Performance Testing Uncovered

First Presented at:

NobleStar Systems Corp.

London, UK 26 Sept. 2003

Scott Barber

Chief Technology Officer

PerfTestPlus, Inc.

Performance Testing Uncovered

Agenda:

ROI of Performance Testing - Outsource vs. Insource

What is Effective Performance Testing?

Rational and Mercury – a Load Generation Tool Comparison

ROI of Performance Testing

Costs and Comparisons
- Insource
- Outsource

Why Performance Test?

ROI Justification
- Benefits
- Logical
- Financial
- Insurance

Costs and Comparisons

Insource

- Regular testing over years.
- Typically multiple development efforts/applications.
- Must maintain tools, expertise, and environments.
- High start-up costs, plus recurring costs.

Outsource

- Irregular or infrequent testing.
- Typically single efforts/applications.
- Minimal recurring costs.
- Isolated expense.

Costs and Comparisons

Insource

Start-up Costs

- Tools: U.S. $25K to $250K+ (based on tool and number of simulated users)
- Equipment: $25K to $250 K (based on environment)
- Performance testers: $75K to $150K
- Training: $25K+ first year

Total Costs

- $150k to $675k+ first year
- $75K to $250K+ following years

Costs and Comparisons

Outsource

Costs / Duration

- Tool lease: U.S. $5K to $100K+ (20-40% of purchase)
- Equipment lease: $5K to $100K+ (20-40% of purchase)
- Performance testers: $100 to $250 per hour
- 6 weeks to 9 months

One time costs

- $25K to $575K+

Fixed fee ‘snapshots’

- $5k – 25K

Why Performance Test?

Speed - Does the application respond quickly enough for the intended users?

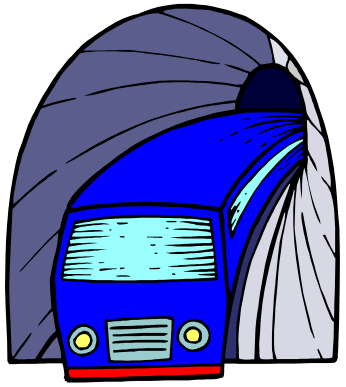
Scalability – Will the application handle the expected user load and beyond? (AKA Capacity)

Stability – Is the application stable under expected and unexpected user loads? (AKA Robustness)

Confidence – Are you sure that users will have a positive experience on go-live day?

Why Performance Test?

Speed



User Expectations
- Experience
- Psychology
- Usage

System Constraints
- Hardware
- Network
- Software

Costs
- Speed can be expensive!

Why Performance Test?

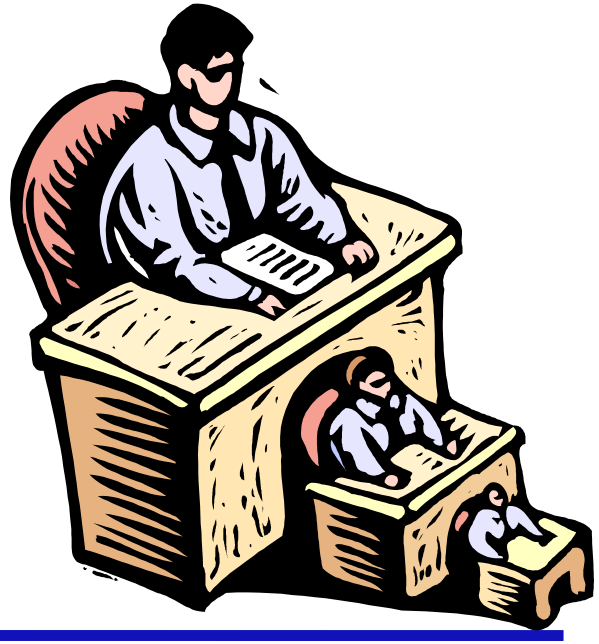
Scalability

How many users…

- before it gets “slow”?
- before it stops working?
- will it sustain?
- do I expect today?
- do I expect before the next upgrade?

How much data can it hold?

- Database capacity
- File Server capacity
- Back-up Server capacity
- Data growth rates



Why Performance Test?

Stability

What happens if…

- there are more users than we expect?
- all the users do the same thing?
- a user gets disconnected?
- there is a Denial of Service Attack?
- the web server goes down?
- we get too many orders for the same thing?

Why Performance Test?

Confidence

If you know what the performance is…

- you can assess risk.
- you can make informed decisions.
- you can plan for the future.
- you can sleep the night before go-live day.

The peace of mind that it will work on go-live day alone justifies the cost of performance testing.

ROI Justification

Benefits

Application Tuning

- Find and fix performance bottlenecks and architectural defects.

Infrastructure Tuning

- Improve efficiency of existing hardware (improved speed and volume of responses).

Capacity Planning

- How much hardware we need?

ROI Justification

Benefits

Performance testing builds confidence that users will not encounter problems in live operation, and is a type of risk-mitigating insurance. It enables us to evaluate whether:

- A system is ready for release into production.
- A system is likely to meet its goals.
- The users will be satisfied with the service they receive.
- Resources are being used effectively.
- The system operation is likely to be trouble-free.

ROI Justification

Logical

Survey Results (source: Newport Group)

Activity	Experience in Production	
	OK	NOT OK
Simulated performance during design phase	21%	---
Tested early in development	35%	06%
Tested late in development	38%	26%
Did post-deployment testing	---	08%
Did not test	06%	60%

ROI Justification

Financial

The cost of Performance Testing averages ~ 2.5% of the total cost of development.

Fixing poorly performing applications after they are released into production averages ~ 25% of the total cost of development… If they can be fixed at all!!

ROI Justification

Insurance

Testing isn't really an investment, it's insurance.

- ROI for insurance is meaningless without a risk-assessment.
- The best financial option for insurance without adjusting for risk, is not to insure at all.

Traditional example:

- $1,000,000 life insurance policy.
- Cost is $25,000 per year.
- Likelihood of payout has an annual probability of 2%.
- In about 30 years we could save the $1,000,000 ourselves!
- We get more without insurance … unless catastrophe occurs before 30 years.

60% of "uninsured" applications had a "catastrophe" in first year.

Summary & Questions

Performance Testing ROI:

- Performance Testing is Expensive
- NOT Performance Testing is MORE Expensive
- Performance Testing is like a Performance Insurance Policy

Effective Performance Testing

What is Performance Related Testing?

Intro to Effective Performance Testing

Summary / Q&A

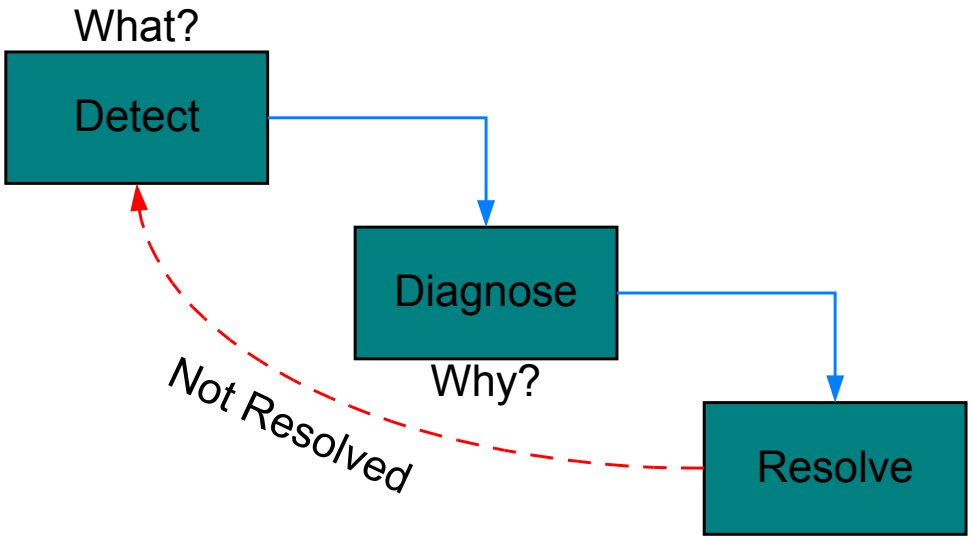
What is Performance Related Testing?

Performance Validation

Performance Testing

Performance Engineering

Compare & Contrast



Performance Validation

"Performance validation is the process by which software is tested with the intent of determining if the software meets pre-existing performance requirements. This process aims to evaluate compliance."

Primarily used for…

- determining SLA compliance.
- IV&V (Independent Validation and Verification).
- validating subsequent builds/releases.

Performance Testing

"Performance testing is the process by which software is tested to determine the current system performance. This process aims to gather information about current performance, but places no value judgments on the findings."

Primarily used for…

- determining capacity of existing systems.
- creating benchmarks for future systems.
- evaluating degradation with various loads and/or configurations.

Performance Engineering

"Performance engineering is the process by which software is tested and tuned with the intent of realizing the required performance. This process aims to optimize the most important application performance trait, user experience."

Primarily used for…

- new systems with pre-determined requirements.
- extending the capacity of old systems.
- "fixing" systems that are not meeting requirements/SLAs.

Compare and Contrast

Validation and Testing:

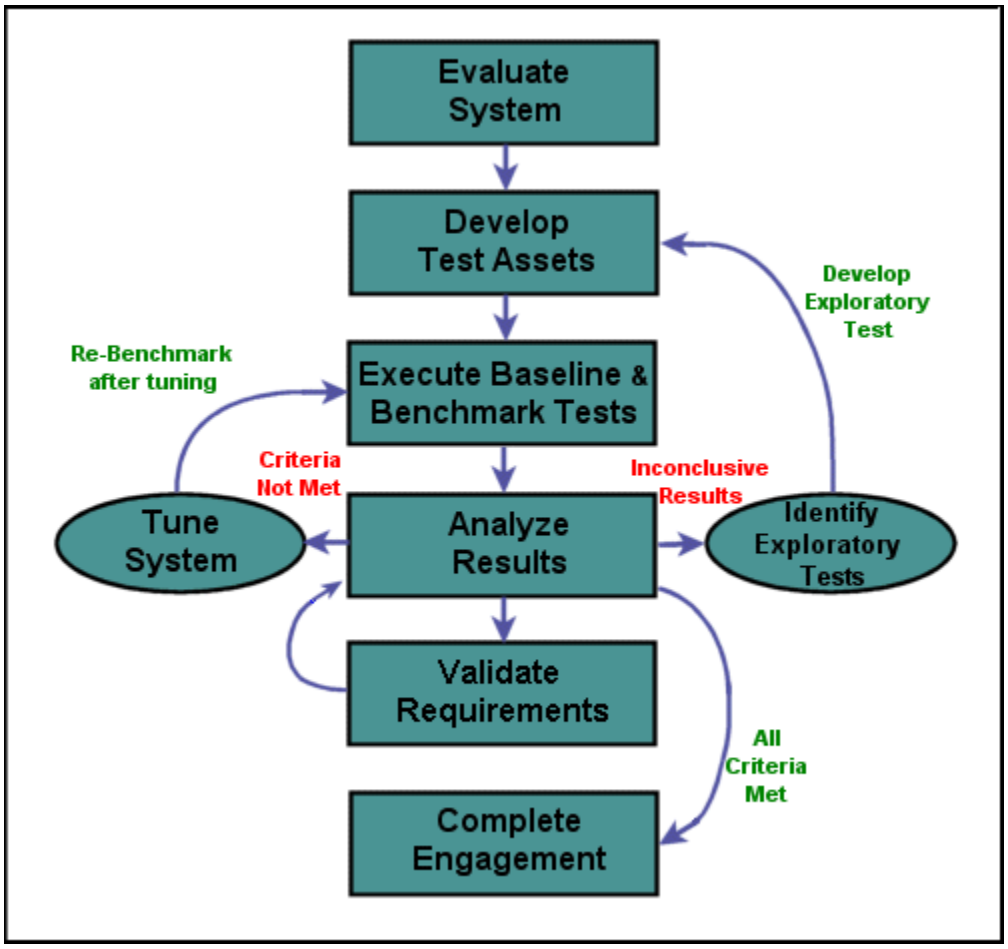
- Are a subset of Engineering.
- Are essentially the same except:
  - Validation usually focuses on a single scenario and tests against pre-determined standards.
  - Testing normally focuses on multiple scenarios with no pre-determined standards.
- Are generally not iterative.
- May be conducted separate from software development.
- Have clear end points.

Compare and Contrast

Engineering:

- Is iterative.
- Has clear goals, but ‘fuzzy’ end points.
- Includes the effort of tuning the application.
- Focuses on multiple scenarios with pre-determined standards.
- Heavily involves the development team.
- Occurs concurrently with software development.

Approach to Performance Testing



Evaluate System

Develop Test Assets

Baselines and Benchmarks

Analyze Results

Tune

Identify Exploratory Tests

Validate Requirements

Complete Engagement

Evaluate System

Determine performance requirements.

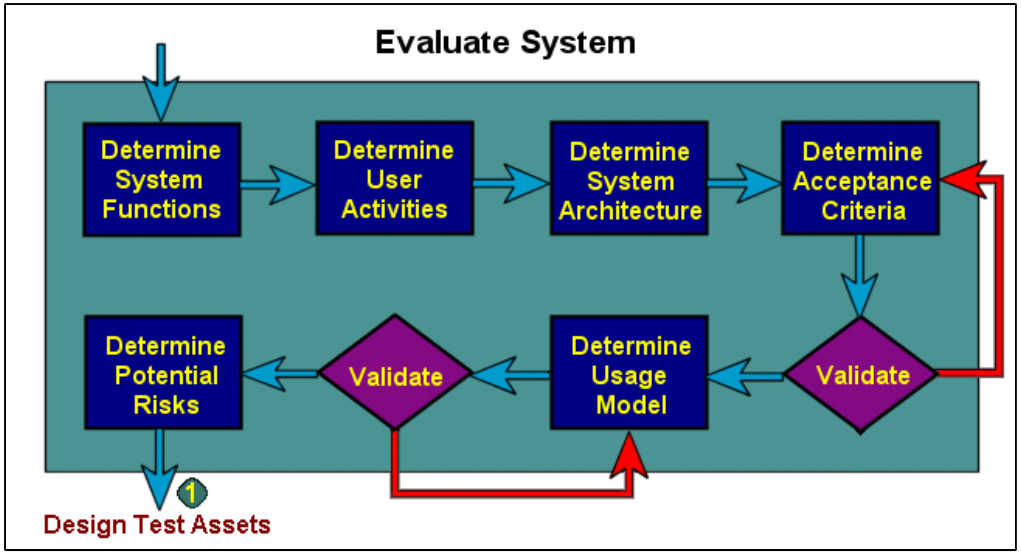
Identify expected and unexpected user activity.

Determine test and/or production architecture.

Identify non-user-initiated (batch) processes.

Identify potential user environments.

Define expected behavior during unexpected circumstances.



Develop Test Assets

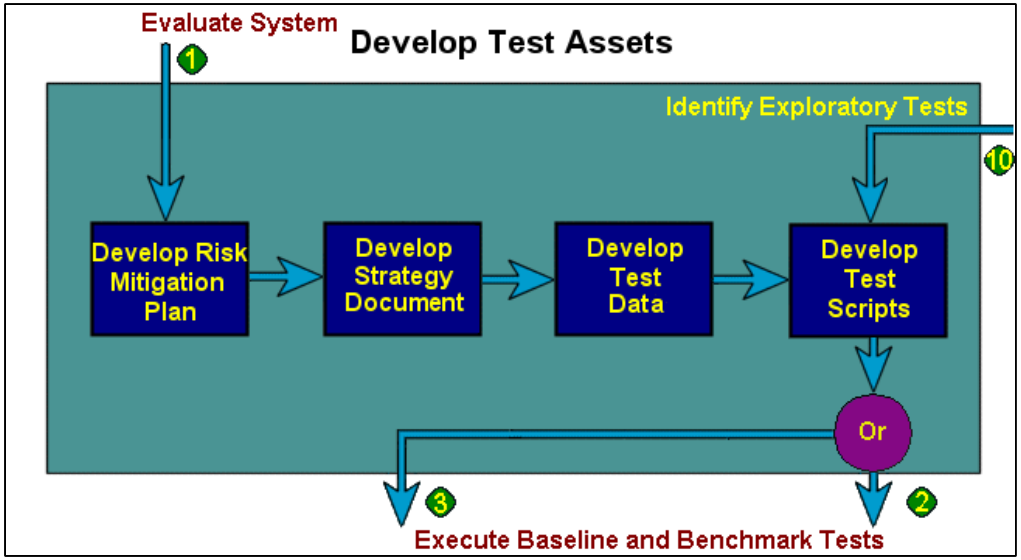
Create Strategy Document.

Develop Risk Mitigation Plan.

Develop Test Data.

Automated test scripts:

- Plan
- Create
- Validate

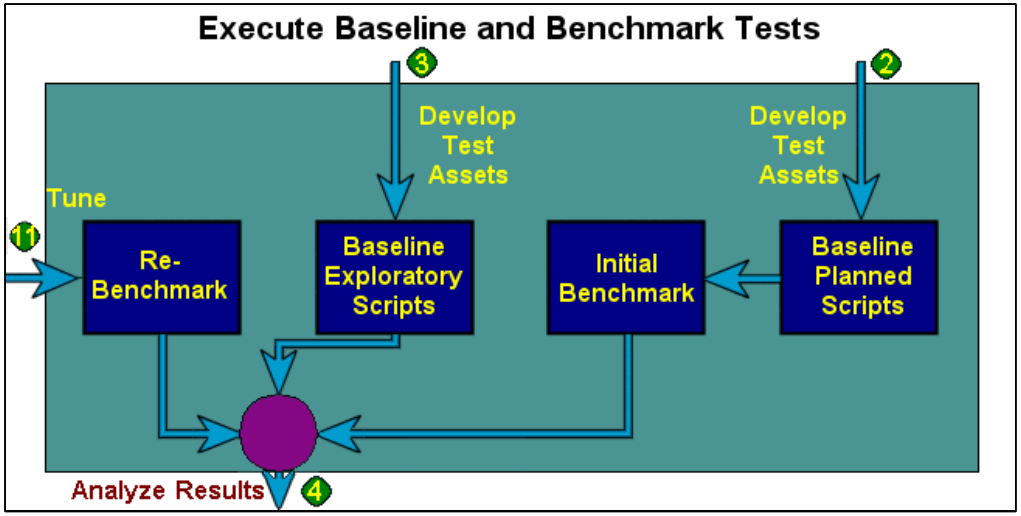


Baseline and Benchmarks

Most important for iterative testing.

Baseline (single user) for initial basis of comparison and 'best case'.

Benchmark (15-25% of expected user load) determines actual state at loads expected to meet requirements.



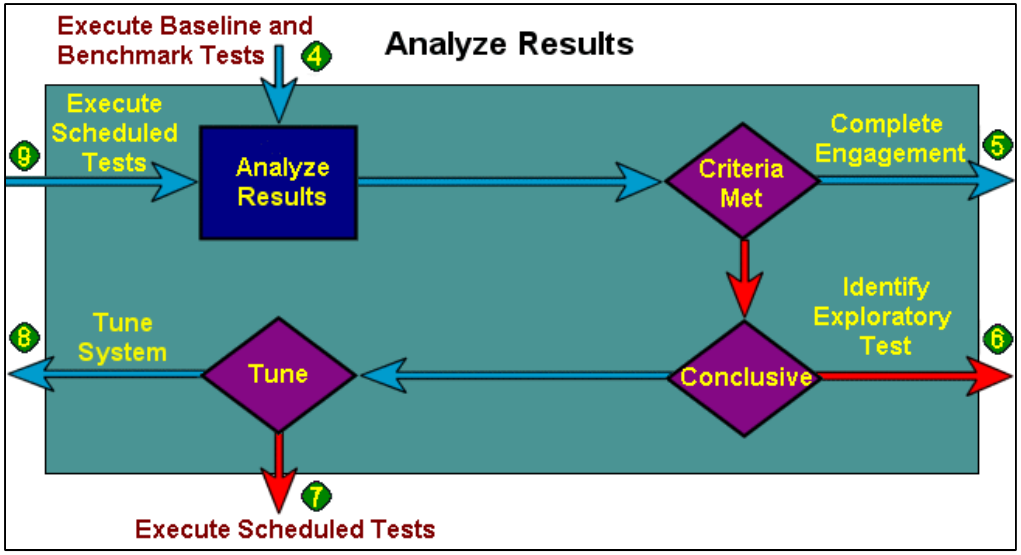
Analyze Results

Most important.

Most difficult.

Focuses on:

- Have the performance criteria been met?
- What are the bottlenecks?
- Who is responsible to fix those bottlenecks?
- Decisions.



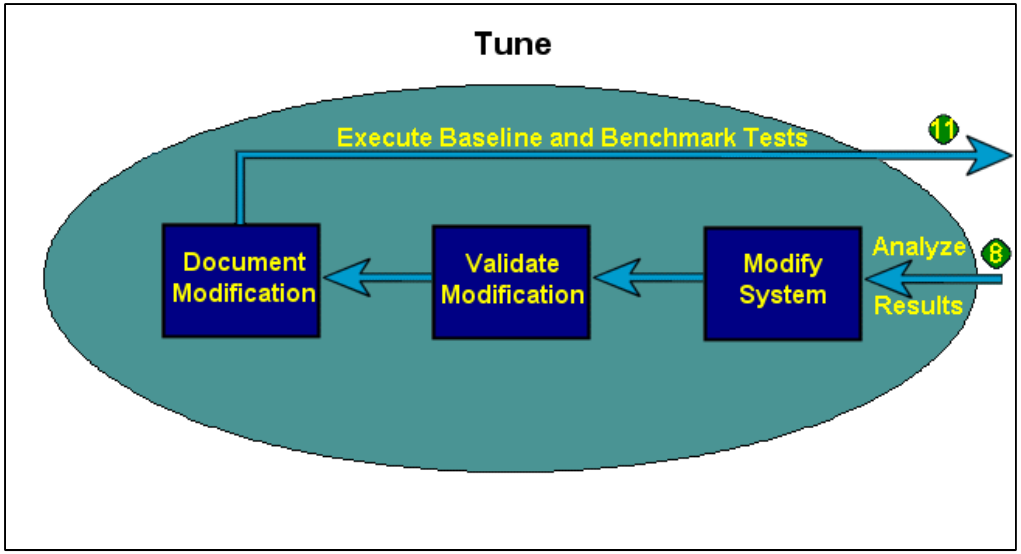
Tune

Engineering only.

Highly collaborative with development team.

Highly iterative.

Usually, performance engineer ‘supports’ and ‘validates’ while developers/admins ‘tune’.



Identify Exploratory Tests

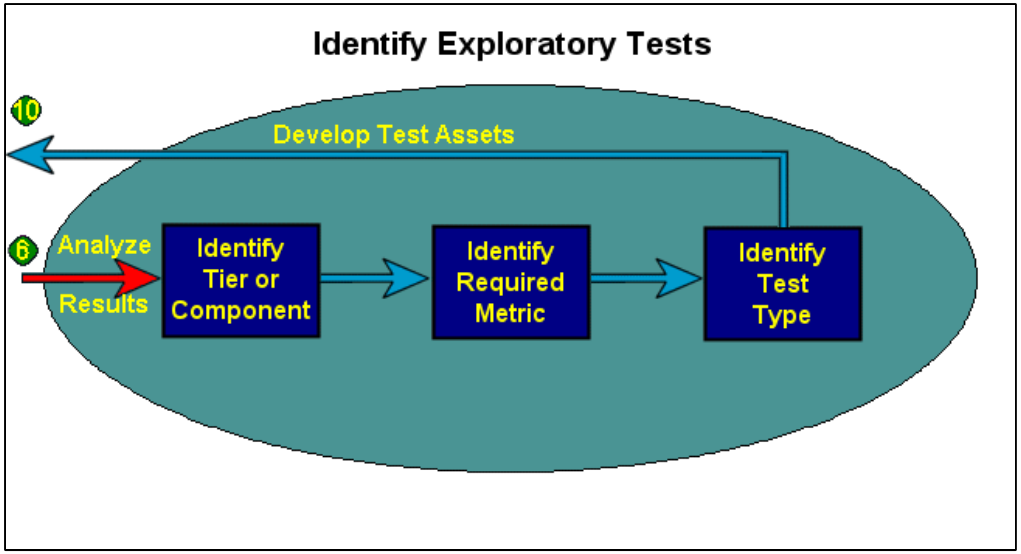
Engineering only.

Exploits known bottleneck.

Assists with analysis & tuning.

Significant collaboration with ‘tuners’.

Not robust tests – quick and dirty,not often reusable or relevant after tuning is complete.

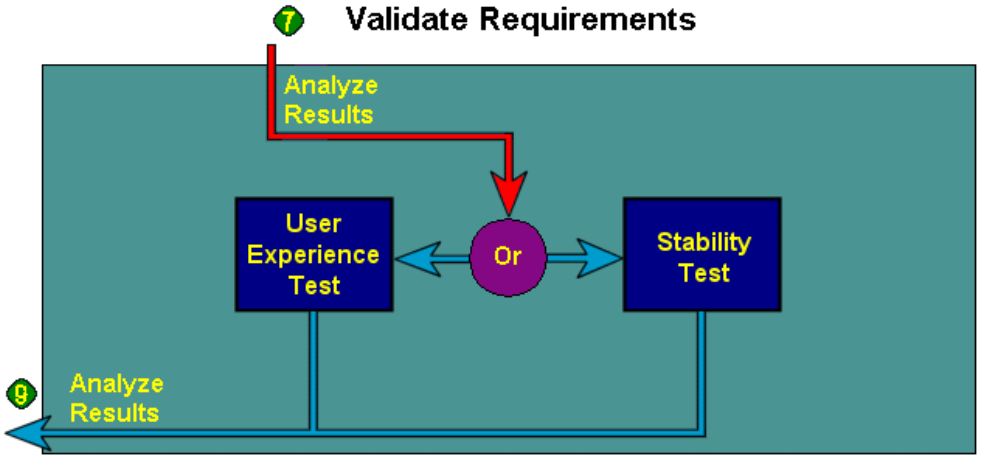


Validate Requirements

Only after Baseline and/or Benchmark tests.

These tests evaluate compliance with documented requirements.

Often are conducted on multiple hardware/configuration variations.



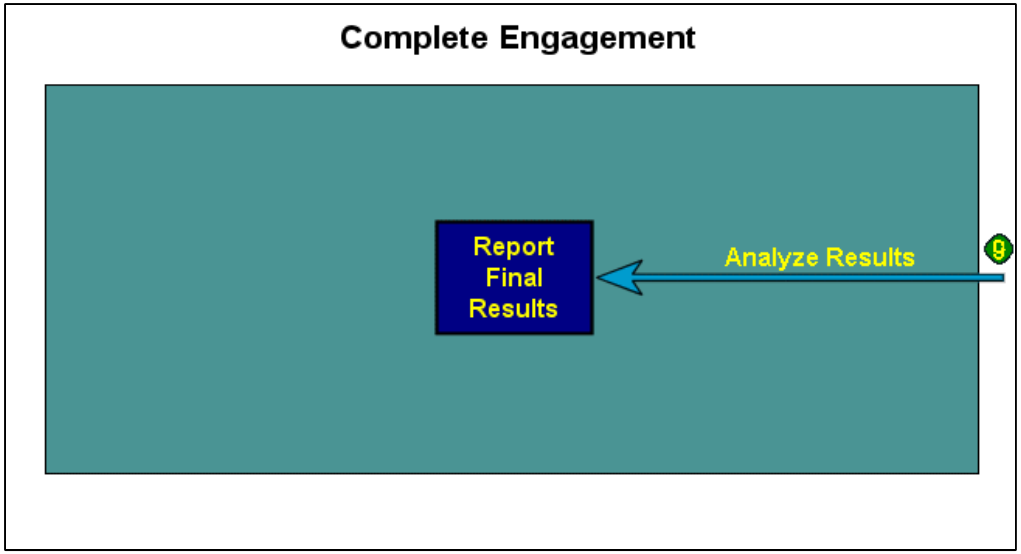
Complete Engagement

Document:

- Actual Results
- Tuning Summary
- Known bottlenecks not tuned
- Other supporting information
- Recommendation

Package Test Assets:

- Scripts
- Documents
- Test data



Summary & Questions

Performance Testing Approach:

- Ensures goals are accomplished.
- Defines tasks.
- Identifies critical decision points.
- Shortens testing lifecycle.
- Increases confidence in results.

Rational and Mercury - Comparison

Requirements Management

Application Modelling

Code/Asset Management

Defect Management

Test Asset Management

Functional Testing

Performance Testing (Focus area)

Metrics Analysis and Reporting

Rational and Mercury - Comparison

Requirements Management

Rational RequisitePro

- Uses MS Word
- Includes Traceability
- Includes Coverage Analysis
- Integrates with Rational:
  - Rose (Modeling)
  - XDE (IDE)
  - ClearQuest (Defect Tracking)
  - ClearCase (CM)
  - TestManager (Test Assets and Execution)

Mercury TestDirector

- Requirements Manager module
- Includes Traceability
- Includes Coverage Analysis
- Integrates with Mercury:
  - Test Plan (Test Plan Assets)
  - Test Lab (Test Execution)
  - Defects Manager
- Open API for other Integrations

Rational and Mercury - Comparison

Application Modeling

Rational Rose

- UML Compliant
- Includes Traceability
- Includes Coverage Analysis
- Integrates with Rational:
  - RequisitePro
  - XDE
  - ClearQuest
  - ClearCase
  - TestManager

Mercury TestDirector

- Open API for 3rd Party Integrations
- Contact Mercury Interactive for details about your preferred modeling software.

Rational and Mercury - Comparison

Code/Asset Management

Rational ClearCase

- Powerful and Popular CM tool
- UCM (Unified Change Management) for Test Assets
- Integrates with Rational:
  - Rose
  - RequisitePro
  - XDE
  - ClearQuest
  - TestManager

Mercury TestDirector

- Open API for 3rd Party Integrations
- Contact Mercury Interactive for details about your preferred modeling software.

Rational and Mercury - Comparison

Defect Management

Rational ClearQuest

- UCM to tie test results to development resources
- Includes Traceability
- Includes Analysis tools
- Customizable
- Integrates with Rational:
  - RequisitePro
  - ClearCase
  - Robot

Mercury TestDirector

- Defects Manager module
- Includes Traceability
- Includes Analysis tools
- Customizable
- Integrates with Mercury:
  - Test Plan
  - Test Lab
  - Defects Manager
- Open API for other Integrations

Rational and Mercury - Comparison

Test Asset Management

Rational TestManager

- Manages Test Results, Analysis, Manual, Automated and Performance test execution.
- UCM to tie test results to development resources
- Integrates with Rational:
  - Rose
  - RequisitePro
  - XDE
  - ClearCase
  - ClearQuest
  - Robot

Mercury TestDirector

- Manages Test Results, Analysis, Manual, Automated and Performance test execution.
- 4 modules to manage Requirements, Defects, Test Planning and Test Execution
- Integrates with all Mercury testing tools
- Open API for other Integrations

Rational and Mercury - Comparison

Automated Functional Testing

Rational (GUI) Robot / XDE Tester

- Work on many platforms/ environments
- Large user community
- Scripting based on common languages
- Integrates with Rational:
  - TestManager
  - XDE (XDE Tester only)

Mercury WinRunner / QuickTest

- Work on many platforms/ environments
- Large user community
- Scripting based on common languages
- Integrates with Mercury:
  - TestDirector
- Open API for other Integrations

Rational and Mercury - Comparison

Automated Performance Testing

Rational (VU) Robot

- Work on many platforms/ environments
- Large user community
- Most users extremely loyal
- High project success rate
- Significant learning curve
- Scripting based on C
- Misleading results if tool improperly used
- Integrates with Rational:
  - TestManager

Mercury LoadRunner

- Work on many platforms/ environments
- Large user community
- Most users extremely loyal
- High project success rate
- Significant learning curve
- Scripting based on C
- Misleading results if tool improperly used
- Integrates with Mercury:
  - TestDirector

Rational and Mercury - Comparison

Performance Testing – Test Management/Planning

Robot

- Tight integration with TestManager (delivered coupled)
- Loose integration with ReqPro, ClearQuest and ClearCase
- Most 3rd party integrations not officially supported
- File/asset storage must be in a Rational Project

LoadRunner

- Loose integration with TestDirector (delivered un-coupled)
- Many 3rd party integrations supported
- File/asset storage may be configured in a custom manner

Each tool supports different management/planning approaches differently. "Better fit" based entirely on your specific needs.

Rational and Mercury - Comparison

Performance Testing – Test Automation

Robot

- Can script against unsupported protocols
- Multiple script capture methods options
- Scripts generally longer
- Strong support for manual (custom code) data correlation
- Can generate test data

LoadRunner

- Officially supports more protocols
- Strong automatic (GUI based) data correlation
- Visual IDE
- Language supports data structures
- Can import test data from existing database

The process of test automation and script editing is quite different between the two tools. Most people who have used both have a strong preference, but the split is pretty even.

Rational and Mercury - Comparison

Performance Testing – Test Scheduling

Robot

- Extremely intuitive interface
- Easy to modify suites from one test to the next
- GUI tools for script/user/group pacing.
- Interface makes suites (schedules) essentially self-documenting

LoadRunner

- Much scheduling done at script level
- Allows adding or removing virtual users during test execution
- Abstracts playback behavior for easy modification
- Allows performance goals to be added as a schedule parameter

The process of test scheduling is significantly different between the two tools, but both can be used to create both simple and complex schedules.

Rational and Mercury - Comparison

Performance Testing – Test Execution

Robot

- Good real-time script code and activity viewing and reporting
- Handles cookies explicitly

LoadRunner

- Playback as a thread or process
- Has real-time script execution and response time viewing

Both tools allow for collection of resource counters on remote machines and view them real-time. With Rational it is easy to determine what the scripts are doing at runtime. With Mercury, it is easy to see what a specific virtual user is doing during runtime.

Rational and Mercury - Comparison

Performance Testing – Test Analysis

Robot

- Reports part of TestManager
- Short learning curve on creating and viewing charts and reports

LoadRunner

- Reporting interface is a separate tool with it's own learning curve
- Huge number of options for custom charts and reports

Most expert performance testers create their own charts and graphs, regardless of the tool they are using for detailed analysis and only use the reporting capability of the tool for initial analysis. Both of these tools are fully adequate for this purpose.

Rational and Mercury - Comparison

Metrics Analysis and Reporting

Rational TestManager

- Many pre-built reports
- Customizable reports
- Can export data to create external spreadsheets

Mercury TestDirector

- Separate reporting interface
- Many pre-built reports
- Customizable reports
- Easy to export data to create external spreadsheets

Summary

Both tools satisfy most clients most of the time.

Neither tool is “better”. Some projects/organizations “fit better” with one tool or the other.

Selecting the correct tool is important.

Knowing how to use the tool you have/hiring experts is MORE important.

Where to go for more Information

http://www.PerfTestPlus.com (My site)

http://www.QAForums.com (Huge QA Forum)

http://www.loadtester.com (Good articles and links)

http://www.segue.com/html/s_solutions/papers/s_wp_info.htm (Good articles and statistics)

http://www.keynote.com/resources/resource_library.html (Good articles and statistics)

Questions and Contact Information

Scott Barber

Chief Technology Officer

PerfTestPlus, Inc

E-mail:

sbarber@perftestplus.com

Web Site:

www.PerfTestPlus.com