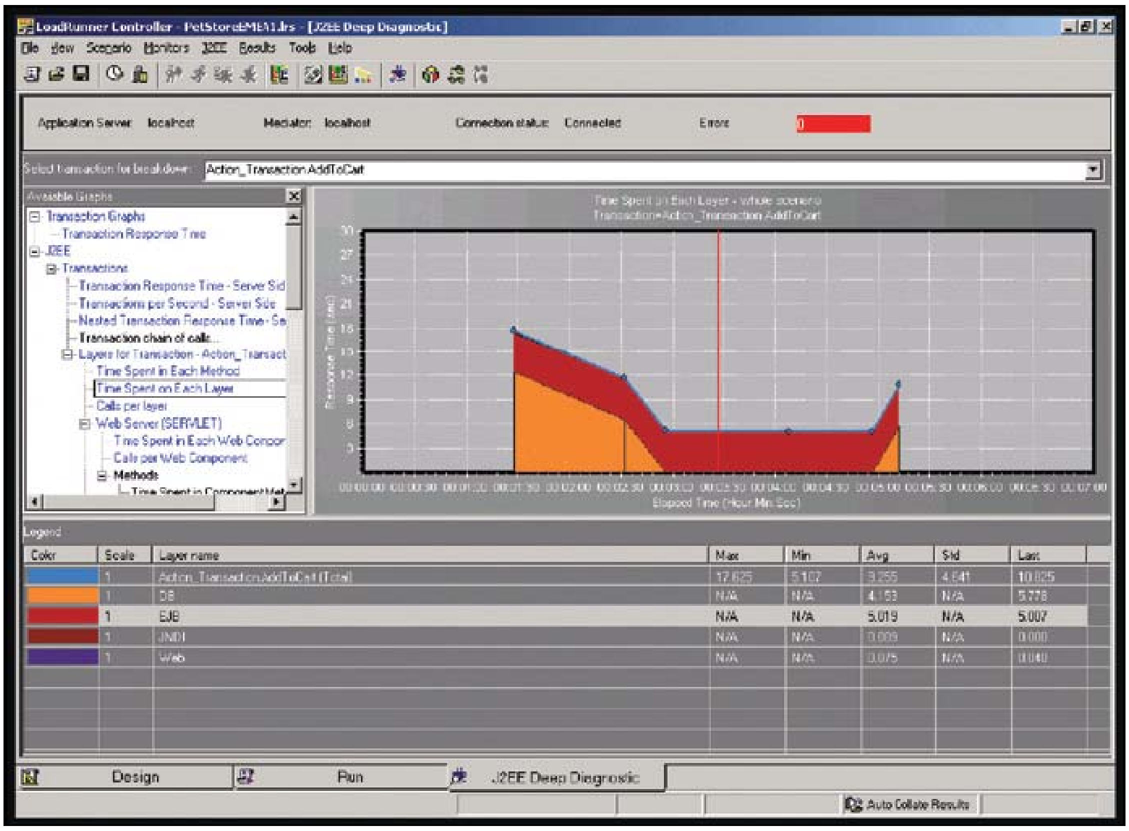
Mercury LoadRunner® is the industry-standard load testing solution for predicting system behavior and performance, and the only integrated load testing, tuning, and diagnostics solution in the market today. It prevents costly performance problems in production by detecting bottlenecks before a new system or upgrade is deployed. You can verify that new or upgraded applications will deliver intended business outcomes before go-live, preventing over-spending on hardware and infrastructure. With LoadRunner web testing software, you can measure end-to-end performance, diagnose application and system bottlenecks, and tune for better performance – all from a single point of control. It supports a wide range of enterprise environments, including Web Services, J2EE, and .NET.

LoadRunner allows you to perform the following actions:

- Obtain an accurate picture of end-to-end system performance
- Verify that new or upgraded applications meet specified performance requirements
- Identify and eliminate performance bottlenecks during the development lifecycle.

Functionality	Good
Ease of learning and test development	Good
Documentation and support quality	Excellent
Price	Poor
Total Evaluation	★★★★

Real-time performance monitors obtain and display performance data from every tier, server, and system component, and diagnostics probes gather code-level data to isolate bottlenecks. This combination of end-user, system-, and codelevel visibility dramatically reduces time to problem resolution.


LoadRunner Controller - PetStoreEMEA1.lrs - [J2EE Deep Diagnostic]
File View Scenario Monitors J2EE Results Tools Help
Available Graphs
Transaction Graphs
Transaction Response Time
J2EE
Transactions
Transaction Response Time - Server Sid
Transactions per Second - Server Side
Nested Transaction Response Time - Se
Transaction chain of calls
Layers for Transaction - Action_Transact
Time Spent in Each Method
Time Spent on Each Layer
Calls per layer
Web Server (SERVLET)
Time Spent in Each Web Compon
Calls per Web Component
Methods
Time Spent in Each Layer - whole scenario
Transaction=Action_Transaction AddToCart
Response Time (sec)
Elapsed Time (Hour:Min:Sec)
Design
Run
J2EE Deep Diagnostic
Auto Collate Results

Application Server: localhost   Mediator: localhost   Connection status: Connected   Errors: 0

Select transaction for breakdown: Action_Transaction AddToCart

Legend

Color	Scale	Layer name	Max	Min	Avg	Std	Last
🟦	1	Action_Transaction.AddToCart (Total)	17.625	5.107	9.255	4.641	10.825
🟧	1	DB	N/A	N/A	4.153	N/A	5.778
🟥	1	EJB	N/A	N/A	5.019	N/A	5.007
🟥	1	JNDI	N/A	N/A	0.009	N/A	0.000
🟪	1	Web	N/A	N/A	0.075	N/A	0.040

Mercury LoadRunner Diagnostics pinpoints code bottlenecks.

Mercury LoadRunner supports performance testing for the widest range of wireless and enterprise environments. It can test WAP, i-Mode, Multimedia Messaging (MMS), XHTML, HTML, WBXML, web, Web Services, Client-Server, Legacy, Citrix, Java, .NET, and all ERP/CRM applications. Mercury LoadRunner has more than 40 non-intrusive monitors tailored for these systems and provides diagnostics for J2EE, .NET, Siebel, Oracle, and SAP. Mercury LoadRunner is one solution for your entire range of wireless and enterprise load testing needs.

Features and Benefits

- **On-Demand Production Workloads**

  Feature: LoadRunner is able to drive hundreds and thousands of virtual users executing different business processes to emulate the production conditions a deployed application will face.

  Benefit: Ability to uncover performance and scalability bottlenecks that would otherwise surface in production before going live, minimize production downtime and poor performance, and meet Service Level and Uptime requirements.

- **Enterprise Environment Support**

  Feature: LoadRunner supports almost 60 protocols– more than any other load-testing solution. This includes Web, J2EE, .NET, XML, ERP/CRM, wireless, Citrix, and client-server applications.

  Benefit: You can use the same tool for performance testing – even as the types of applications being deployed change from client-server to web to Java. One tool, one set of employee skills – even if applications change over time.

- **Enterprise Monitoring Support**

  Feature: LoadRunner's non-intrusive, real-time performance monitors provide detailed metrics on all parts of the system under test. This includes web servers, application servers, databases, ERP and CRM systems, firewalls, load balancers, etc.

  Benefit: LoadRunner allows you to identify hardware limitations and software configuration issues that might otherwise go undetected.

- **Enterprise Diagnostics (J2EE, .NET, Siebel, Oracle Applications, SAP R/3)**

  Feature: LoadRunner is the only performance testing solution that can trace, time, and troubleshoot individual application components under load. Users can drill-down from a slow end-user transaction to the bottlenecked method or SQL statement that is causing the slowdown.

  Benefit: This granularity of results data ensures that every load test provides development with actionable results, thus reducing the cost and time required to optimize J2EE and Siebel deployments.

5.1.2 Mercury QuickTest Professional

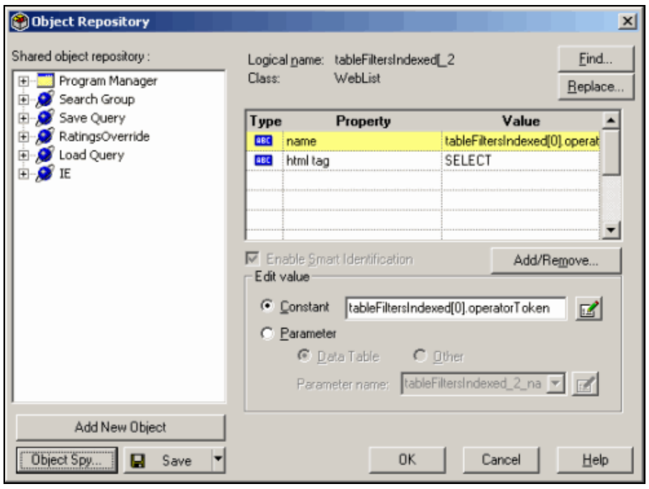
http://www.mercury.com/us/products/quality-center/functional-testing/quicktest-professional/

Mercury QuickTest Professional™ provides the industry's best solution for functional test and regression test automation - addressing every major software application and environment. This next-generation automated testing solution deploys the concept of Keyword-driven testing to radically simplify test creation and maintenance. Unique to QuickTest Professional's Keyword-driven approach, test automation experts have full access to the underlying test and object properties, via an integrated scripting and debugging environment.

Functionality	Good
Ease of learning and test development	Good
Documentation and support quality	Excellent
Price	Good
Total Evaluation	★★★★★

Object Repositories management in QTP is rather limited, because of their complex structure. User cannot copy a "child" of one, say, frame to another, cannot (easily) copy an object from one OR to another, cannot manually insert new object (without launching AUT),etc.

The **QTP Object Repository** has the following view



Logical name: tableFiltersIndexed[_2
Class: WebList

Type	Property	Value
ABC	name	tableFiltersIndexed[0].operat
ABC	html tag	SELECT

[✓] Enable Smart Identification

Edit value

[● Constant] tableFiltersIndexed[0].operatorToken

○ Parameter

[● Data Table] ○ Other

Parameter name: tableFiltersIndexed_2_na

QuickTest Professional has the following features:

- **Object Repository Manager**: Enables collaboration within tester workgroups by keeping application object data in sync. Also it provides ability to merge, import/export to XML files, and add objects from application screens or meta-data.

- **Robust Function Libraries**: Enables sharing of function libraries within tester workgroups.

- **Intelligent Pre-Run Validation**: Runs a pre-execution resource check automatically, notifying users of missing files or resources.

- **Enhanced Keyword View**: Drag-and-drop test steps within the Keyword View's natural language environment.

- **Open XML Report Format for Test Results**: Stores test results in an open XML format, enabling you to easily customize the reports according to your own requirements, and to integrate the test result information with other applications. Test results can now be exported to HTML.

- **New IDE Environment**: Offers a highly customizable test development environment.

- **New Debugger**: Enables testers to pin-point test errors when building and maintaining test cases.

- **Keyword Management**: Manage keywords, including turning on/off specific methods from the Keyword View.

- **New Application Area Management**: Leverages Mercury Business Process Testing so application area definitions can now include multiple object repositories.

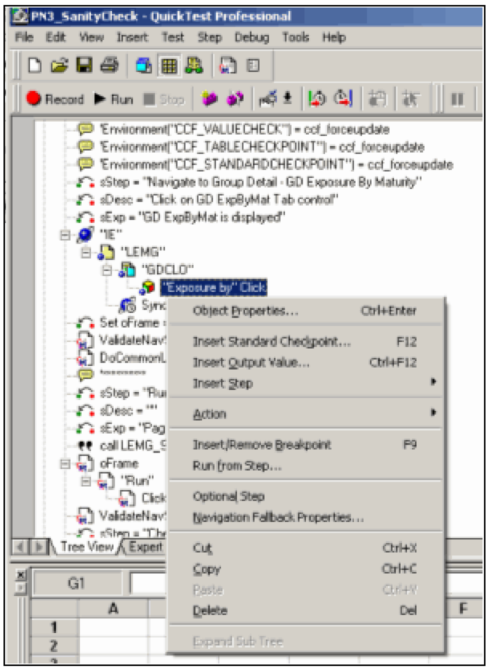
- **Multiple Document Interface for Function Libraries, Object Repositories**: Allows users to concurrently open and edit multiple function libraries and Object Repositories within the QuickTest Professional interface.

- **Unicode Support**: Lets you test global, multi-language deployments of your enterprise applications.

QuickTest Professional supports functional testing of all popular environments, including Windows, Web, .Net, Visual Basic, ActiveX, Java, SAP, Siebel, Oracle, PeopleSoft, and terminal emulators. New Environment Support: Supports Web services, .NET 2.0, Firefox 1.5, Netscape 8, Macromedia Flex 2, Win XP 64 bit, Internet Explorer 7, and the latest ERP/CRM applications.

Thus, QTP's IDE lacks some very common (for, even, most simple code editors) features, but at the same time it provides user with some very advanced features, related to managing test assets (Objects Repository, Data pool, Checkpoints, Test script as a tree).

The **QTP Tree** has the following view:


PN3_SanityCheck - QuickTest Professional
File Edit View Insert Test Step Debug Tools Help
Record Run Stop
Tree View Expert
G1
A
F
1
2
Object Properties... Ctrl+Enter
Insert Standard Checkpoint... F12
Insert Output Value... Ctrl+F12
Insert Step
Action
Insert/Remove Breakpoint F9
Run from Step...
Optional Step
Navigation Fallback Properties...
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Expand Sub Tree

```
'Environment("CCF_VALUECHECK") = ccf_forceupdate
'Environment("CCF_TABLECHECKPOINT") = ccf_forceupdate
'Environment("CCF_STANDARDCHECKPOINT") = ccf_forceupdate
sStep = "Navigate to Group Detail - GD Exposure By Maturity"
sDesc = "Click on GD ExpByMat Tab control"
sExp = "GD ExpByMat is displayed"
"IE"
  "LEMG"
    "GDCLO"
      "Exposure by" Click
    Sync
Set oFrame
ValidateNav
DoCommonL
'*********
sStep = "Ru
sDesc = ""
sExp = "Pag
call LEMG_S
oFrame
  "Run"
    Click
ValidateNav
sStep = "The
```

Mercury QuickTest Professional offers a fresh approach to automated testing that deploys the concept of Keyword-Driven testing to radically simplify test creation and maintenance. Unique to Mercury QuickTest Professional's Keyword-Driven approach, testers can select to build test cases by capturing flows directly from the application screens using robust capturing technology (record/replay). In addition, power users will find full access to the underlying test and object properties through an integrated scripting and debugging environment that is round-trip synchronized with the Keyword View.

Usage of VBScript brings QTP full compatibility with Microsoft component technologies (COM, ActiveX, OLE Automation, etc.). QTP has OLE Automation interface, that allows user to write batch scripts (on VBScript), that can be run from command line.

From the other hand, the choice of VBScript is not very good, because VBScript has some significant limitations: for instance, it does not support modules/libraries and cannot be considered as "truly" object-oriented language: it does not support inheritance of classes.

5.1.3 Mercury WinRunner®

http://www.mercury.com/us/products/quality-center/functional-testing/winrunner/

Mercury WinRunner® offers a powerful tool for enterprisewide functional and regression testing. Mercury WinRunner captures, verifies, and replays user interactions automatically, so you can identify defects and ensure that business processes work flawlessly upon deployment and remain reliable.

Functionality	Good
Ease of learning and test development	Good
Documentation and support quality	Excellent
Price	Satisfactory
Total Evaluation	★★★★

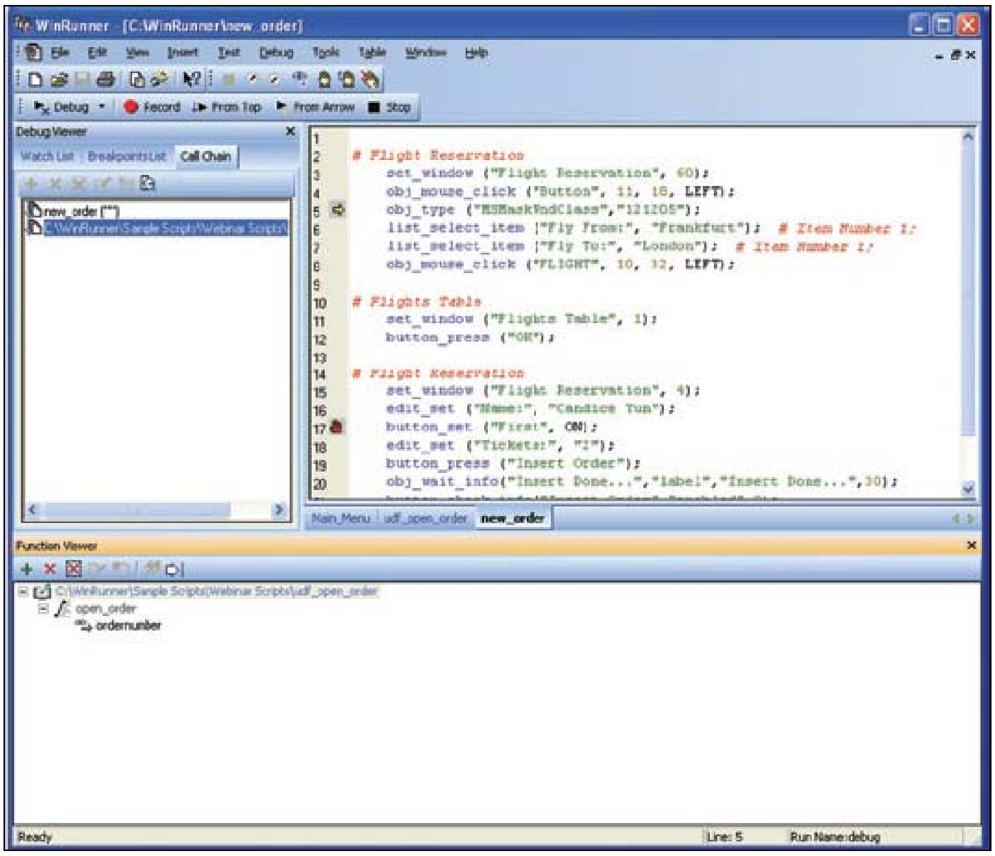
Mercury WinRunner has several advantages, including:

- Reduced testing time by automating repetitive tasks.
- Optimized testing efforts by covering diverse environments with a single testing tool.
- Maximized return on investment through modifying and reusing test scripts as the application evolves.

Mercury WinRunner's intuitive recording process allows users to produce robust functional tests. To create a test, WinRunner simply records a typical business process by emulating user actions, such as ordering an item or opening a vendor account.

Mercury WinRunner's workspace enables you to easily manage and monitor breakpoints, watch points,and call chains. During recording, you can directly edit generated scripts to meet the most complex test requirements

The example of WinRunner's workspace is represented on the following picture.


WinRunner - [C:\WinRunner\new_order]
File Edit View Insert Test Debug Tools Table Window Help
Debug Record From Top From Arrow Stop
Debug Viewer
Watch List BreakpointsList Call Chain
new_order ("")
C:\WinRunner\Sample Scripts\Webinar Scripts\
Main_Menu udf_open_order new_order
Function Viewer
C:\WinRunner\Sample Scripts\Webinar Scripts\udf_open_order
open_order
ordernumber
Ready Line: 5 Run Name: debug

```

# Flight Reservation
    set_window ("Flight Reservation", 60);
    obj_mouse_click ("Button", 11, 10, LEFT);
    obj_type ("MSMaskWndClass","121205");
    list_select_item ("Fly From:", "Frankfurt");  # Item Number 1;
    list_select_item ("Fly To:", "London");  # Item Number 1;
    obj_mouse_click ("FLIGHT", 10, 32, LEFT);

# Flights Table
    set_window ("Flights Table", 1);
    button_press ("OK");

# Flight Reservation
    set_window ("Flight Reservation", 4);
    edit_set ("Name:", "Candice Tun");
    button_set ("First", ON);
    edit_set ("Tickets:", "2");
    button_press ("Insert Order");
    obj_wait_info("Insert Done...","label","Insert Done...",30);
```

Features and Benefits

The main features of Mercury WinRunner® are as follows:

- **Increasing power and flexibility of tests without any programming**: The Function Generator presents a quick and error-free way to design tests and enhance scripts without any programming knowledge. Testers can simply point at a GUI object, and Mercury WinRunner® will examine it, determine its class and suggest an appropriate function to be used.

- **Using multiple verification types to ensure sound functionality**: Mercury WinRunner provides checkpoints for text, GUI, bitmaps, URL links and the database, allowing testers to compare expected and actual outcomes and identify potential problems with numerous GUI objects and their functionality.

- **Verifying data integrity in your back-end database**: Built-in Database Verification confirms values stored in the database and ensures transaction accuracy, as well as the data integrity of records that have been updated, deleted, and added.

- **Viewing, storing, and verifying at a glance every attribute of tested objects**: Mercury WinRunner's GUI Spy automatically identifies, records and displays the properties of standard GUI objects, ActiveX controls, as well as Java objects and methods. This ensures that every object in the user interface is recognized by the script and can be tested.

- **Maintaining tests and build reusable scripts**: The GUI map provides a centralized object repository, allowing testers to verify and modify any tested object. These changes are then automatically propagated to all appropriate scripts, eliminating the need to build new scripts each time the application is modified.

- **Testing multiple environments with a single application**: Mercury WinRunner supports more than 30 environments, including Web, Java, Visual Basic, etc. In addition, it provides targeted solutions for such leading ERP/CRM applications as SAP, Siebel, PeopleSoft, and a number of others.

- **Simplifying creation of test scripts**: Mercury WinRunner's DataDriver Wizard greatly simplifies the process of preparing test data and scripts. This allows for optimal use of QA resources and results in more thorough testing.

- **Automatically identifying discrepancies in data**: Mercury WinRunner examines and compares expected and actual results using multiple verifications for text, GUI, bitmaps, URLs, and databases. This ensures stable functionality and execution of business transactions when the application is released into production.

- **Validating applications across browsers**: Mercury WinRunner enables you to use the same test to validate applications in Internet Explorer, Netscape, and AOL. This saves testing time and reduces the number of scripts that must be developed and maintained.

- **Automatically recovering tested applications from a crash**: Unexpected events, errors, and application crashes during a test run can disrupt the testing process and distort results. Mercury WinRunner's Recovery Manager enables unattended recovery and provides a wizard that guides the process of defining a recovery scenario.

- **Leveraging investments in other testing products**: Mercury WinRunner fully integrates with our other testing solutions, including Mercury LoadRunner® for load testing and Mercury TestDirector™ for global test management. Moreover, organizations can reuse Mercury WinRunner test scripts with Mercury QuickTest Professional™.

- **Fully integrating with Mercury Business Process Testing**: With Mercury WinRunner 8.2's compatibility with Mercury Business Process Testing™ , you have the ability to create business process components as well as convert existing Mercury WinRunner scripts to components. With Mercury Business Process Testing, subject matter experts and automation engineers collaborate to increase effectiveness.

5.2 Rational

5.2.1 Rational Functional Tester

http://www-306.ibm.com/software/awdtools/tester/functional/features/index.html

IBM Rational Functional Tester is an advanced, automated functional and regression testing tool for testers and GUI developers who need superior control for testing Java, Microsoft® Visual Studio .NET and Web-based applications.

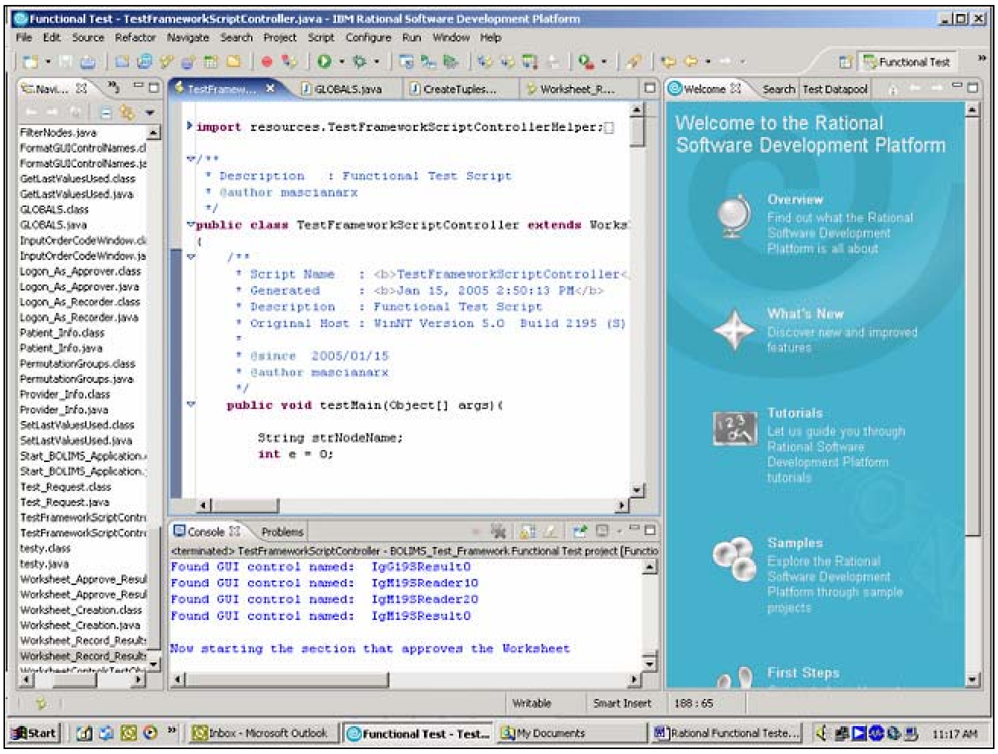
Rational Functional Tester records user interactions with Java, Web, and Visual Studio .NET WinForm-based applications, creating a test script that - when executed - reproduces those actions. During recording, the user can insert verification points that extract specified data or properties from the application under test. During playback, these verification points are used to compare recorded information with live information to ensure consistency.

Functionality	Good
Ease of learning and test development	Poor
Documentation and support quality	Excellent
Price	Satisfactory
Total Evaluation	★★★★

Following any test recording activity, testers have the option of adding custom code to the test script to perform an unlimited array of tasks, including the data manipulation and environment configuration activities that are often necessary to ensure the test lab is properly constituted for the test run.

Following test execution, Rational Functional Tester generates a report listing the results of the verification point comparisons. With Rational Functional Tester, teams are able to more reliably and efficiently expose problems in even the most complex applications, increasing the opportunity for defect capture and repair before product release.

The example of Rational Functional Tester Version 6.1 (Functional Test Perspective) is provided on the picture below:



```
import resources.TestFrameworkScriptControllerHelper;

/**
 * Description   : Functional Test Script
 * @author mascianarx
 */
public class TestFrameworkScriptController extends Works
{
    /**
     * Script Name   : <b>TestFrameworkScriptController<
     * Generated     : <b>Jan 15, 2005 2:50:13 PM</b>
     * Description   : Functional Test Script
     * Original Host : WinNT Version 5.0  Build 2195 (S)
     *
     * @since  2005/01/15
     * @author mascianarx
     */
    public void testMain(Object[] args){

        String strNodeName;
        int e = 0;
```

```
<terminated> TestFrameworkScriptController - BOLIMS_Test_Framework Functional Test project [Functio
Found GUI control named:  IgG19SResult0
Found GUI control named:  IgM19SReader10
Found GUI control named:  IgM19SReader20
Found GUI control named:  IgM19SResult0

Now starting the section that approves the Worksheet
```

IBM Rational Functional Tester Version 6.1 (Functional Test Perspective)

Features and benefits of IBM Rational Functional Tester are as follows:

- Support for testing of Java, Web and Visual Studio .NET WinForm-based applications
- Ability to choose language - Java or Visual Basic .NET - for test script customization
- Native Java and Visual Basic .NET editor and debugger for advanced testers
- ScriptAssure technology to accommodate frequent UI modifications
- Automated data correlation and data-driven testing eliminate need for manual coding
- Multiple verification points with regular expression pattern matching support
- Advanced object map maintenance capabilities
- Linux test editing and test execution support
- Ships with IBM Rational Manual Tester
- Ships with IBM Rational ClearCase LT for automated version control
- Add-on support available for testing 3270/5250 terminal-based applications
- Extended automated functional and regression testing for Siebel® and SAP® applications.

Rational Functional Tester includes the following components:

- IBM Rational Manual Tester
- IBM Rational TestManager

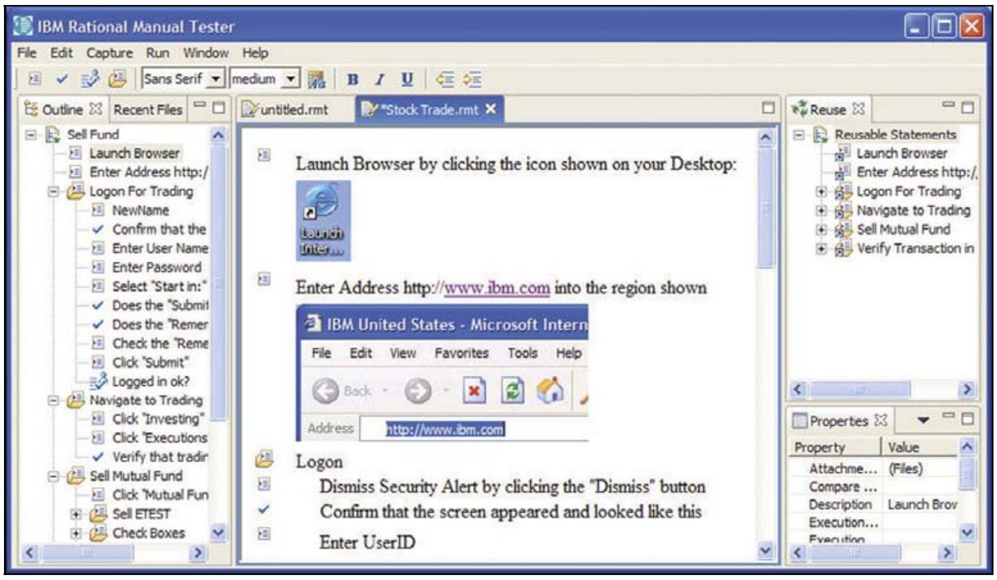
5.2.2 Rational Manual Tester

http://www-306.ibm.com/software/awdtools/tester/manual/index.html

IBM Rational Manual Tester is a manual test authoring and execution tool that promotes test step reuse to reduce the impact of software change on testers and business analysts. Rational Manual Tester adds organization and control to all activities that comprise a manual testing effort, including:

Functionality	Good
Ease of learning and test development	Satisfactory
Documentation and support quality	Excellent
Price	Good
Total Evaluation	★★★★

- Test creation and modification
- Test organization and consolidation for distributed team members
- Test execution and results collection
- Test results reporting



Sell Fund
Launch Browser
Enter Address http:/
Logon For Trading
NewName
Confirm that the
Enter User Name
Enter Password
Select "Start in:"
Does the "Submit
Does the "Remer
Check the "Reme
Click "Submit"
Logged in ok?
Navigate to Trading
Click "Investing"
Click "Executions
Verify that tradir
Sell Mutual Fund
Click "Mutual Fun
Sell ETEST
Check Boxes

Launch Browser by clicking the icon shown on your Desktop:

Enter Address http://www.ibm.com into the region shown

Logon

Dismiss Security Alert by clicking the "Dismiss" button

Confirm that the screen appeared and looked like this

Enter UserID

IBM Rational Manual Tester

Features and benefits of IBM Rational Manual Tester are as follows:

- Test step reuse palette to enable sharing of content across multiple tests
- Assisted data entry and verification during test execution to reduce human error
- Rich text editor supports image and document attachments for individual test steps

- Multiple test result types and customizable data fields
- Imports Microsoft® Word- and Microsoft Excel-based manual tests
- Generates spreadsheet-ready results data for advanced reporting and analysis

Rational Manual Tester is included in IBM Rational Functional Tester box to help teams in performing both automated and manual testing.

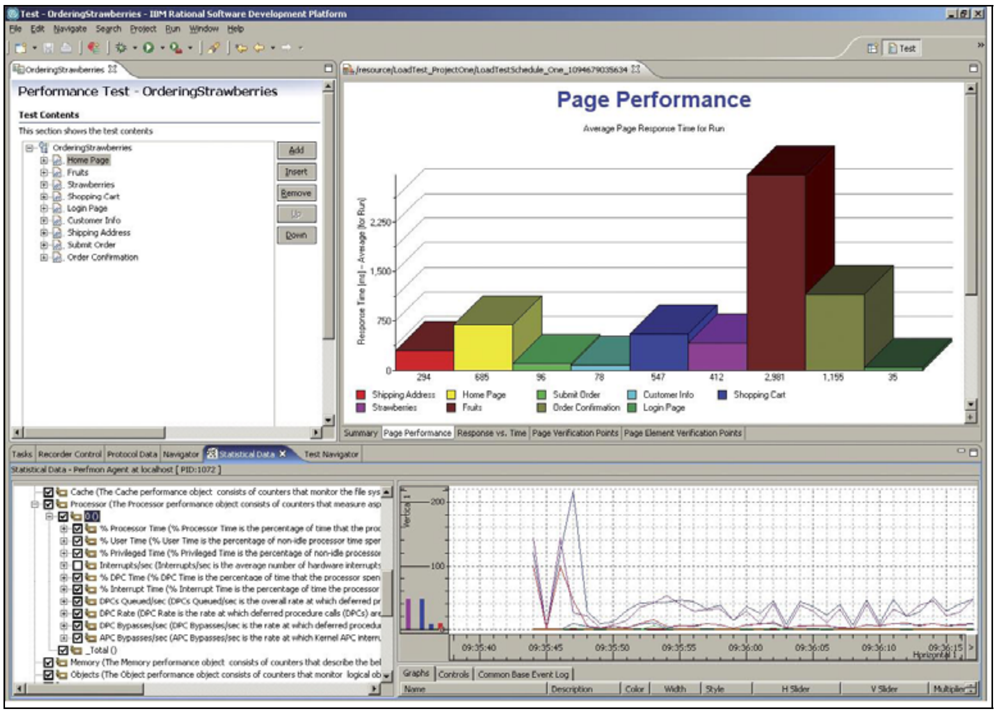
5.2.3 Rational Performance Tester

http://www-306.ibm.com/software/awdtools/tester/performance/index.html

Functionality	Good
Ease of learning and test development	Satisfactory
Documentation and support quality	Excellent
Price	Sufficient
Total Evaluation	★★★★

Rational Performance Tester helps teams pinpoint system bottlenecks before application deployment by simplifying the creation, execution and results analysis of multi-user performance tests. Tests are recordings of a user's activity within a Web browser; absolutely no programming knowledge is required to understand and modify these tests. Just a few mouse clicks engages a data pooling capability that ensures unique data for each emulated user. Using an intuitive graphical test scheduler, teams can then organize their tests to accurately simulate the different types of users and user activities the application under test will support.

During test execution, while emulating the desired number of concurrent users, Rational Performance Tester generates reports that clearly highlight poorly performing web pages, URLs and transactions. Teams are therefore able to expose performance problems in even the most complex systems, increasing the opportunity for problem capture and repair before the system goes live.



IBM Rational Performance Tester

Features and benefits of IBM Rational Performance Tester are as follows:

- Visual test editor delivering both high-level and detailed test views
- Flexible modeling and emulation of diverse user populations
- A low memory and processor footprint that enables large, multi-user tests with limited hardware resources
- Real-time reporting for immediate performance problem identification
- Automatic identification of, and support for, dynamic server responses
- Automated test data variation
- Collection and visualization of server resource data

- Rendered HTML view of Web pages visited during test recording
- Java code insertion for flexible test customization
- Linux-based user interface and test execution agents

Rational Performance Tester includes IBM Performance Optimization Toolkit

5.2.4 Rational Robot

http://www-306.ibm.com/software/awdtools/tester/robot/index.html

Functionality	Excellent
Ease of learning and test development	Poor
Documentation and support quality	Excellent
Price	Satisfactory
Total Evaluation	★★★★

General-purpose test automation tool for QA teams who want to perform functional testing of client/server applications. Lowers learning curve for testers discovering the value of test automation processes. Enables experienced test-automation engineers to uncover more defects by extending their test scripts with conditional logic to cover more of the application, and to define test cases to call external DLLs or executables. Provides test cases for common objects such as menus, lists and bitmaps, and specialized test cases for objects specific to the development environment. Includes built-in test management, and integrates with the tools in the IBM Rational Unified Process® for defect tracking, change management and requirements traceability.

Rational Robot supports:

- Applications constructed with Visual Studio .NET v.7.x WinForms and WebForms, written in .NET compliant languages, including C#, Visual Basic .NET, and C++
- Static and dynamically generated Web pages accessed from HTTP and HTTPS servers and displayed in Microsoft Internet Explorer 4.x, 5.x and 6.0 or Netscape
- 32 bit applications built with Visual Basic 4.0 through 6.0, Oracle Forms 4.5, 5.0, 6.0, PowerBuilder 5.0 through 9.0 and Web-based Oracle Forms 9i and 11i
- Applications constructed using PeopleTools 6.0 to 7.5 and HTML apps created with Web-based versions of PeopleTools 8.0+
- Applications constructed using Delphi 3.0 through 7.0
- Java applications using Sun JDK and JRE 1.1.4 through 1.4; and Microsoft JVM 3.1
- Java applets that run in Internet Explorer 4.x and 5.x with MS JVM or Sun plug-in; Microsoft Appletviewer, Sun Appletviewer from JDK 1.1.4+, Oracle Forms JInitiator 9i and 11i
- Terminal Server - hosts both the application under test and IBM Rational Robot
- Citrix MetaFrame (WIN2K)/CM Client
- Microsoft Terminal Server (WIN2K)/ MTS Server client

- Windows 2000 Server

5.3 Segue (Borland)

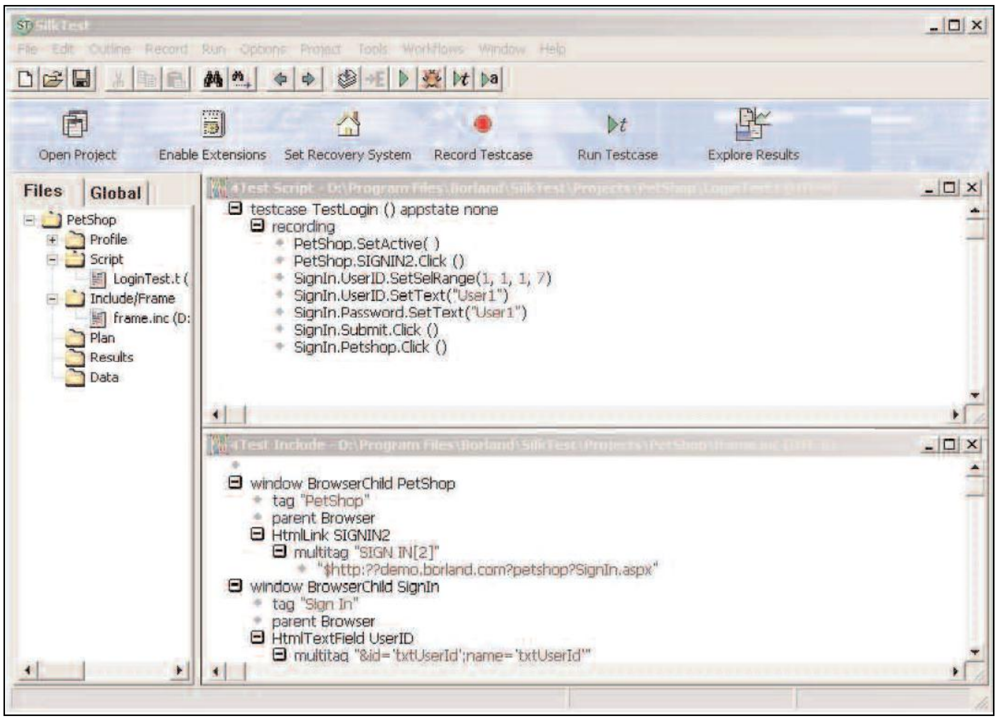
5.3.1 Borland SilkTest

http://www.borland.com/us/products/silk/silktest/index.html

Functionality	Good
Ease of learning and test development	Good
Documentation and support quality	Excellent
Price	Satisfactory
Total Evaluation	★★★★

Borland SilkTest is the leading tool for automating the functional testing process of enterprise software applications via the application Graphical User Interface (GUI). Its powerful test automation capabilities make it the perfect solution for regression, cross-platform and localization testing across a broad set of application technologies including Web, Java™ or .NET and client/server, within the confines of today's short testing cycles. Additionally, SilkTest is a powerful testing framework enables high reusability of test scripts across test projects.

SilkTest provides recording, editing and playing test scripts in the same integrated development environment (see the screen example below).



```
testcase TestLogin () appstate none
    recording
        PetShop.SetActive( )
        PetShop.SIGNIN2.Click ()
        SignIn.UserID.SetSelRange(1, 1, 1, 7)
        SignIn.UserID.SetText("User1")
        SignIn.Password.SetText("User1")
        SignIn.Submit.Click ()
        SignIn.Petshop.Click ()
```

```
window BrowserChild PetShop
    tag "PetShop"
    parent Browser
    HtmlLink SIGNIN2
        multitag "SIGN IN[2]"
            "$http:??demo.borland.com?petshop?SignIn.aspx"
window BrowserChild SignIn
    tag "Sign In"
    parent Browser
    HtmlTextField UserID
        multitag "&id='txtUserId';name='txtUserId'"
```

SilkTest offers many features to promote rapid test development, such as the Basic Workflow for recording tests, the Data Driven Workflow for linking a single test case to test data values stored in an external table, and Code Completion for improved productivity in script creation and automation infrastructure development.

To meet another classic challenge of test automation – fragile test scripts that break when Applications Under Test (AUT) are modified during development - SilkTest provides a powerful and low-maintenance GUI abstraction layer called GUI maps, which map the graphical elements of the AUT to the lower level test objects SilkTest generates.When an AUT's GUI is modified enough to require updating the GUI map, ProjectWorkspaces help you to quickly identify which GUI map to update. Often the only maintenance needed is modification of the abstraction layer; tests that reference objects defined in the layer can continue to run unmodified.

5.3.2 Borland SilkPerformer

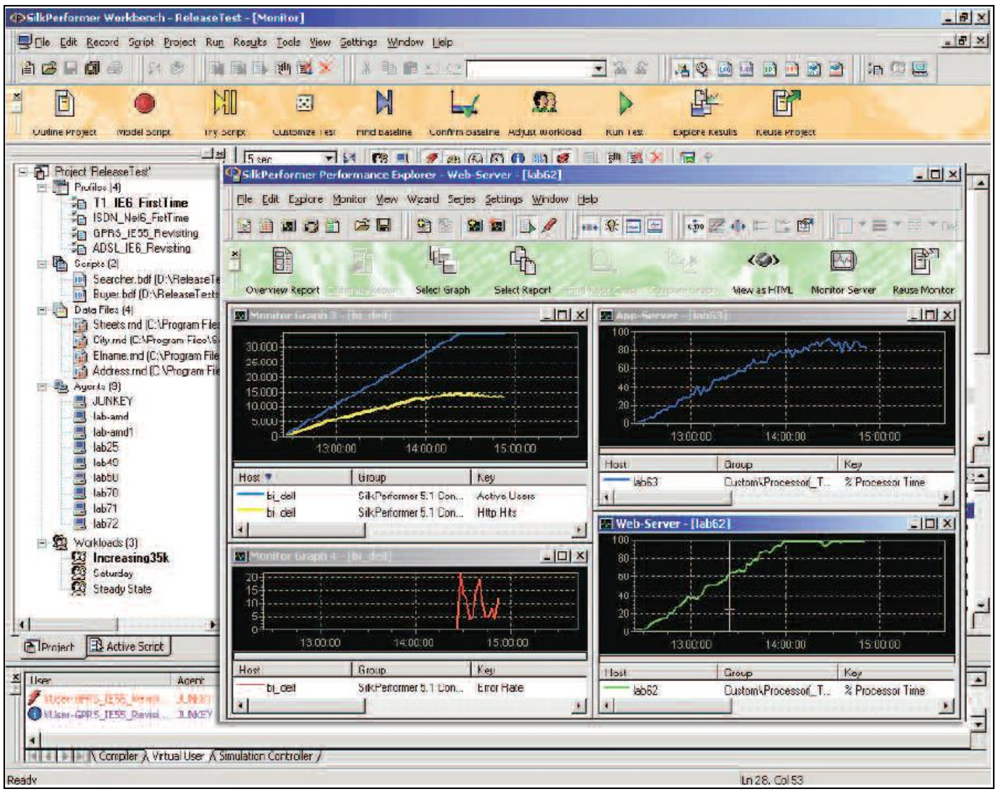
http://www.borland.com/us/products/silk/silkperformer/index.html

Borland SilkPerformer is a powerful - yet easy-to-use - enterprise-class load and stress testing solution for optimizing the quality of mission-critical applications. Visual script generation techniques and the ability to test multiple application environments with thousands of concurrent users allow you to thoroughly test an enterprise application's reliability, performance and scalability before it is deployed, regardless of its size and complexity.

Functionality	Good
Ease of learning and test development	Good
Documentation and support quality	Excellent
Price	Sufficient
Total Evaluation	★★★★

Powerful diagnostic tools and management reports help you isolate problems and make quick decisions, thereby minimizing test cycles and accelerating time-to-market.

SilkPerfomer shows tests results in convenient graphical form (see example in the picture below).



Using its innovative TrueScale technology, SilkPerformer can simulate thousands of concurrent users with a single computer - providing you with the power required for visual content verification while consuming minimal hardware resources. When required, load test agents located at remote test centers can be utilized without compromising firewall integrity.

Within a single load test, virtual users working with different Internet, middleware and database protocols - in addition to varied computing environments - can be simulated. Support for multibyte character sets and UTF-8 allows the testing of internationalized applications that utilize Unicode. Client IP address simulation allows for the testing of load-balanced sites. The Benchmark Description Language (BDL) of SilkPerformer is designed specifically for testing purposes and provides an easy and powerful means of controlling user activitiy.

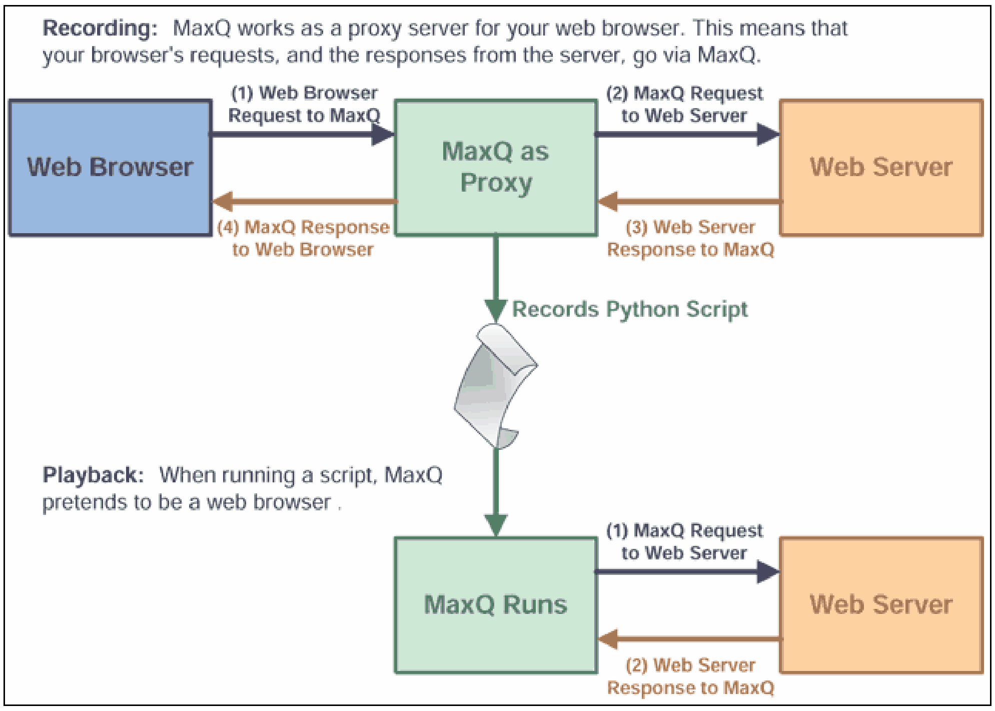
5.3.3 MaxQ

http://maxq.tigris.org/

MaxQ is a web functional testing tool. MaxQ records you using a web site. It turns the links you click on and any other input into a Python script that you can play back at any time.

The working process of the tool is represented in the picture below:

Functionality	Sufficient
Ease of learning and test development	Satisfactory
Documentation and support quality	Poor
Price	Excellent
Total Evaluation	★★★


Recording: MaxQ works as a proxy server for your web browser. This means that your browser's requests, and the responses from the server, go via MaxQ.
Web Browser
(1) Web Browser Request to MaxQ
MaxQ as Proxy
(2) MaxQ Request to Web Server
Web Server
(4) MaxQ Response to Web Browser
(3) Web Server Response to MaxQ
Records Python Script
Playback: When running a script, MaxQ pretends to be a web browser .
MaxQ Runs
(1) MaxQ Request to Web Server
Web Server
(2) Web Server Response to MaxQ

You might use MaxQ to:

- Check that your web site still works (regression test).

- Check that your web site is producing valid HTML (using JTidy).
- Automatically extract information from, or take some action on, someones else's web site.

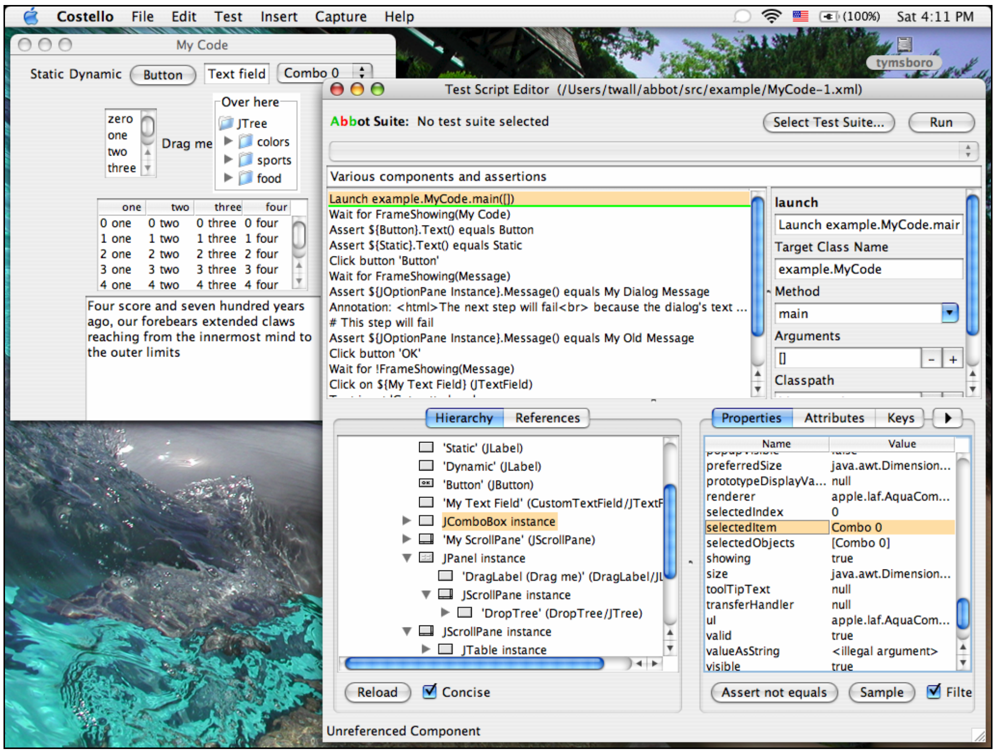
MaxQ is written in Java so it runs anywhere. MaxQ is free and open source.

5.3.4 Abbot

http://abbot.sourceforge.net/

Abbot is free and open source. The Java GUI Test Framework of this tool helps you to test your Java UI. It comprises Abbot, which lets you programmatically drive UI components, and Costello (built on Abbot), which allows you to easily launch, explore and control an application. The framework may be used with both scripts and compiled code.

Functionality	Sufficient
Ease of learning and test development	Sufficient
Documentation and support quality	Sufficient
Price	Excellent
Total Evaluation	★★★



Test Script Editor (/Users/twall/abbot/src/example/MyCode-1.xml)

Abbot Suite: No test suite selected

Various components and assertions

```
Launch example.MyCode.main([])
Wait for FrameShowing(My Code)
Assert ${Button}.Text() equals Button
Assert ${Static}.Text() equals Static
Click button 'Button'
Wait for FrameShowing(Message)
Assert ${JOptionPane Instance}.Message() equals My Dialog Message
Annotation: <html>The next step will fail<br> because the dialog's text ...
# This step will fail
Assert ${JOptionPane Instance}.Message() equals My Old Message
Click button 'OK'
Wait for !FrameShowing(Message)
Click on ${My Text Field} (JTextField)
```

launch: Launch example.MyCode.main
Target Class Name: example.MyCode
Method: main
Arguments: []
Classpath

Four score and seven hundred years ago, our forebears extended claws reaching from the innermost mind to the outer limits

```
'Static' (JLabel)
'Dynamic' (JLabel)
'Button' (JButton)
'My Text Field' (CustomTextField/JTextF
JComboBox instance
'My ScrollPane' (JScrollPane)
JPanel instance
  'DragLabel (Drag me)' (DragLabel/JL
  JScrollPane instance
    'DropTree' (DropTree/JTree)
JScrollPane instance
  JTable instance
```

Name	Value
preferredSize	java.awt.Dimension...
prototypeDisplayVa...	null
renderer	apple.laf.AquaCom...
selectedIndex	0
selectedItem	Combo 0
selectedObjects	[Combo 0]
showing	true
size	java.awt.Dimension...
toolTipText	null
transferHandler	null
ul	apple.laf.AquaCom...
valid	true
valueAsString	<illegal argument>
visible	true

Unreferenced Component

Example of Costello script editor screen

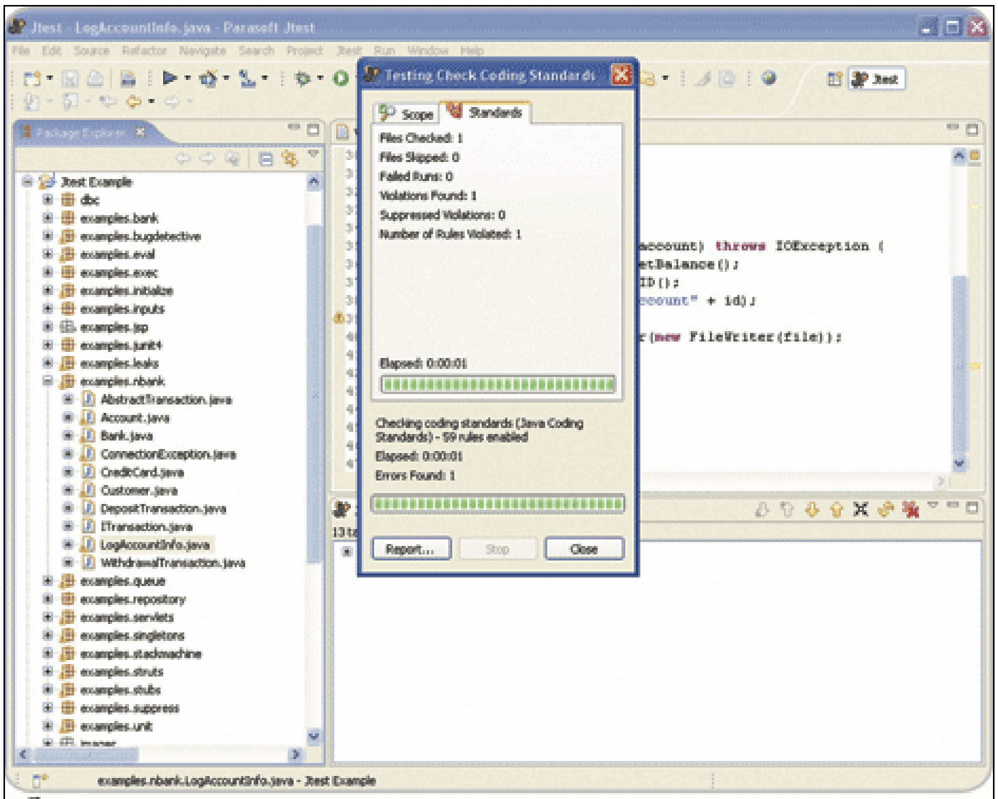
Users can also record actions directly into a script, which controls the event playback and testing. This form of test is more suitable to integration/functional testing. The provided Costello script editor can record user actions and facilitate script construction and maintenance.

5.4 Parasoft JTest 8.0

http://www.parasoft.com/jsp/products/home.jsp?product=Jtest

Parasoft Jtest is a Java static analysis and testing product for development teams working on Java EE, SOA, Web, and other Java applications. It promotes a "test early and often" strategy so that quality is built into the code, and bugs are exposed upon introduction-- when fixing them requires minimal rework.

Functionality	Excellent
Ease of learning and test development	Good
Documentation and support quality	Good
Price	Poor
Total Evaluation	★★★★



```
Files Checked: 1
Files Skipped: 0
Failed Runs: 0
Violations Found: 1
Suppressed Violations: 0
Number of Rules Violated: 1

Elapsed: 0:00:01

Checking coding standards (Java Coding
Standards) - 59 rules enabled
Elapsed: 0:00:01
Errors Found: 1
```

```
account) throws IOException {
etBalance();
ID();
ccount" + id);

r(new FileWriter(file));
```

Figure 1

Basic testing with JTest

Static code analysis

Jtest statically analyzes code to check compliance with specified coding standards. Coding standards in Jtest are implemented as rules. Jtest is preconfigured to perform coding standards analysis with built-in rule sets that include 700+ coding standard rules related to:

- Coding conventions
- Duplicate code
- EJBs
- Exceptions

- Formatting
- Garbage collection
- Hibernate
- Initialization
- Internationalization
- Java EE
- Java ME / J2ME
- JavaBeans
- Javadoc comments
- JDBC
- JDK 1.5
- JSP
- JUnit test cases
- Naming conventions
- Object-oriented programming
- Portability
- Security
- Serialization
- Servlets
- Struts
- Threads and synchronization
- Unused code

The rules enabled by default in most coding standard-checking Test Configurations (for instance, Parasoft's Recommended Rules, Check Coding Standards, Run All Tests, etc.) have been shown to make an immediate and significant improvement to code. Java software that follows this core set of guidelines will be faster, more secure, easier to maintain, and less likely to experience functional problems.

In addition, Jtest includes coding standards that help you add Design by Contract (DbC) comments to your code and coding standards that help you check whether metric guidelines are satisfied. Violations of 200+ rules can be corrected automatically with Jtest's Quick Fix feature.

Unit testing

Unit testing involves testing software code at its smallest functional point, which is typically a single class or function. By testing each unit in isolation, unit testing helps detect errors that application level testing might not find.

Jtest can automate any or all of these types of unit testing:

- Reliability testing (also known as white-box testing or construction testing) is used to confirm that Java code is structurally sound, and can process a wide variety of permissible inputs without throwing an uncaught runtime exception

- Functional testing (black-box testing) is performed with the automatically-generated JUnit test cases to verify whether the class's public interface operates as described in the specification.

- Regression testing is performed to retest code with all available test cases. Jtest reports error messages when test case outcomes from the current test run do not match the expected test case outcomes or (for unverified automatically-generated test cases) when they do not match the test case outcomes achieved during the initial test run.

The level and scope of testing performed can be customized to suit your requirements and your preferences.

Memory usage and memory leaks

In addition, Jtest can be configured to watch memory usage and report any detected memory leaks. If memory leak detection is enabled, Jtest monitors object allocate events and object free events during test execution.

Uncaught runtime exceptions

Jtest's automated unit testing of uncaught runtime exceptions allows to:

- Remove the causes of problematic runtime exceptions.

- Verify that expected runtime exceptions function correctly and are clearly documented.

- Verify that input restrictions are clearly documented to prevent illegal inputs which could lead to unwanted runtime exceptions.

Stubs

When Jtest automatically generates test cases, Jtest adds stubs to test cases if your class references external resources. However, you can define your own stubs for any test case-- automatically-generated test cases as well as user-defined test cases. When you use user-defined stubs, you have complete control over what values or exceptions an external method returns to the class under test-- without having to have the actual external method completed and/or available.

Test Case Editor

The Test Case Editor provides a way to specify test cases without writing or modifying code. It is especially helpful if you want a convenient way to run the same user-defined test case with different method inputs (for instance, to better test functionality or increase test coverage). You define the range of test values in the GUI or in a data source, then set up one "template" test case. During testing, Jtest cycles through the full input set, creating and running multiple test cases based on the one "template" test case that you defined. In this way, you can test a large number of user-defined inputs by configuring just one test case.

Design by Contract

Design by Contract (DbC) is a structured way of writing comments to define what code should do. The contract requires components of the code (such as classes or methods) to follow certain specifications as they interact with each other. The interactions between these components must fulfill a set of predetermined mutual obligations.

This is an example of a class with Design by Contract comments:

```
public class ShoppingCart {
/**
* @pre item != null
* @post $result > 0
*/
public float add (Item item) {
_items.addElement (item);
_totalCost += item.getPrice ();
return _totalCost;
}
private float _totalCost = 0;
private Vector _items = new Vector ();
```

The contract specifies:

1) A precondition ("@pre item != null") which specifies that the item to be added to the shopping cart shouldn't be "null".

2) A postcondition ("@post $result > 0") which specifies that the value returned by the method should always be greater than 0.

Preconditions and postconditions can be thought of as sophisticated assertions. Preconditions are conditions that the method’s client needs to satisfy before the method can execute; a violation of a precondition indicates a problem with the client (the client is misusing the method). Postconditions are conditions that the implementor of the class guarantees will always be satisfied after a method completes; a postcondition violation indicates a problem within the method.

Jtest and Jcontract, use the benefits of DbC so:

- Functional test cases are created automatically. If you currently create your functional test cases manually, this means fewer resources spent creating test cases and more resources you can dedicate to more complex tasks, such as design and coding. If you do not currently perform functional testing, this will translate to more reliable software/components.
- Functional test cases are automatically updated when you modify the contracts.
- Class/component misuse is automatically detected.

- The class implementation can assume that input arguments satisfy the preconditions, so the implementation can be simpler and more efficient.
- The class client is guaranteed that the results will satisfy the postconditions.

Test Configurations

A Test Configuration is a collection of settings that define a test scenario that Jtest will run.

Each time Jtest runs a test it uses the designated Test Configuration (or the Default Test Configuration, if no Test Configuration was explicitly selected). The Test Configuration determines all test parameters. For example, it determines parameters such as:

- The type of tests (coding standard checking, test case generation, test case execution, etc.)
- The rules checked during coding standard checking
- The parameters for automatic test case generation
- The scope of each test (what lines to cover, what cutoff date to use, etc.)

Jtest includes a set of preconfigured "built-in" Test Configurations that represent the test scenarios that Java developers run most frequently (according to Parasoft).

5.5 Push-to-test TestMaker 6.2

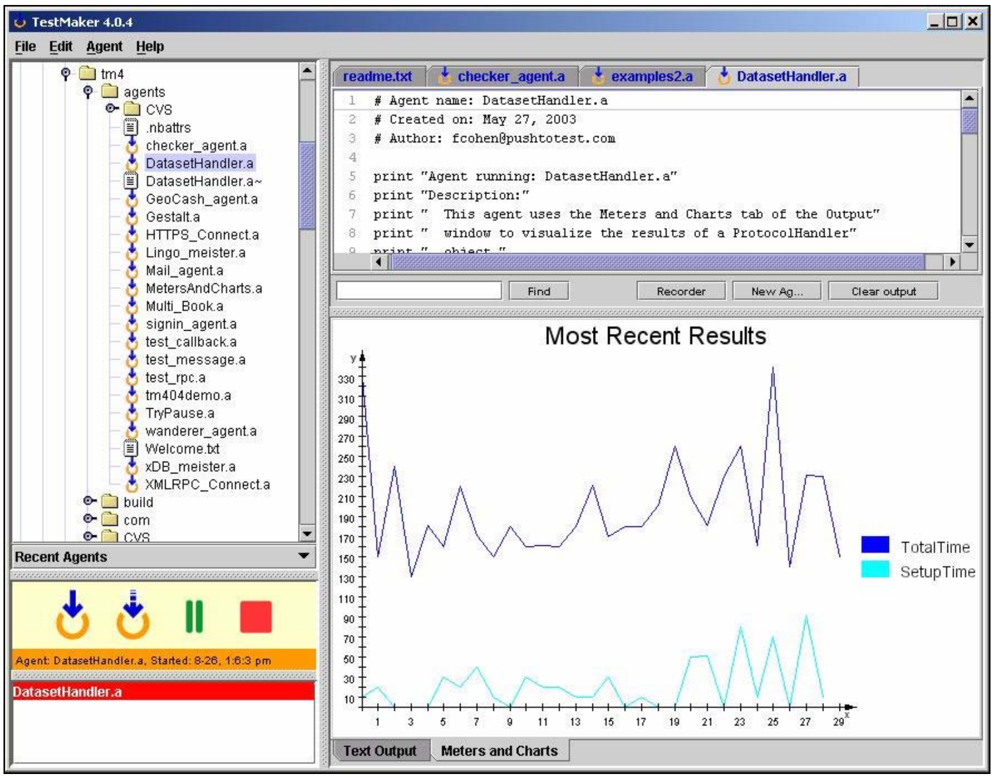
TestMaker is a utility and framework to build and use intelligent test agents to check Web-enabled applications for scalability, functionality and performance. TestMaker test agents implement user behavior to drive a Web-enabled application as a real user would. TestMaker is a flexible, powerful central place to measure an application's ability to enable a user to achieve their goals.

TestMaker is a common framework to build and run tests. While the TestMaker framework does not require extensive software coding experience for the person wanting to mount a simple test, TestMaker's environment provides all the richness and versatility of the Python and Java languages combined to build intelligent test agents. Plus, the TestMaker environment makes it easy to begin writing intelligent test agents for your Web-enabled application, especially for Web Services.

Functionality	Good
Ease of learning and test development	Satisfactory
Documentation and support quality	Satisfactory
Price	Excellent
Total Evaluation	★★★★

TestMaker is a free open source utility with a GPL license.

The example of TestMaker window is represented on the picture below:

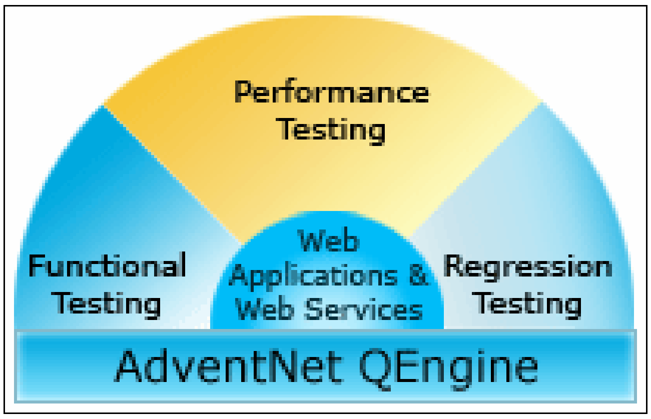


```
# Agent name: DatasetHandler.a
# Created on: May 27, 2003
# Author: fcohen@pushtotest.com

print "Agent running: DatasetHandler.a"
print "Description:"
print "  This agent uses the Meters and Charts tab of the Output"
print "  window to visualize the results of a ProtocolHandler"
print "  object "
```

5.6 AdventNet QEngine 6.7

AdventNet QEngine offers integrated solutions to test and verify the functionality and performance of both web applications and web services.



Web browser events are recorded as Jython scripts which provides the fastest and easiest way to create, read, and manipulate test scripts, with no programming.

QEngine realistically simulates load for a large number of users to gather the performance and stability information of your web application. It accurately simulates a large number of virtual users performing a defined set of transactions (or business cases) in your web application. This supports various workload types such as steady state workload (Normal), increasing workload (Ramp-up) and repeated load testing for a large number of virtual users based on exit criteria (Bburn-in).

Functionality	Good
Ease of learning and test development	Good
Documentation and support quality	Good
Price	Good
Total Evaluation	★★★★

5.7 SLAMD 2.0.0-alpha1

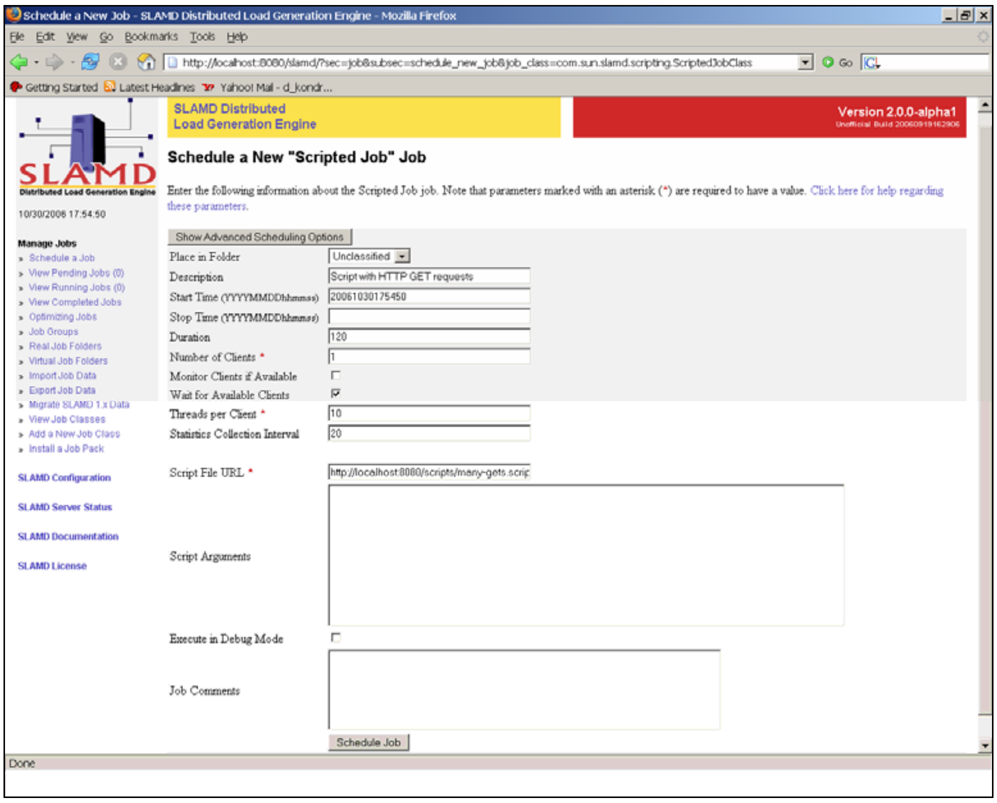
http://www.slamd.com/

The SLAMD Distributed Load Generation Engine is a Java-based application designed for stress testing and performance analysis of network-based applications. Unlike many other load generation utilities, SLAMD provides an easy way to schedule a job for execution, either immediately or at some point in the future. It distributes job information to a number of client systems, and then executes concurrently on those clients to generate higher levels of load and more realistic usage patterns than a standalone application operating on a single system. Upon completing the assigned task, the clients report the results of their execution back to the server where the data is combined and summarized.

Functionality	Excellent
Ease of learning and test development	Good
Documentation and support quality	Good
Price	Excellent
Total Evaluation	★★★★★

Using an HTML-based administrative interface, users can then view those results either in summary form or in varying levels of detail. Users may also view graphs of the statistics collected and may even export that data into a format that may be imported into spreadsheets or other external applications for further analysis.

The SLAMD environment is also highly extensible. Custom jobs that interact with network applications and collect statistics may be developed either by writing Java class files or executed using the embedded scripting engine. The kinds of statistics that are collected while jobs are being executed may also be customized, as well as the kinds of information that may be provided to a job to control the way in which it operates.


Schedule a New Job - SLAMD Distributed Load Generation Engine - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8080/slamd/?sec=job&subsec=schedule_new_job&job_class=com.sun.slamd.scripting.ScriptedJobClass
Getting Started Latest Headlines Yahoo! Mail - d_kondr...
SLAMD Distributed Load Generation Engine
Version 2.0.0-alpha1
SLAMD Distributed Load Generation Engine
10/30/2006 17:54:50
Manage Jobs
Schedule a Job
View Pending Jobs (0)
View Running Jobs (0)
View Completed Jobs
Optimizing Jobs
Job Groups
Real Job Folders
Virtual Job Folders
Import Job Data
Export Job Data
Migrate SLAMD 1.x Data
View Job Classes
Add a New Job Class
Install a Job Pack
SLAMD Configuration
SLAMD Server Status
SLAMD Documentation
SLAMD License
Show Advanced Scheduling Options
Schedule Job
Done

Schedule a New "Scripted Job" Job

Enter the following information about the Scripted Job job. Note that parameters marked with an asterisk (*) are required to have a value. Click here for help regarding these parameters.

Place in Folder	Unclassified
Description	Script with HTTP GET requests
Start Time (YYYYMMDDhhmmss)	20061030175450
Stop Time (YYYYMMDDhhmmss)	
Duration	120
Number of Clients *	1
Monitor Clients if Available	☐
Wait for Available Clients	☑
Threads per Client *	10
Statistics Collection Interval	20
Script File URL *	http://localhost:8080/scripts/many-gets.scrip
Script Arguments	
Execute in Debug Mode	☐
Job Comments	

Scheduling a job with SLAMD administrative interface

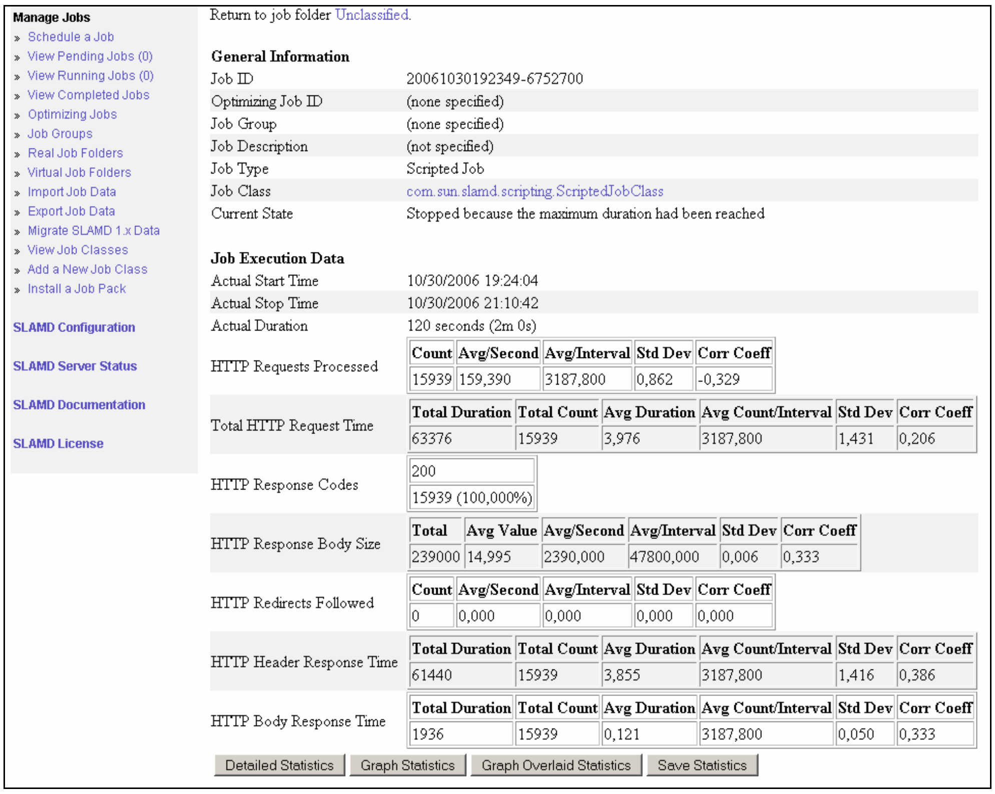
Test Jobs

SLAMD was originally developed for the purpose of benchmarking and analyzing the performance of network applications.

SLAMD provides test jobs for Web servers and Web-based applications, relational databases, and mail servers. It can also be used for non-network based applications (and in fact, it is used for comparing things like CPU power and memory latency across a number of different kinds of systems), although its distributed nature makes it ideal for systems that can be accessed remotely.

SLAMD provides a Java-based API to make it possible to quickly develop custom workloads, and it also contains an embedded scripting engine that can make it easy to stress applications using protocols like LDAP, HTTP, SMTP, IMAP, and POP, or any database that can be accessed via JDBC.

SLAMD also includes tools for recording and playing back TCP traffic, and a utility for intercepting LDAP communication and writing it as a script that may be executed in the SLAMD scripting engine.



Return to job folder Unclassified.

General Information

Job ID	20061030192349-6752700
Optimizing Job ID	(none specified)
Job Group	(none specified)
Job Description	(not specified)
Job Type	Scripted Job
Job Class	com.sun.slamd.scripting.ScriptedJobClass
Current State	Stopped because the maximum duration had been reached

Job Execution Data

Actual Start Time	10/30/2006 19:24:04
Actual Stop Time	10/30/2006 21:10:42
Actual Duration	120 seconds (2m 0s)

HTTP Requests Processed

Count	Avg/Second	Avg/Interval	Std Dev	Corr Coeff
15939	159,390	3187,800	0,862	-0,329

Total HTTP Request Time

Total Duration	Total Count	Avg Duration	Avg Count/Interval	Std Dev	Corr Coeff
63376	15939	3,976	3187,800	1,431	0,206

HTTP Response Codes

200
15939 (100,000%)

HTTP Response Body Size

Total	Avg Value	Avg/Second	Avg/Interval	Std Dev	Corr Coeff
239000	14,995	2390,000	47800,000	0,006	0,333

HTTP Redirects Followed

Count	Avg/Second	Avg/Interval	Std Dev	Corr Coeff
0	0,000	0,000	0,000	0,000

HTTP Header Response Time

Total Duration	Total Count	Avg Duration	Avg Count/Interval	Std Dev	Corr Coeff
61440	15939	3,855	3187,800	1,416	0,386

HTTP Body Response Time

Total Duration	Total Count	Avg Duration	Avg Count/Interval	Std Dev	Corr Coeff
1936	15939	0,121	3187,800	0,050	0,333

SLAMD Job information including average statistics

Clients

Because SLAMD is a distributed load generation engine, the SLAMD server itself does not execute any of the jobs. Rather, the actual execution is performed by SLAMD clients, and the server merely coordinates the activity of those clients.

Because SLAMD is a distributed load generation engine, the SLAMD server itself does not execute any of the jobs. Rather, the actual execution is performed by SLAMD clients, and the server merely coordinates the activity of those clients.

The client application communicates with the SLAMD server using a TCP-based protocol. Therefore, it is possible to install clients on different machines than the SLAMD server is installed. In fact, this is recommended so that the client and server do not compete for the same system resources, which could interfere with the ability of the client to obtain accurate results. Further, it is possible to connect to the SLAMD server with a large number of clients to process multiple jobs concurrently, and in such cases it is best to have those clients distributed across as many machines as possible to avoid problems in which the clients are competing with each other for system resources.

Resource Monitor Client

The resource monitor client is a special type of client that may be used to collect statistical data beyond what may be collected by the job. In particular, the resource monitor client is intended to measure system metrics like CPU utilization, network traffic, or I/O load. Resource monitor clients may be run on any system that can communicate with the SLAMD server, although they will typically be run on a server against which load is being generated. It may also be beneficial to run monitor clients on the same system as SLAMD clients to ensure that the load generation process is not overloading the client system.

Client Manager

The client manager is an application that runs on the client system and handles the process of starting and stopping individual clients. Unlike the clients themselves, however, the client manager is able to remain running even if the SLAMD server is shut down. If that occurs, then the client manager will periodically check for the server to come back online and should re-establish the connection within 30 seconds of the server coming back online. This does not automatically re-establish the individual client connections, but does provide a simple means of creating those client connections through the administration interface.

Administration Interface

All interaction with the SLAMD server is performed through an HTML administration interface. This interface provides a number of capabilities, including:

- Scheduling new jobs for execution
- The results view of jobs that have completed execution
- Status information about the SLAMD server
- New job classes available for use in the SLAMD environment
- Configurftion of the SLAMD server
- HTML interface appearance customization

Scheduling Jobs for Execution

One of the most important capabilities of the SLAMD server is the ability to schedule jobs for execution. It is possible to schedule them to execute immediately or at some point in the future, on one or more client systems, using one or more threads per client system, and with a number of other options.

Managing Scheduled Jobs

Once a job has been scheduled for execution, it will be added into the pending jobs queue to await execution. Once all of the criteria required to start the job have been met (e.g., the start time has arrived, the job is not disabled, all dependencies have been satisfied, and an appropriate set of clients is available), then that job will be moved from the pending jobs queue into the running jobs queue and the job will be sent out to the clients for processing. When that job has completed execution, the job will be removed from the running jobs queue and the job information will be updated in the configuration directory.

Viewing Job Execution Results

Once a job has been executed and all clients have sent the results back to the SLAMD server, those results will be made available through the administration interface. Those results will be made available in a variety of forms, and the data collected can even be exported for use with external programs like spreadsheets or databases.

Optimizing Jobs

When using SLAMD to run benchmarks against a network application, it is often desirable to find the configuration that yields the best performance. In many cases, this also involves trying different numbers of clients or threads per client to determine the optimal amount of load that can be placed on the server to yield the best results. To help automate this process, SLAMD offers optimizing jobs.

An optimizing job is actually a collection of smaller jobs. It will run the same job repeatedly with increasing numbers of threads per client until it finds the number that yields the best performance for a particular statistic.

Organizing Job Information

After the SLAMD server is used to schedule and run a large number of jobs, the page that stores completed job information can grow quite large, and the process of displaying that page can take more time and consume more server resources.

Therefore, for the purposes of both organization and conserving system resources, the server offers the ability to arrange jobs into folders. It is also possible to specify a variety of criteria that can be used to search for job information, regardless of the folder in which it is contained.

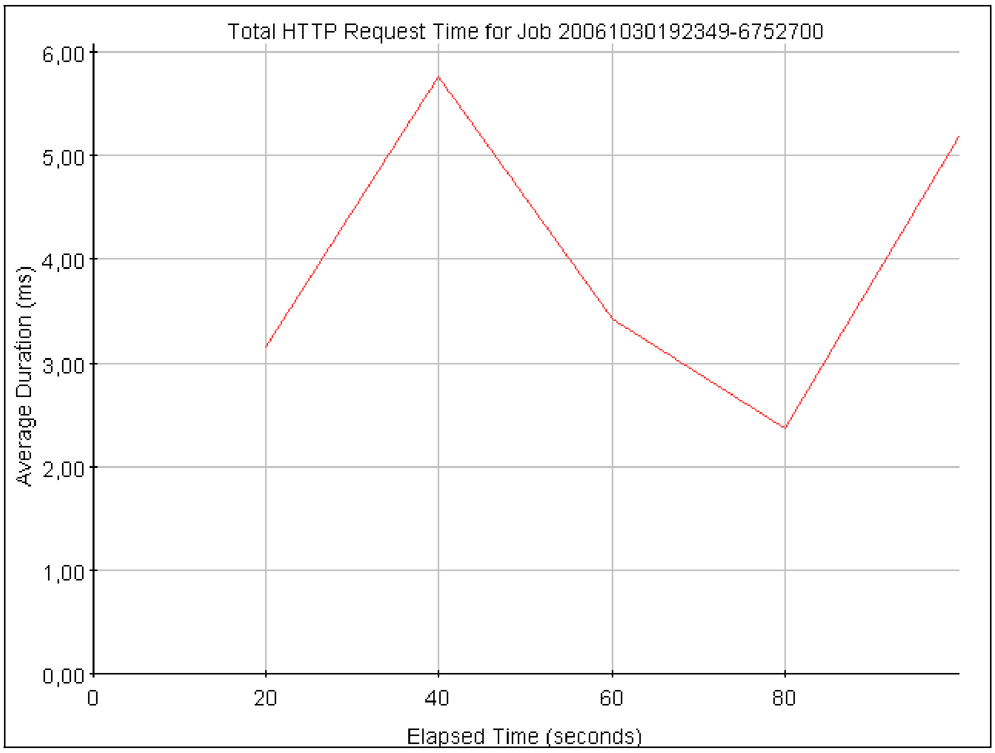
Adding New Job Classes

The SLAMD server was designed in such a way that it is very extensible. One of the ways that this is evident is the ability for an end user to develop a new job class and add that class to the SLAMD server. Once that class has been added to the SLAMD server, it will immediately be possible to schedule and run jobs that make use of that job class. It is not necessary to copy that job class to all client systems, as that will be done automatically whenever a client is asked to run a job for which it does not have the appropriate job class.

Generating Reports

SLAMD report may contain information about a single job or optimizing job, a selected set of jobs or optimizing jobs within a folder, or information about all jobs and optimizing jobs contained in a given folder.

The example of available report is represented on the picture below:



Scripting Language

The scripting language provided by SLAMD was developed specifically for the purpose of executing SLAMD jobs. As such, it is not based on any single existing language, but rather uses its own syntax (largely a hybrid of Java and Pascal). However, it is a very simple language and there should not be any significant learning curve for anyone with at least a basic background in programming.

There are three basic building blocks in the SLAMD scripting language:

- Variables are entities that will generally hold a value and that have several methods associated with them that can perform various kinds of operations on that value and/or on zero or more other variables (roughly the equivalent of a Java class). There are a number of basic and specialized variable types provided with SLAMD, and it is possible to extend the scripting language to add new variable types.

- Instructions provide a means of performing flow control within the script, and for actually invoking the methods associated with variables. The set of instructions available for use in the SLAMD scripting language is fixed and cannot be readily extended.

- Arguments are elements that can be provided to an instruction to customize the way that it operates. An argument may take a few different forms, and can include variables, method calls, and literal values.

The SLAMD scripting language is inherently method-driven. Therefore, the real functionality of a script is defined by the types of variables that it uses and the methods that those variables offer. While some kinds of variables provided by SLAMD are specialized and are useful only for certain kinds of tests (e.g., those variables intended for communicating with HTTP/S, JDBC, LDAP, SMTP, POP, IMAP, FTP servers), there is also a set of core variable types that are suitable for use in a wide range of settings.

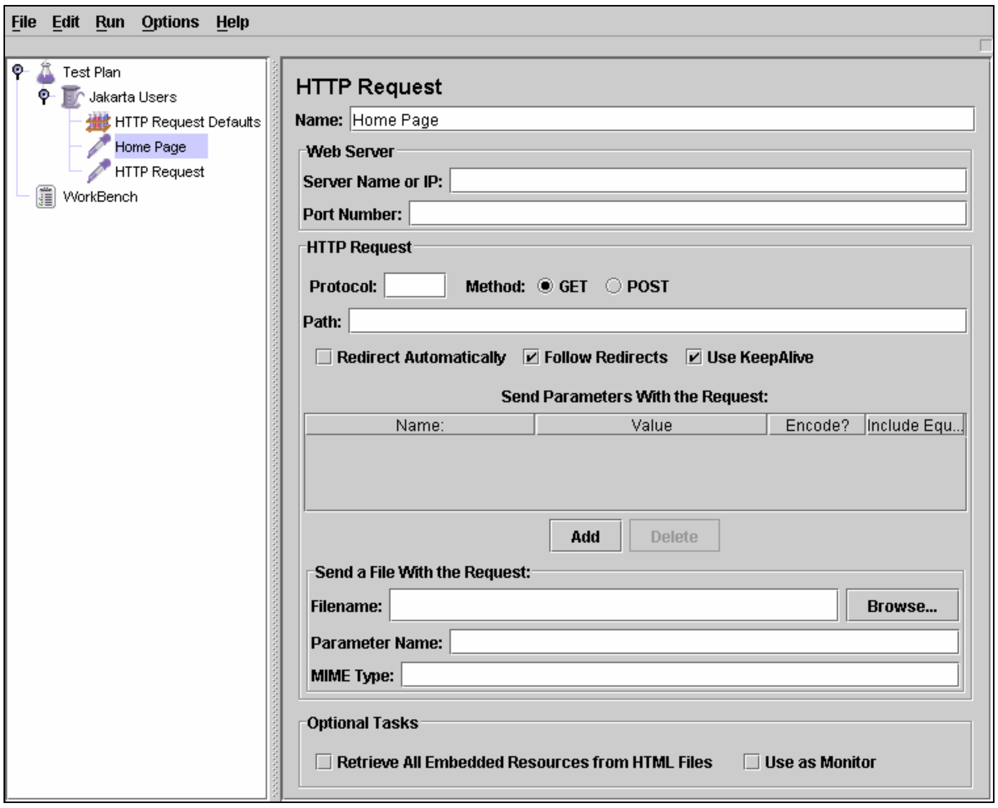
5.8 JMeter 2.2

http://jakarta.apache.org/jmeter/

Apache JMeter is a 100% pure Java desktop application designed to load test client/server software (such as a web application). It may be used to test performance both on static and dynamic resources such as static files, Java Servlets, CGI scripts, Java objects, databases , FTP servers , and more. JMeter can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types.

Functionality	Good
Ease of learning and test development	Satisfactory
Documentation and support quality	Good
Price	Excellent
Total Evaluation	★★★★

Additionally, JMeter can help you regression test your application by letting you create test scripts with assertions to validate that your application is returning the results you expect. For maximum flexibility, JMeter lets you create these assertions using regular expressions.



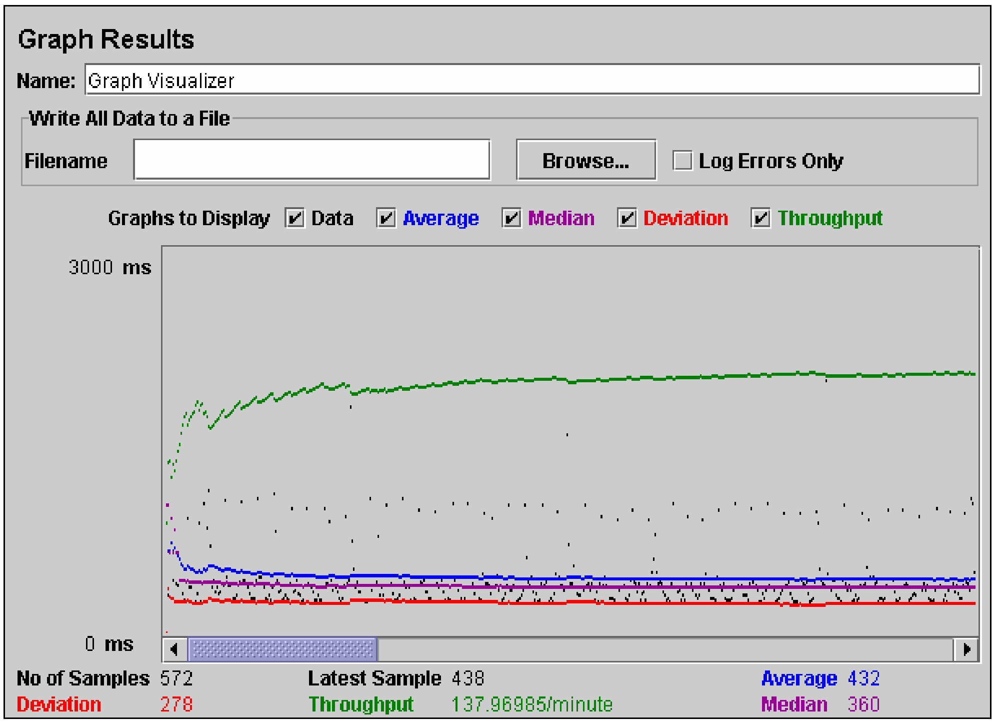
Name: Home Page

Method: [✓ GET] POST

[ ] Redirect Automatically [✓] Follow Redirects [✓] Use KeepAlive

[ ] Retrieve All Embedded Resources from HTML Files [ ] Use as Monitor

Create HTTP Request


Graph Results

Name: Graph Visualizer

Write All Data to a File

Filename [ ] Browse... ☐ Log Errors Only

Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

3000 ms

0 ms

No of Samples 572 Latest Sample 438 Average 432

Deviation 278 Throughput 137.96985/minute Median 360

View Graph Results

Apache JMeter features include:

- Load and performance testing of HTTP and FTP servers as well as arbitrary database queries (via JDBC)
- Complete portability and 100% Java purity
- Full Swing and lightweight component support (precompiled JAR uses packages javax.swing.*).
- Full multithreading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.
- Careful GUI design allows faster operation and more precise timings.
- Caching and offline analysis/replaying of test results.
- High Extensibility:
  - Pluggable Samplers allow unlimited testing capabilities.
  - Several load statistics may be chosen with pluggable timers .
  - Data analysis and visualization plug-ins allow great extendibility as well as personalization.
  - Functions (which include JavaScript) can be used to provide dynamic input to a test
  - Scriptable Samplers (BeanShell is supported in version 1.9.2 and above)

5.9 IValidator 2.2.0

http://www.ivalidator.org/ivalidator/en/index.html

Functionality	Satisfactory
Ease of learning and test development	Excellent
Documentation and support quality	Satisfactory
Price	Excellent
Total Evaluation	★★★★

iValidator is a framework for XML-based test automation of complex test scenarios. iValidator is completely written in Java. The framework is available under an open source licence.

iValidator has proven to help implementing complex automatic integration tests that are easily maintainable and extensible (even by non-developers, at least to a certain extent). Therefore it supplements JUnit for test scenarios derived from business use cases. The automatic integration tests show that the system under test (still) fulfills the business requirements and drives the business processes, even when changed in unexpected ways.

Simply spoken, iValidator implements a test runner like JUnit. But since integration testing is a little bit more complex than class testing, it defines some more building blocks and interfaces.

iValidator provides the following advantages for testing automation:

- Support of complex scenarios
- Extensive control mechanisms, e.g. parallel tests
- Reusability of unit tests
- Flexibility based on separation of code and data
- Initialisation of test environment
- Validation, Reporting, Logging
- System independency
- Support of all test levels
- Eclipse Plug-In
- JUnit Support
- Flexible interfaces for the test descriptions

The building blocks of test scenarios are:

- **Setups and Teardowns**: In contrast to the simple setUp and tearDown methods in JUnit we have classes that implement setups and teardowns since we want to reuse in different test scenarios, maybe even parameterizing them.
- **Test steps**: Each test step realizes a reusable business task, e. g., creating a customer. Since this step may be part of different test scenarios it has to be parameterized.

- **Checkups**: A checkup checks the results of a test or several test steps for the expected outcome. If it finds an error, it may decide how severe the failure is – and in the test scenario description it has to be decided if the scenario has to be stopped or may continue on this error.

- **Flows**: A flow combines several Setups, test steps, checkups and/or tear downs to create a reusable higher abstraction. Flows may be nested. A test scenario is therefore a flow – maybe consisting of several sub-flows.

- **Adapters**: An adapter abstracts from the technical and business interface of the SUT or a simulator. It is used to call the SUT or a simulator from the setups, test steps, checkups, and teardowns. When the technical realization of the interface changes, e. g., from CORBA to SOAP, only the adapter has to be changed – the test scenarios using the adapters are left unchanged. If the business interface changes, e. g., taking one more parameter, the adapter may provide a default parameter while another adapter exposes the new parameter for usage in new scenarios.

- **Test parameters and test data**: Test parameters and test data are fed into the scenario and flow through the scenario, being changed or enhanced (consider generated technical ids, for example) by the setups and test steps. The teardowns may use them to clean up.

The interfaces of iValidator are used to integrate the framework into different environments:

- **Repository**: The repository contains the definition of test scenarios, test parameters and test data. The repository may be, e. g., a set of XML files, some database tables. iValidator simply requires the environment to supply these resources. iValidator contains an XML reader as standard adapter for the adapter.

- **Reporting:** The result of the test run is reported via the reporting interface. The standard implementation creates XML files but other reporting sinks as, e. g., a database, are supported.

- **Logging**: iValidator may use different logging systems, e. g., log4j or Java 1.4 logging, to log its execution. This logging is separate from the SUT logging.

5.10 Jamelon

http://jameleon.sourceforge.net/index.html

Functionality	Good
Ease of learning and test development	Satisfactory
Documentation and support quality	Satisfactory
Price	Excellent
Total Evaluation	★★★★

Jameleon is an automated testing framework that can be easily used by technical and non-technical users alike. One of the main concepts behind Jameleon is to create a group of keywords or tags that represent different screens of an application. All of the logic required to automate each particular screen can be defined in Java and mapped to these keywords. The keywords can then be organized with different data sets to form test scripts without requiring an in-depth knowledge of how the application works. The test scripts are then used to automate testing and to generate manual test case documentation.

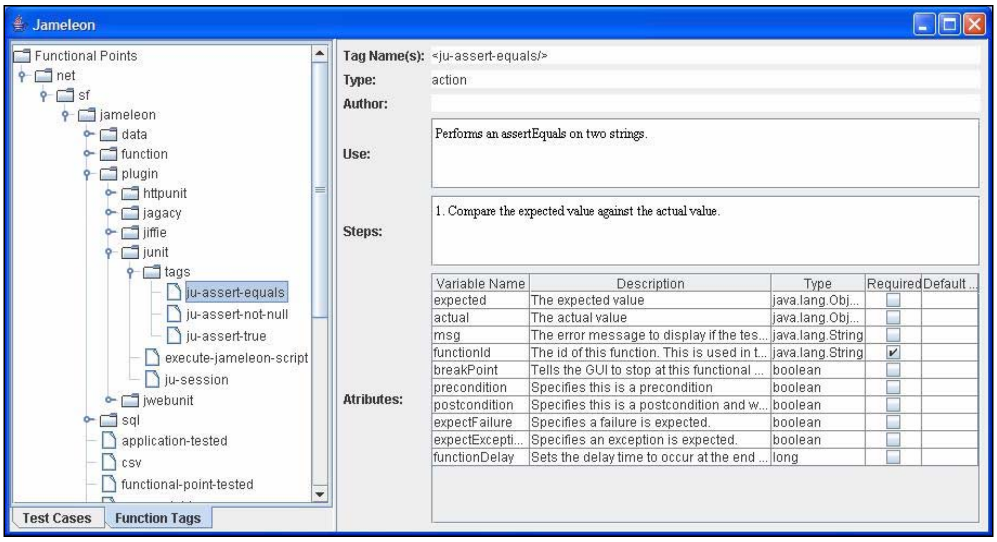
Jameleon was designed to test many kinds of applications. To make this possible, Jameleon was designed with a plug-in model. Currently, there are five plug-ins offered:

- A **JUnit plug-in** can be used to test at a white box level. All JUnit function tags can be used in conjunction with other plug-ins.
- A **Jiffie plug-in** drives Internet Explorer and can therefore only be run on Windows. Most testers like this plug-in the most.
- An **HtmlUnit plug-in** emulates a browser and supports JavaScript quite well. Because it emulates a browser, tests written in this plug-in can be executed on any OS supported by Java.
- A **Selenium Plug-in** drives the most popular browsers. tests written in this plug-in can execute on the most popular OSes. See the selenium site for more in-depth information about Selenium-RC.
- An **HttpUnit plug-in** emulates a browser, but doesn't have very good JavaScript support. Developers like this plug-in the most. Because it emulates a browser, tests written in this plug-in can execute on any OS supported by Java.
- A **3270 (Jagacy) plug-in** is used to automate mainframe applications.
- A **jWebUnit plug-in** is the most basic among the provided plug-ins and currently has no generic tags other than a session tag.

Jameleon is an engine, not a specific solution, so the same architecture and scripting language can be used for almost any testing problem. Jameleon offers a macro language to make scripting easier, but it is designed from the beginning with the assumption that the macro language will be extended to meet each organization's specific testing needs.

Because Jameleon is written to open standards such as Java and XML, it gives test script developers full control by its ability to interface with code written in any language. The automated scripts themselves are simply a series of keywords that represent complete actions or "functional points" which are, in fact, components that are often custom-written for the application.

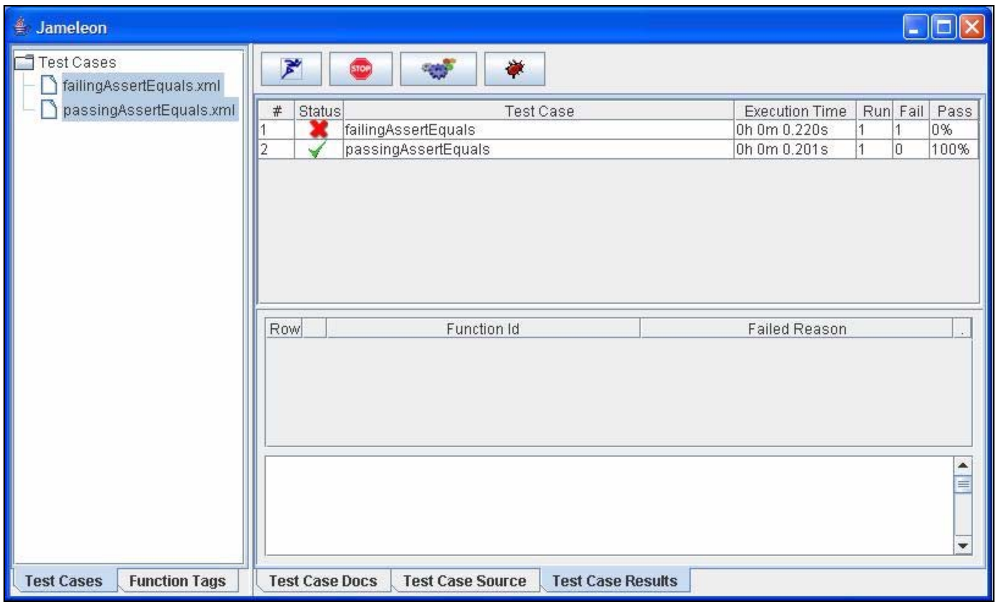
Currently, Jameleon comes with plug-ins that offer ready-made functional points for Java, web applications and mainframes; but it is architected to be extensible to permit the community to add support for other technologies.



Tag Name(s): <ju-assert-equals/>
Type: action
Author:
Use: Performs an assertEquals on two strings.
Steps: 1. Compare the expected value against the actual value.
Atributes:

Variable Name	Description	Type	Required	Default ...
expected	The expected value	java.lang.Obj...	☐	
actual	The actual value	java.lang.Obj...	☐	
msg	The error message to display if the tes...	java.lang.String	☐	
functionId	The id of this function. This is used in t...	java.lang.String	✓	
breakPoint	Tells the GUI to stop at this functional ...	boolean	☐	
precondition	Specifies this is a precondition	boolean	☐	
postcondition	Specifies this is a postcondition and w...	boolean	☐	
expectFailure	Specifies a failure is expected.	boolean	☐	
expectExcepti...	Specifies an exception is expected.	boolean	☐	
functionDelay	Sets the delay time to occur at the end ...	long	☐	

View available tags and their corresponding attributes



#	Status	Test Case	Execution Time	Run	Fail	Pass
1	✗	failingAssertEquals	0h 0m 0.220s	1	1	0%
2	✓	passingAssertEquals	0h 0m 0.201s	1	0	100%

Jameleon Test Case Results

Jameleon allows test-scripts to be data-driven from any external data source. Currently, there is support for CSV (Comma Separated Values) and JDBC (Java Database Connectivity). This allows one feature to be tested by one script, but with multiple values being passed for each row in the data file. A test can then be defined as one execution for each row of the data source or as one execution for all rows of the data source.

Even though it would be possible to write unit tests using Jameleon, Jameleon was designed with integration, regression, functional, and acceptance-level testing in mind.

5.11 Comparison Matrix

We compared the products according to estimation criteria (described in section "Estimation Criteria") and created the comparison matrixes for each group of comparison parameters (see Appendix) to facilitate the analysis.

Appendix

General Test Automation Parameter Matrix

Tool name	SLAMD	Silk Test	Silk Performer	MaxQ	Abbot	Jmeter	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Manual Tester	Rational Performance Tester	Rational Robot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
Tool URL	http://www.slamd.com/	http://www.borland.com/us/products/silk/silktest/index.html	http://www.borland.com/us/products/silk/silkperformer/index.html	http://maxq.tigris.org/	http://abbot.sourceforge.net/	http://jakarta.apache.org/jmeter/	http://jameleon.sourceforge.net/index.html	http://www.mercury.com/us/products/quality-center/functional-testing/quicktest-professional/	http://www.mercury.com/us/products/quality-center/functional-testing/winrunner/	http://www.mercury.com/us/products/performance-center/loadrunner/	http://www-306.ibm.com/software/awdtools/tester/functional/features/index.html	http://www-306.ibm.com/software/awdtools/tester/manual/index.html	http://www-306.ibm.com/software/awdtools/tester/performance/index.html	http://www-306.ibm.com/software/awdtools/tester/robot/index.html	http://www.parasoft.com/jsp/products/home.jsp?product=Jtest	http://www.pushtotest.com/	http://www.adventnet.com/products/qengine/	http://www.ivalidator.org/ivalidator/en/index.html
Test Design tools																		
"Key-Word / "Test Plan Driven" Method	Yes	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Method based on	No	Yes	Yes

Tool name	SLAMD	Silk Test	Silk Performer	MaxQ	Abbot	Jmeter	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Manual Tester	Rational Performance Tester	Rational Robot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
															"template" test cases			
Record / Playback Method	No	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes	No
Functional Testing	Yes	Yes	Possible	Yes	Yes	No	Yes	Yes	Yes		Yes	Yes	No	Yes	Yes	Yes	Yes	No
High-level Test Flows	Using Job Groups and complex scripts	N/A	N/A	N/A	N/A	Seperate thread groups	Using sessions	Yes	N/A	N/A				Yes	Yes	N/A	N/A	Using Flows
Scripting	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes	No
Scripting language	Proprietary and BSF compatible	Proprietary OOP	Proprietary Non-OOP	Jython	XML +Java		Jelly	VBScript	TSL (Testing Script Language)	Java	Own, Java, C	Own	Own	own	No	Jyphon	Jyphon	
Support for Java BSF	Yes	No	No	N/A	N/A			No	No	No					N/A	N/A	N/A	
Accesses to	Yes	No	No	No	No			N/A	N/A	N/A	Yes	Yes	Yes	Yes	N/A	N/A	N/A	

Tool name	SLAMD	Silk Test	Silk Performer	MaxQ	Abbot	Jmeter	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Manual Tester	Rational Performance Tester	Rational Robot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
Test Repository																		
Statistics and reporting functions	Yes	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	N/A	N/A	
Test Documentation	Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes	No	N/A	No
Test Plan Support	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/A	No	N/A	Yes
Parameterization and Reuse	Yes	Yes	Yes	Possible	Possible	Yes	Yes	Yes	N/A	Yes	Yes	Yes	Yes	Yes	Yes, with “Test Configuration”	Yes	Yes	Yes
Test Scheduling	Yes	No	Yes	No	No	Yes	No	No	No	Yes	Yes	No	Yes	Yes		No	Yes	No
Automated periodic runs	Yes	No	Yes	No	No	No	No	N/A	N/A	Yes	Yes	No	Yes	Yes	No	No	Yes	
Every time application under test changes	No	No	No	No	No	No	No				Yes	No	N/A	N/A	Yes	No	N/A	
Interface																		
WEB-based	Yes	No	No	No	N/A	No	No	N/A	N/A	Yes	No	No	No	No	N/A	No	Yes	No

Tool name	SLAMD	Silk Test	Silk Performer	MaxQ	Abbot	Jmeter	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Manual Tester	Rational Performance Tester	Rational Robot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
Command line	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	No	No	No	Yes	Yes	N/A	No
Support of Test Repository	Yes, using Berkley DB	No	No	No	No	XML	XML				Yes	Yes	Yes	Yes	N/A	No	Yes	XML
Types of objects stored in Test Repository	Test Jobs, Test Plans, Test Job Statistics	N/A	N/A	N/A	N/A		Test Jobs, Test Plans								N/A	N/A	N/A	Test scenarios, test parameters and test data.
Integration with application build process	No	N/A	N/A	N/A	N/A		Yes					No			Yes	No	N/A	No
Support of Version Control, Systems	No	Yes, Star Team	Yes, StarTeam	No	No	No	No				Yes	No			N/A	No	N/A	No
Supported protocols for remote testing	HTTP/S, JDBC, LDAP, SMTP, POP,	N/A	HTTP(S)/HTML,Macromedia Flash,	HTTP	N/A	HTTP/S, JDBC, LDAP,	HTTP/S, Plug-in model			Internet Messaging (IMAP),	HTTP/S	HTTP/S	HTTP/S	HTTP/S	N/A	HTTP, HTTPS, SOAP, XML-RPC, SMTP,	HTTP, HTTPS, SOAP, WSDL, AJAX	No

Tool name	SLAMD	Silk Test	Silk Performer	MaxQ	Abbot	Jmeter	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Manual Tester	Rational Performance Tester	Rational Robot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
	IMAP, FTP		WebDAV,SMTP/POP,IMAP,MAPI,FTP, Streaming Media (MS, Real), RSS,SSL,LDAP,TCP/IP based applications,UDP based applications,WAP2 (including WTLS) Multimedia Message Service			SOAP, JMS				POP3, SMTP, FTP, Palm, SOAP, Web (HTTP/HTML), Web Services, Multimedia Message Service (MMS), WinSock\Web Dual Protocol , i-Mode, VoiceXML						POP3, IMAP		

Tool name	SLAMD	Silk Test	Silk Performer	MaxQ	Abbot	Jmeter	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Manual Tester	Rational Performance Tester	Rational Robot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
			(MMS), RADIUS,i-Mode															
Supported Testing Environments																		
J2EE	yes	No	No	No	No	No	No	Yes	Yes	Yes					Yes	Yes	N/A	
WebLogic Web Server Monitoring	No	No	Yes	No	No	No	No	No	No	Yes						No	N/A	
JBoss Web Server Monitoring	No	No	Yes	No	No	No	No	No	No	Yes						No	N/A	
Web Services	No	No	Yes	No	No	No	No	Yes	Yes	Yes						Yes	Yes	
.NET	No	No	Yes	No	No	No	No	Yes	Yes	Yes						Yes	N/A	
AJAX	No	No	N/A	No	No	No	No	No	No	No						No	Yes	
Other	LDAP servers	No	IE6 SP1/SP2,Firefox 1.5,AWT,Swing,SWT,Symantec Visual	No	AWT, Swing, SWT	No		IE6, Firefox 1.5,AWT, Symantec Visual Café	IE6, AWT, Symantec Visual Café						EJB, JDBC, JSP, Hibernate, Java EE, ME / J2ME, Java		Internet Explorer 6, Firefox 1.5.x and Mozilla 1.7.x, Windows XP,	

Tool name	SLAMD	Silk Test	Silk Performer	MaxQ	Abbot	Jmeter	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Manual Tester	Rational Performance Tester	Rational Robot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
			Café,. Net WinForms,Infragistics NetAdvantage WinForms 5.2, 5.3 and 6.1,Win32®, MFC,VisualBasic® 6 / ActiveX®,Motif (on Solaris and Linux)												Beans, Servlets, Struts		2000 and Linux	
Statistics and Reporting tools																		
Comparison reports	no	No	No	No	No	no	No	No	No	Yes	yes	no	yes	Yes	No	N/A	N/A	No
Graphical charts	yes	No	Yes	No	No	yes	No	No	No	Yes		no	yes	Yes	No	Yes	Yes	No
Test Data export																		No

Tool name	SLAMD	Silk Test	Silk Performer	MaxQ	Abbot	Jmeter	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Manual Tester	Rational Performance Tester	Rational Robot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
To text files: tab or comma delimited, XML files	yes	No	No	No	N/A		Yes	Yes	No	No	yes	yes	yes	Yes	N/A	N/A	N/A	
To MS Excel, Statistica, Mathcad, etc.	no	No	No	No	No			No	No	No			yes	Yes	N/A	N/A	Excel	
Integration with QA tools	N/A	Yes, Silk Central	Yes, SilkCentral	No	No	No	No	Yes	Yes	Yes	yes	yes	yes	Yes	N/A	N/A	Built-in Issue Tracker	No
Integration with CRM/ERP Systems	N/A	No	No	No	No	No	No	Yes	No	Yes					N/A	N/A	N/A	No
Team Work	N/A	N/A	N/A	No	No	No	No	Yes	Yes	N/A	yes	yes	yes	Yes	Yes	N/A	Yes	No

Tool name	SLAMD	Silk Test	Silk Performer	MaxQ	Abbot	Jmeter	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Manual Tester	Rational Performance Tester	Rational Robot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
Learning the tool	Easy	Medium	Medium	Easy	Medium	Easy	Medium	Easy	Easy	Medium	Hard	Easy	Medium	Medium		N/A	N/A	Easy
Price	Free	N/A	N/A	Free	Free	Free	Free	N/A	N/A	N/A	High	High	High	Big		Free, commercial TestNetwork starts from $10k	Starting from $799	Free

Regression / Unit / Integration Testing Parameter Matrix

Tool name	SLAMD	SilkTest	MaxQ	Abbot	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Robot	Abbot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
Tool URL	http://www.slamd.com/	http://www.borland.com/us/products/silk/silktest/index.html	http://maxq.tigris.org/	http://abbot.sourceforge.net/	http://jameleon.sourceforge.net/index.html	http://www.mercury.com/us/products/quality-center/functional-testing/quicktest-professional/	http://www.mercury.com/us/products/quality-center/functional-testing/winrunner/	http://www.mercury.com/us/products/performance-center/loadrunner/	http://www-306.ibm.com/software/awdtools/tester/functional/features/index.html	http://www-306.ibm.com/software/awdtools/tester/robot/index.html	http://abbot.sourceforge.net/	http://www.parasoft.com/jsp/products/home.jsp?product=Jtest	http://www.pushtotest.com/	http://www.adventnet.com/products/qengine/	http://www.ivalidator.org/ivalidator/en/index.html
Support of complex test scenarios	Yes	Yes	No		Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	N/A	N/A	Yes
Hierarchy of test suits and test parameters	No	Yes	No		Yes	No	No	No	Yes	Yes	No		N/A	N/A	Yes

Tool name	SLAMD	SilkTest	MaxQ	Abbot	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Robot	Abbot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
Concurrent execution of hierarchic and sequential test structures	Yes, Sequential Job Groups	Yes	No		Yes				Yes	Yes	No		N/A	N/A	Yes
Conditional test behavior	Yes	No	Yes		No	No	No	Yes	Yes	Yes	Yes		Yes	Yes	Yes
Access to complete test context and test history	Yes	No	No		Yes	Yes	No	Yes	Yes	Yes	No		No	N/A	Yes
Reusability and parameterization									Yes						
Flexible parameterization of test setups, checkups, and teardowns.	Yes	Yes	Yes		Yes				Yes	Yes	Yes		Yes	Yes	Yes
Parameters in XML	No	No	No		Yes				Yes	Yes	Yes		N/A	Yes	Yes
Supported test levels															

Tool name	SLAMD	SilkTest	MaxQ	Abbot	Jameleon	HP Mercury QTP	HP Mercury WinRunner	HP Mercury LoadRunner	Rational Functional Tester	Rational Robot	Abbot	Parasoft JTest	PushToTest TestMaker 4.2	AdventNet QEngine 6.7	iValidator 2.2.0
Developer tests	Yes	No	No		Yes				Yes	No	No		N/A	N/A	Yes
Integration test	Yes	No	No		Yes				Yes		No		N/A	N/A	No
Acceptance test	Yes	No	No		Yes				Yes		No		N/A	N/A	No
JUnit Support	N/A	No	No		Yes	No	No	No	Yes	No	Yes		Yes	N/A	Yes
IDE Plug-in	No	No	No		Yes				Yes	No	No		No	N/A	Yes

Load/Stress Testing Parameter Matrix

Tool name	SLAMD	SilkTest	SilkPerformer	MaxQ	Abbot	JMeter	HP Mercury LoadRunner	Rational Performance Tester	Rational Robot	PushToTest TestMaker 4.2	AdventNet QEngine 6.7
Tool URL	http://www.slamd.com/	http://www.borland.com/us/products/silk/silktest/index.html	http://www.borland.com/us/products/silk/silkperformer/index.html	http://maxq.tigris.org/	http://abbot.sourceforge.net/	http://jakarta.apache.org/jmeter/	http://www.mercury.com/us/products/performance-center/loadrunner/	http://www-306.ibm.com/software/awdtools/tester/performance/index.html	http://www-306.ibm.com/software/awdtools/tester/robot/index.html	http://www.pushtotest.com/	http://www.adventnet.com/products/qengine/
Flexible job scheduler	Yes	Yes	Yes	No		Yes	N/A	Yes	Yes	N/A	Yes
Statistics	Yes	Yes	Yes	No		Yes	N/A	Yes	Yes	Yes	Yes
Trackers	Yes	N/A	N/A	N/A			N/A	Yes	Yes	No in free product, Yes in commercial TestNetwork	N/A
In-progress results	Yes	Yes	Yes	No		Yes	N/A	Yes	No	N/A	N/A
Distributed testing	Yes	Yes	Yes	No		Yes	Yes	Yes	Yes	No in free product, Yes in commercial TestNetwork	Yes

Tool name	SLAMD	SilkTest	SilkPerformer	MaxQ	Abbot	JMeter	HP Mercury LoadRunner	Rational Performance Tester	Rational Robot	PushToTest TestMaker 4.2	AdventNet QEngine 6.7
Centralized control of test jobs	Yes	Yes	Yes	No		Yes	N/A	Yes	Yes	No in free product, Yes in commercial TestNetwork	Yes
Multiple concurrent clients	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes	Yes
Self-optimizing jobs	Yes	N/A	N/A	N/A		No	N/A	Yes	No	N/A	N/A
Security and access control	Yes (SSL)	N/A	N/A	No		No	N/A	Yes (SSL)	Yes (SSL)	N/A	Yes
System Resource monitoring	Yes	Yes	Yes	No		No	Yes	Yes	Yes	No in free product, Yes in commercial TestMaker Framework	Yes
Extensibility	Yes	Yes	Yes	Yes		Yes	N/A	Yes	Yes	Yes	Yes