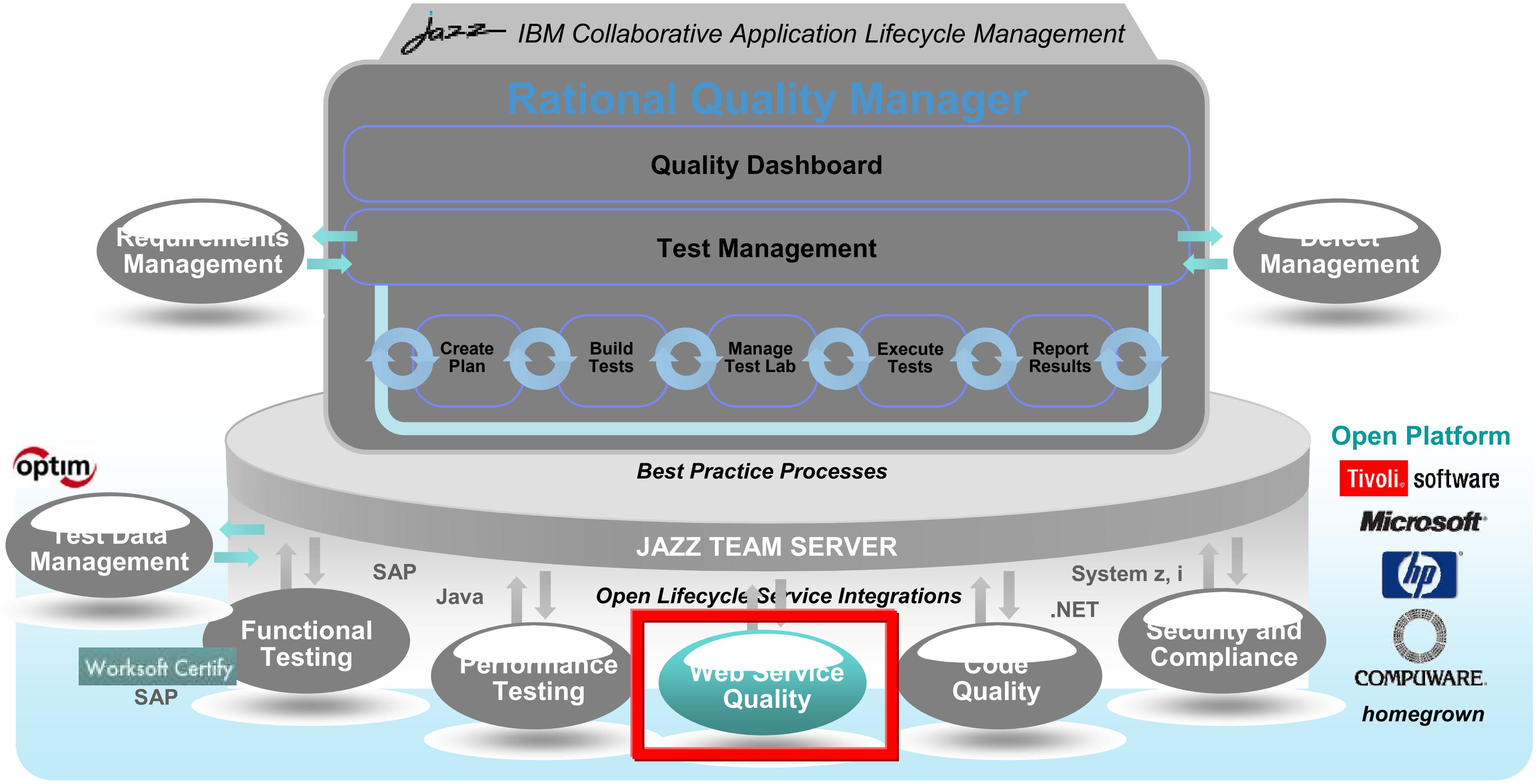
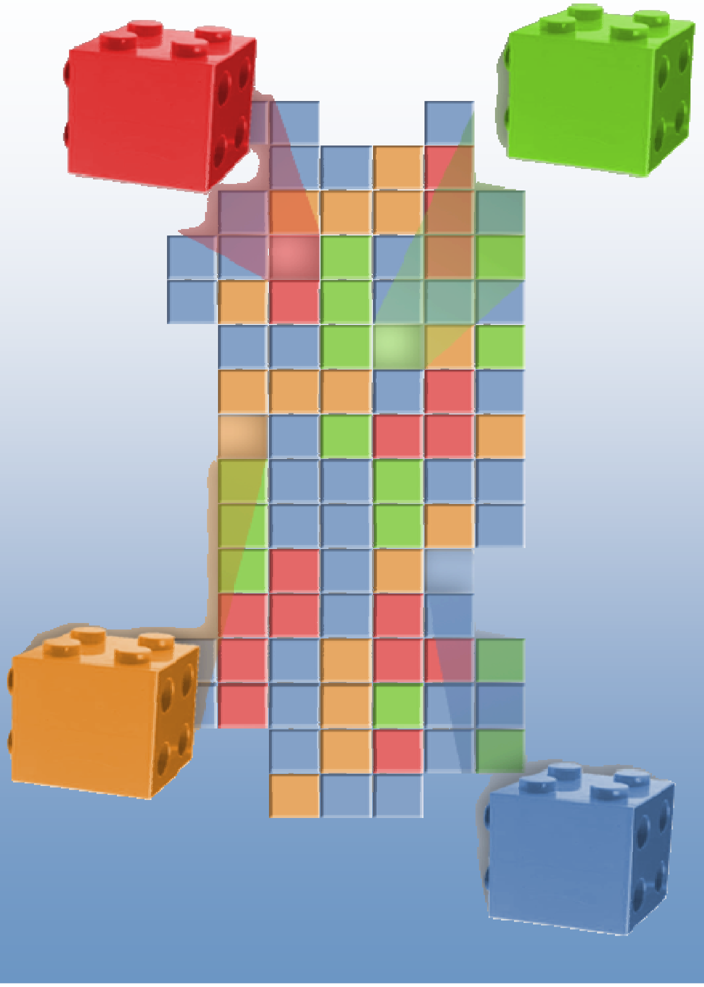
Service Testing
Rational Service Tester

Smarter software for a smarter planet
smarter planet
smarter software
Smarter software for a smarter pl
marter software
IBM
smarter planet
Software and Systems Engineering

Centralized test management offering allowing full lifecycle support across all types of testing and platforms



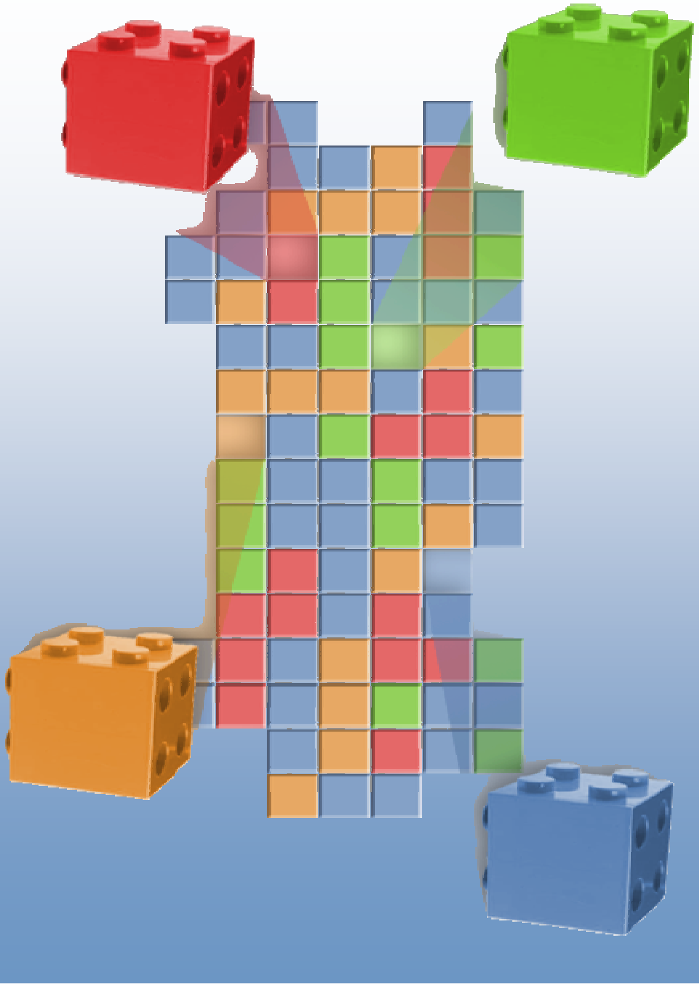
SOA: Service Oriented Architecture Definitions



To the IT Executive

Flexible applications built upon **re-usable** building blocks that are **easily connected**

SOA: Service Oriented Architecture Definitions



To the IT Executive

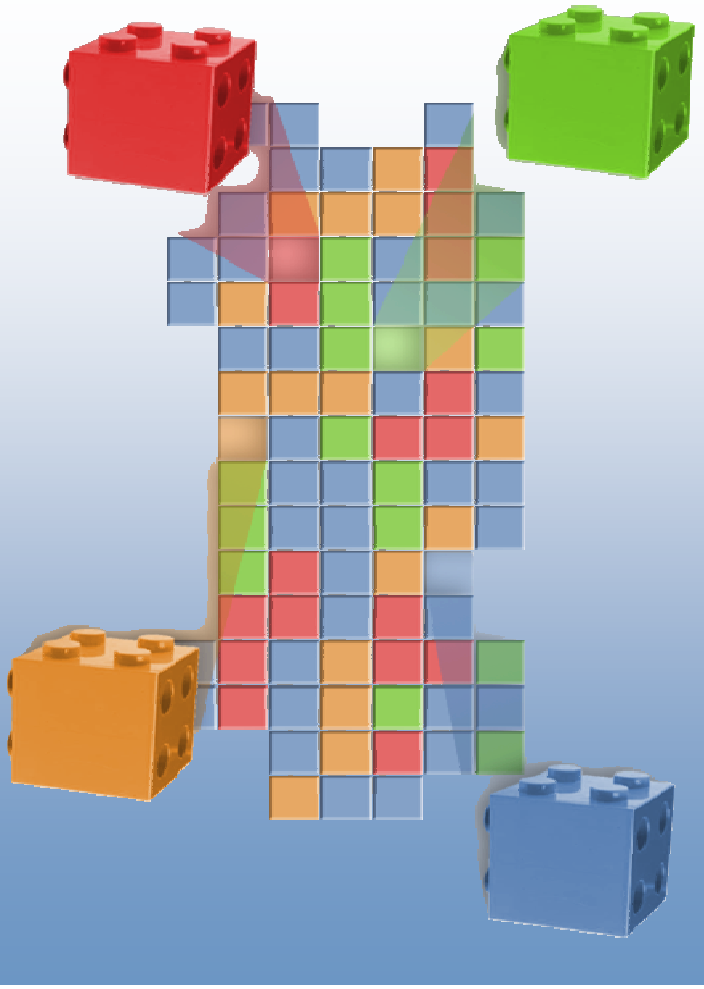
Flexible applications built upon **re-usable** building blocks that are **easily connected**

To the Developers and Testers

Web Services.

Period.

SOA: Service Oriented Architecture Definitions



To the IT Executive

Flexible applications built upon **re-usable** building blocks that are **easily connected**

To the Software Architect

An IT **architectural style** which assembles loosely coupled distributed services to implement a business process

To the Developers and Testers

Web Services.

SOA: Implications for Quality Management

To the IT Executive

Flexible applications built upon **re-usable** building blocks that are **easily connected**

Validate Business Process

Challenges

- **Identifying test cases**
- **Managing Data Complexity**
  - **Requirements, Test Cases, Defects**
- **Ensuring optimal test & configuration coverage**

SOA: Implications for Quality Management

To the Developers and Testers

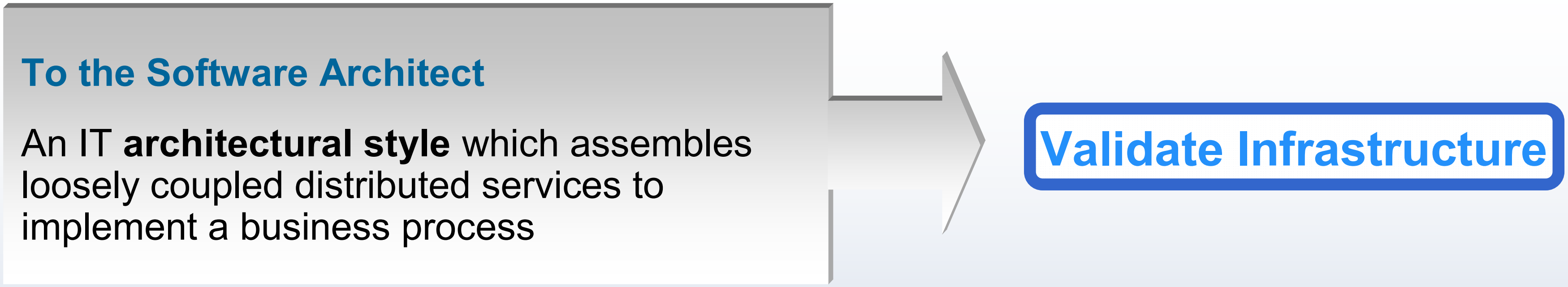
Web Services.

Validate Web Services

Challenges

- **No user accessible interface**
- **Multiple test case data cases per test**
- **Service interaction testing**

SOA: Implications for Quality Management



To the Software Architect

An IT **architectural style** which assembles loosely coupled distributed services to implement a business process

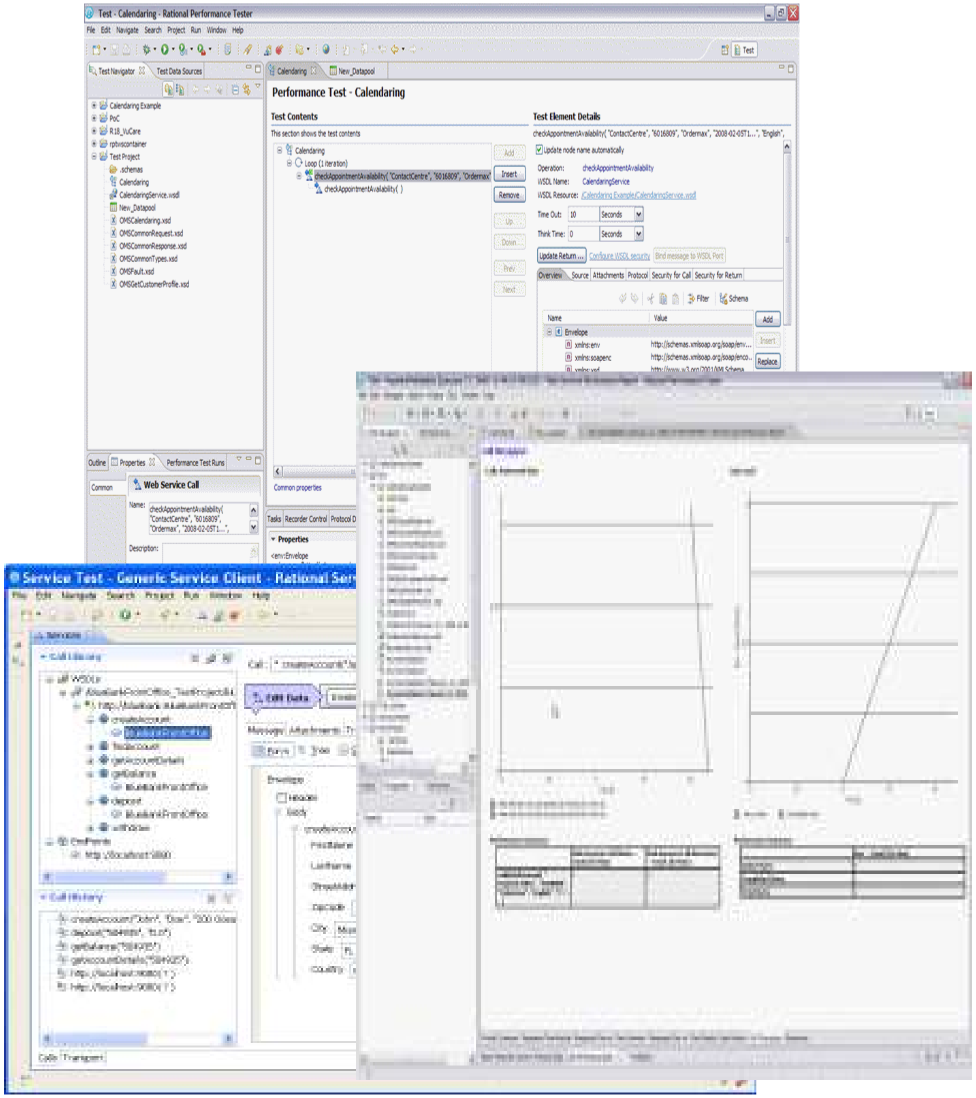
Validate Infrastructure

Challenges

- **Ensuring service operability post deployment**
- **Service upgrade & interoperability management**
- **Service Performance**

Rational Service Tester for SOA Quality

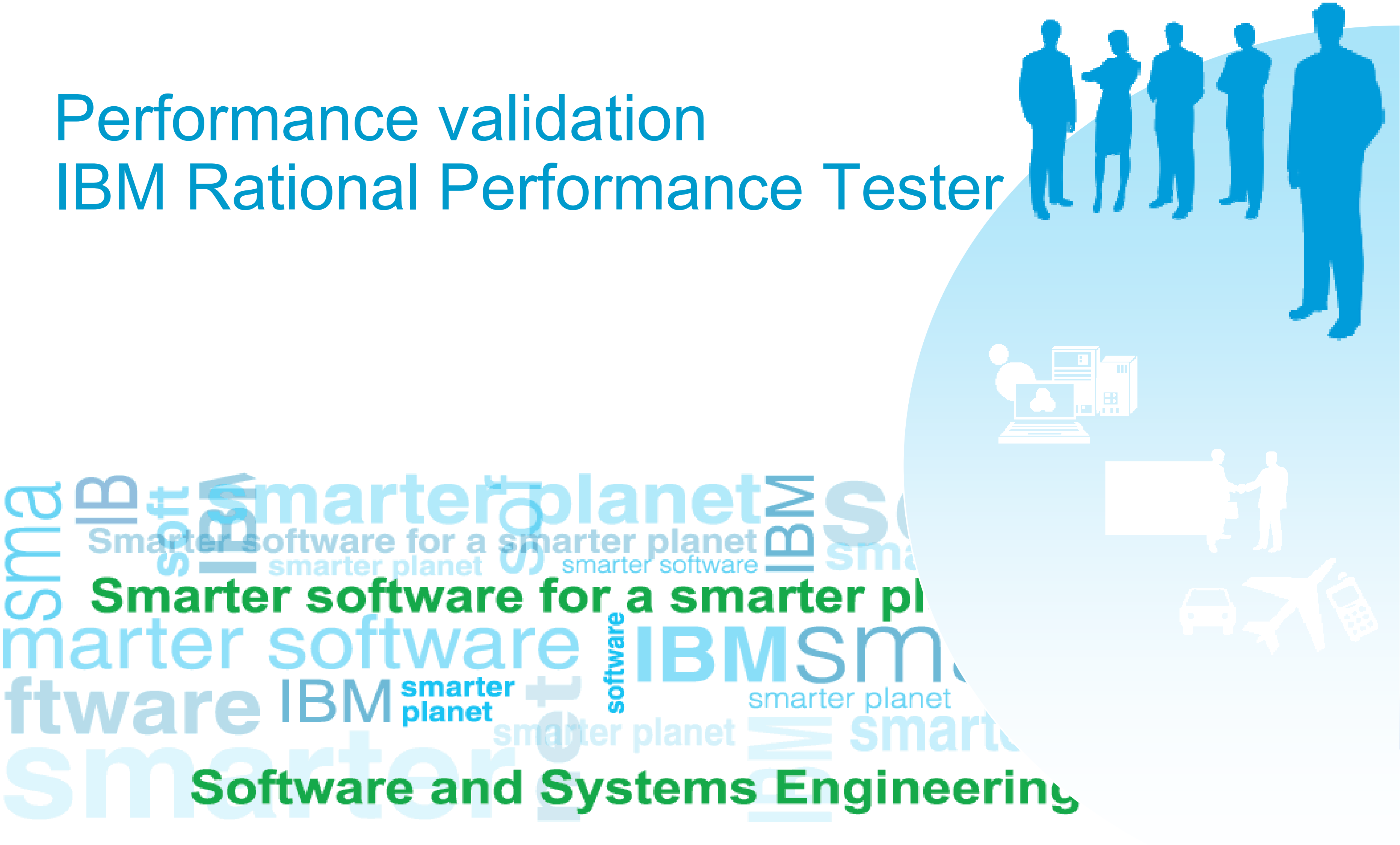
- Used to test web services
- Key features
  - Generic Web Services Client
  - XML editing, viewing
  - WSDL/Schema validation
  - Messaging and logging
  - Load and stress functions
  - Data driven testing
  - Java scripting
  - Automated Response validation
  - Performance Testing and Analysis


Test - Calendaring - Rational Performance Tester
Test Navigator
Test Data Sources
Performance Test - Calendaring
Test Contents
Test Element Details
Web Service Call
Service Test - Generic Service Client - Rational Service Tester
Call Library
Call History
Edit Data

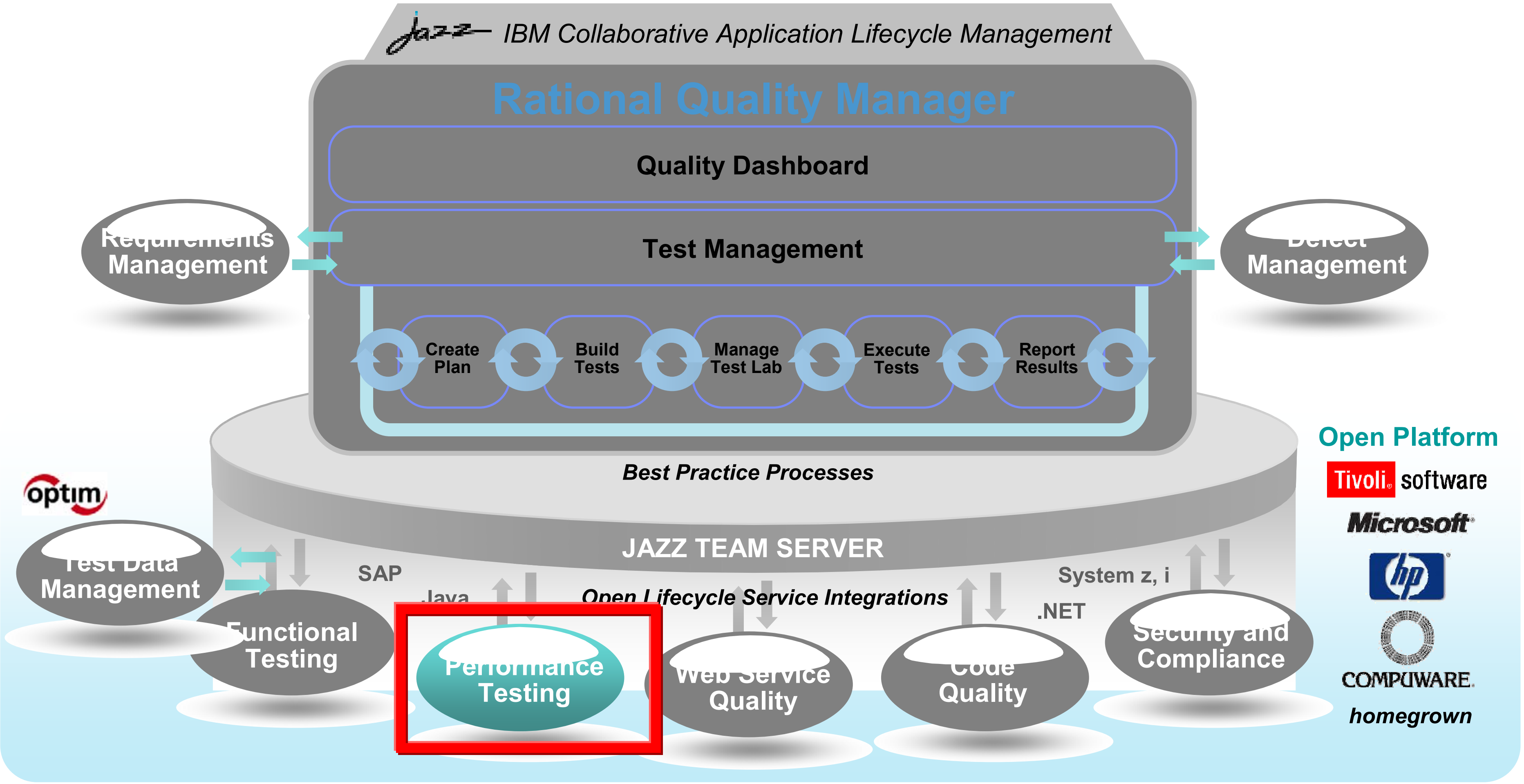
Rational Service Tester for SOA Quality

	Features	Benefits
Packaging	• **Performance and functional testing personas** • **Monitoring/Response Time Breakdown Package**	• **Improved consumability of functional testing capabilities** • **Improved visibility and support, and leverage value of performance problem determination features**
Environment Support	• **Support for additional WS-* standards** • **Text / JSON message formats** • **Support for IPv6**	• **Extend the range of supported SOA environments** • **Meet government requirements for IPv6 support**
Enterprise readiness	• **Improved support for multi-day runs with the ability to capture and process large volume of performance measurements**	• **Ability to address larger and more complex performance test opportunities**
Usability	• **Universal Service Test Client** • **Improved functional testing capabilities (creation, execution, reporting)**	• **Simple and unique user experience to create tests for all supported protocols** • **Improved consumability of functional testing capabilities**
Product Integrations	• **Support for Rational Quality Manager** • **Support for Rational Test Lab Manager**	• **Support quality throughout the life cycle through integration with Quality Management and Lab Management solutions**

Performance validation
IBM Rational Performance Tester


Smarter software for a smarter planet
Software and Systems Engineering

Centralized test management offering allowing full lifecycle support across all types of testing and platforms



IBM Collaborative Application Lifecycle Management

Rational Quality Manager

Quality Dashboard

Test Management

Create Plan

Build Tests

Manage Test Lab

Execute Tests

Report Results

Requirements Management

Defect Management

Best Practice Processes

JAZZ TEAM SERVER

Open Lifecycle Service Integrations

SAP

Java

System z, i

.NET

optim

Test Data Management

Functional Testing

Performance Testing

Web Service Quality

Code Quality

Security and Compliance

Open Platform

Tivoli software

Microsoft

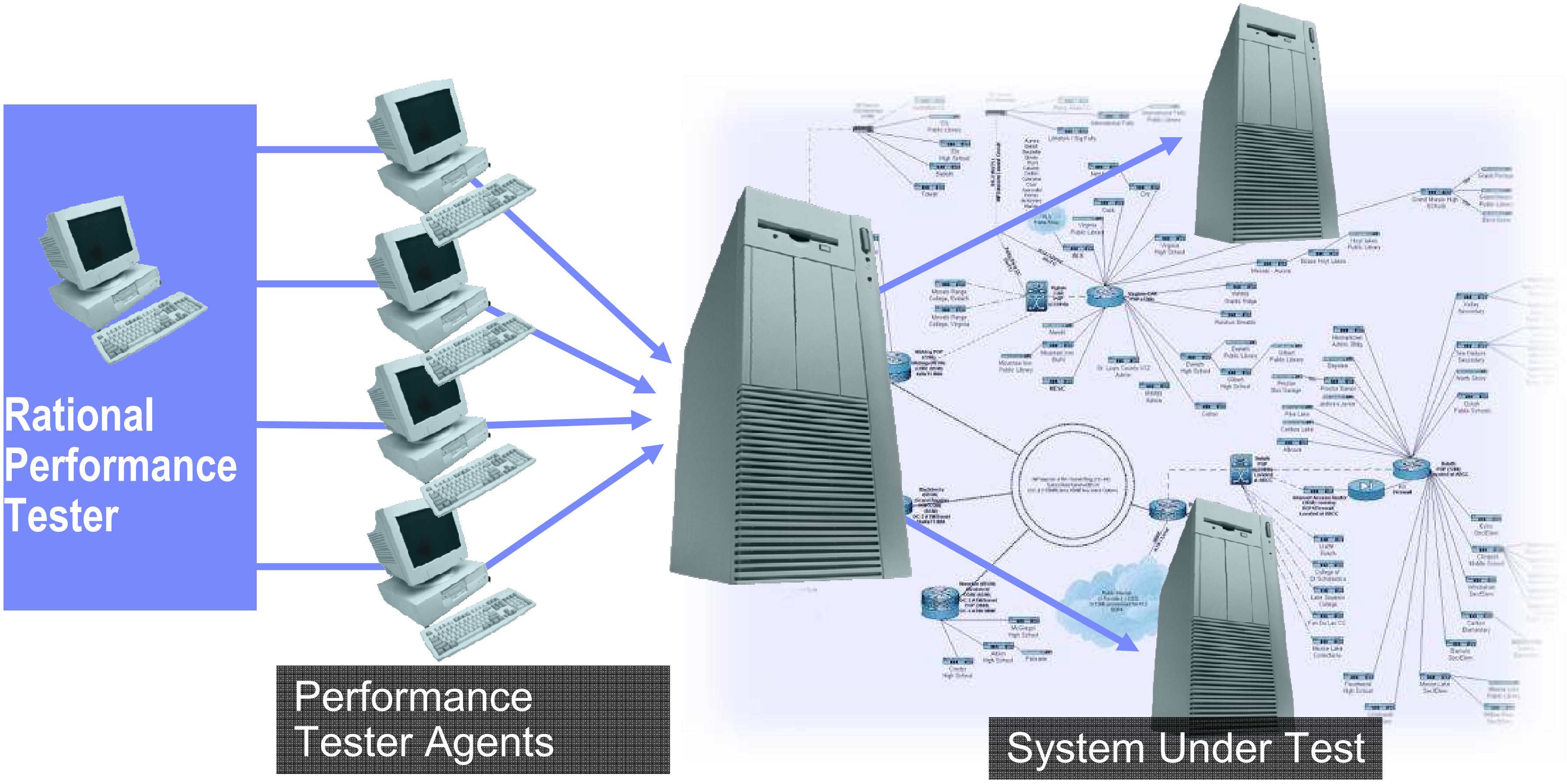
hp

COMPUWARE

homegrown

What Is Performance Testing?

- The process of exercising an application by emulating actual users with a load generation tool for the purpose of finding system bottlenecks



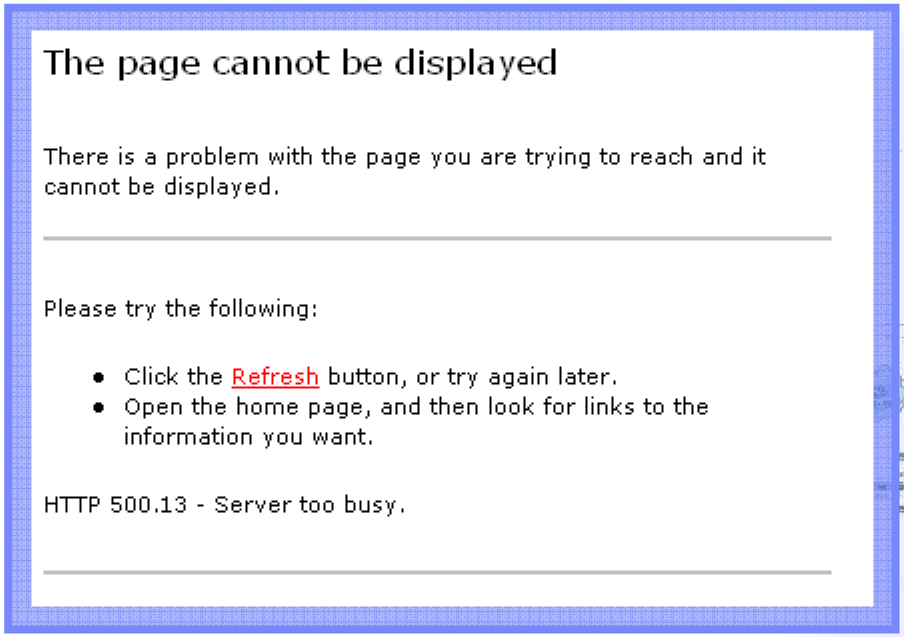
Rational Performance Tester

Performance Tester Agents

System Under Test

Why do Performance Testing?

- Because a break at any point in your system means your customers are not getting the service you think they are



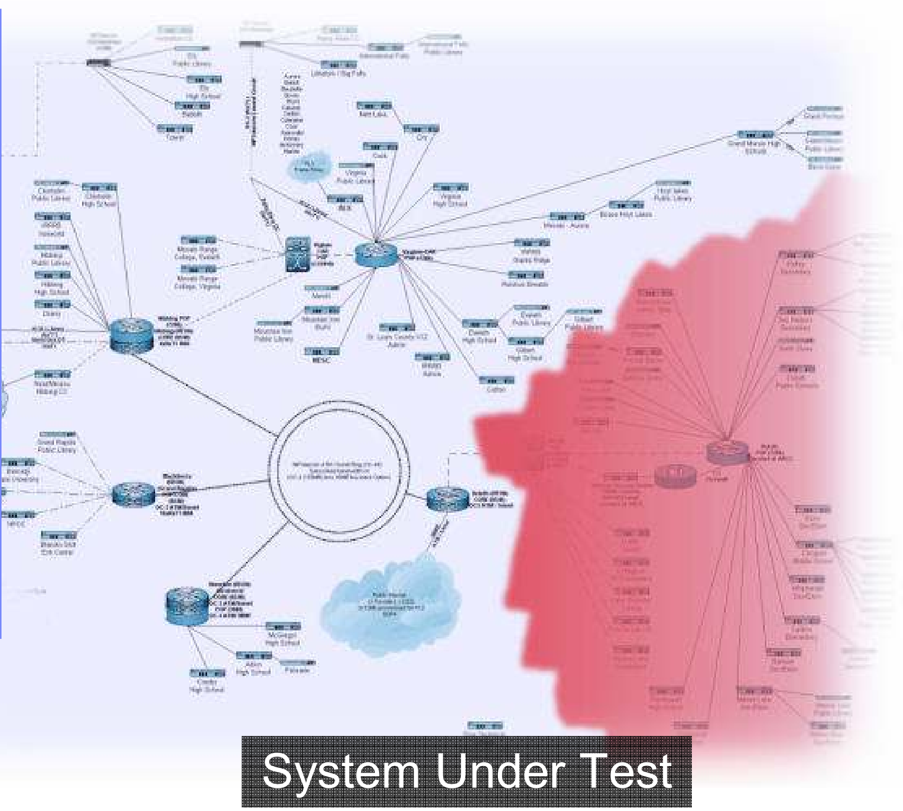
The page cannot be displayed

There is a problem with the page you are trying to reach and it cannot be displayed.

Please try the following:

- Click the Refresh button, or try again later.
- Open the home page, and then look for links to the information you want.

HTTP 500.13 - Server too busy.



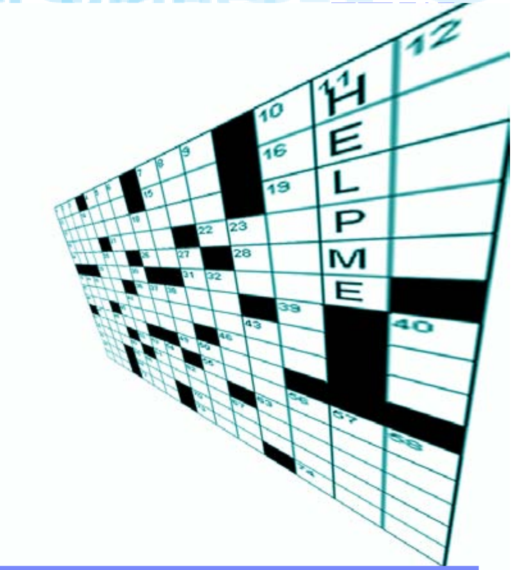
System Under Test

Performance Testing with IBM Rational Performance Tester

Test automation for the novice and the professional

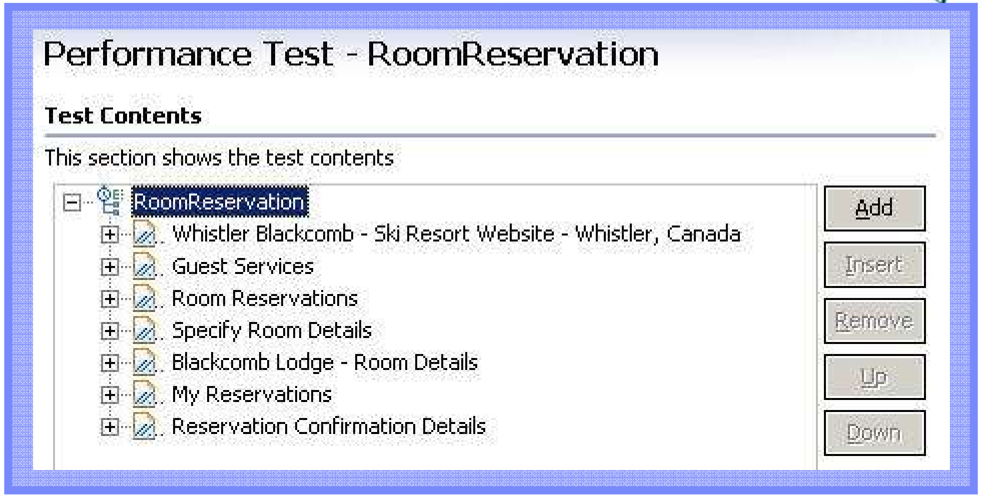
- **IBM Rational Performance Tester**
  - Performance problem identification and diagnosis for Web, SAP, 3270, Siebel, Oracle and Citrix based applications
- **Performance test automation**
  - **Built for Day 1 Productivity**
    - Mask complexity to get the job done
  - **Advanced Data Access & Manipulation**
    - Automated data variation and synchronization
  - **Root Cause Analysis**
    - Identifies location and **root cause** of performance problem in hardware and software

Challenge 1: No in-house experience



- **Challenge**
  - Tool complexity and lack of experience intimidates many first time users

- **Resolution**
  - Represent tests as a tree view of sequential flow through application
  - Simplify test editing with wizards
    - Looping
    - Conditional events
    - Data validation
  - Integrate Java code to handle unique performance challenges



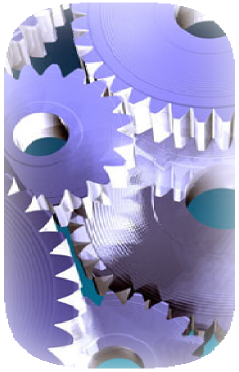
Performance Test - RoomReservation

Test Contents

This section shows the test contents

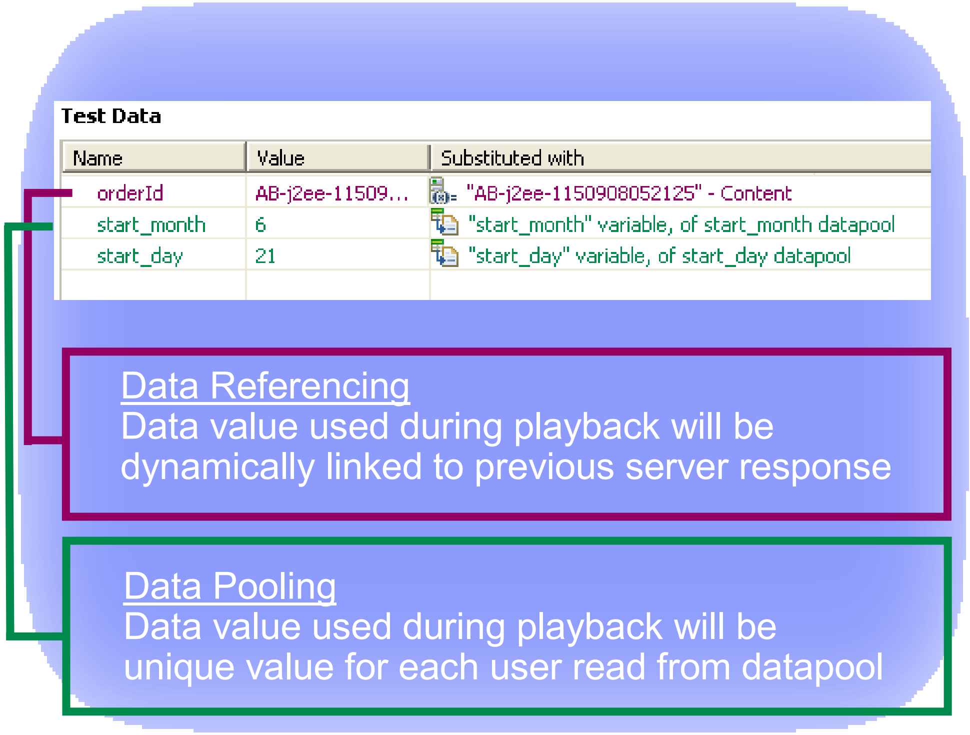
- RoomReservation
  - Whistler Blackcomb - Ski Resort Website - Whistler, Canada
  - Guest Services
  - Room Reservations
  - Specify Room Details
  - Blackcomb Lodge - Room Details
  - My Reservations
  - Reservation Confirmation Details

Challenge 2: Complexity of System Under Test



- **Challenge**
  - Complexity of system under test prohibits simple record and playback

- **Resolution**
  - Integrate data pooling technology to ensure each unique data for each user
  - Identify data relationships to dynamically reference server generated data during playback
  - Utilize unique TCP/IP addresses for each user to ensure realistic load



Test Data

Name	Value	Substituted with
orderId	AB-j2ee-11509...	"AB-j2ee-1150908052125" - Content
start_month	6	"start_month" variable, of start_month datapool
start_day	21	"start_day" variable, of start_day datapool

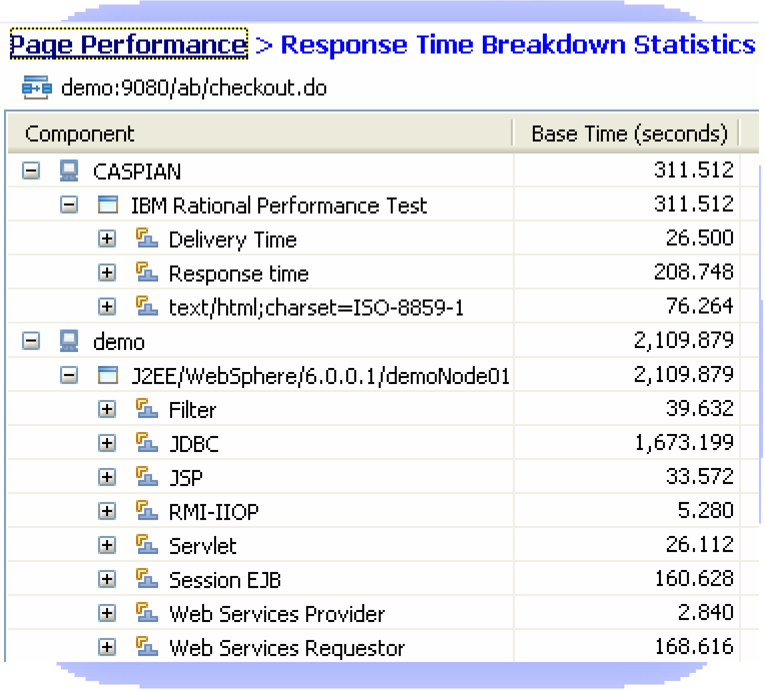
Data Referencing
Data value used during playback will be dynamically linked to previous server response

Data Pooling
Data value used during playback will be unique value for each user read from datapool

Challenge 2: Tools Lack Insight

- **Challenge**
  - Tool can find the problem, but not diagnose the root cause

- **Resolution**
  - Root Cause Analysis features provide additional insight to diagnose the cause of a bottleneck
  - Resource Monitoring data monitors hardware during test
  - Response Time Breakdown report breaks down response times into



Page Performance > Response Time Breakdown Statistics

demo:9080/ab/checkout.do

Component	Base Time (seconds)
⊟ CASPIAN	311.512
⊟ IBM Rational Performance Test	311.512
⊞ Delivery Time	26.500
⊞ Response time	208.748
⊞ text/html;charset=ISO-8859-1	76.264
⊟ demo	2,109.879
⊟ J2EE/WebSphere/6.0.0.1/demoNode01	2,109.879
⊞ Filter	39.632
⊞ JDBC	1,673.199
⊞ JSP	33.572
⊞ RMI-IIOP	5.280
⊞ Servlet	26.112
⊞ Session EJB	160.628
⊞ Web Services Provider	2.840
⊞ Web Services Requestor	168.616

Creating a Performance Test

Creating a performance test is a three step process

Build Scripts

- **Script Creation Considerations**
  - Visual test editor, varying input data & correlating server responses

Creating a Performance Test

Creating a performance test is a three step process

Build Scripts Schedule Workload

- **Script Creation Considerations**
  - Visual test editor, varying input data & correlating server responses
- **Scheduling Considerations**
  - Accurately representing a true user workload

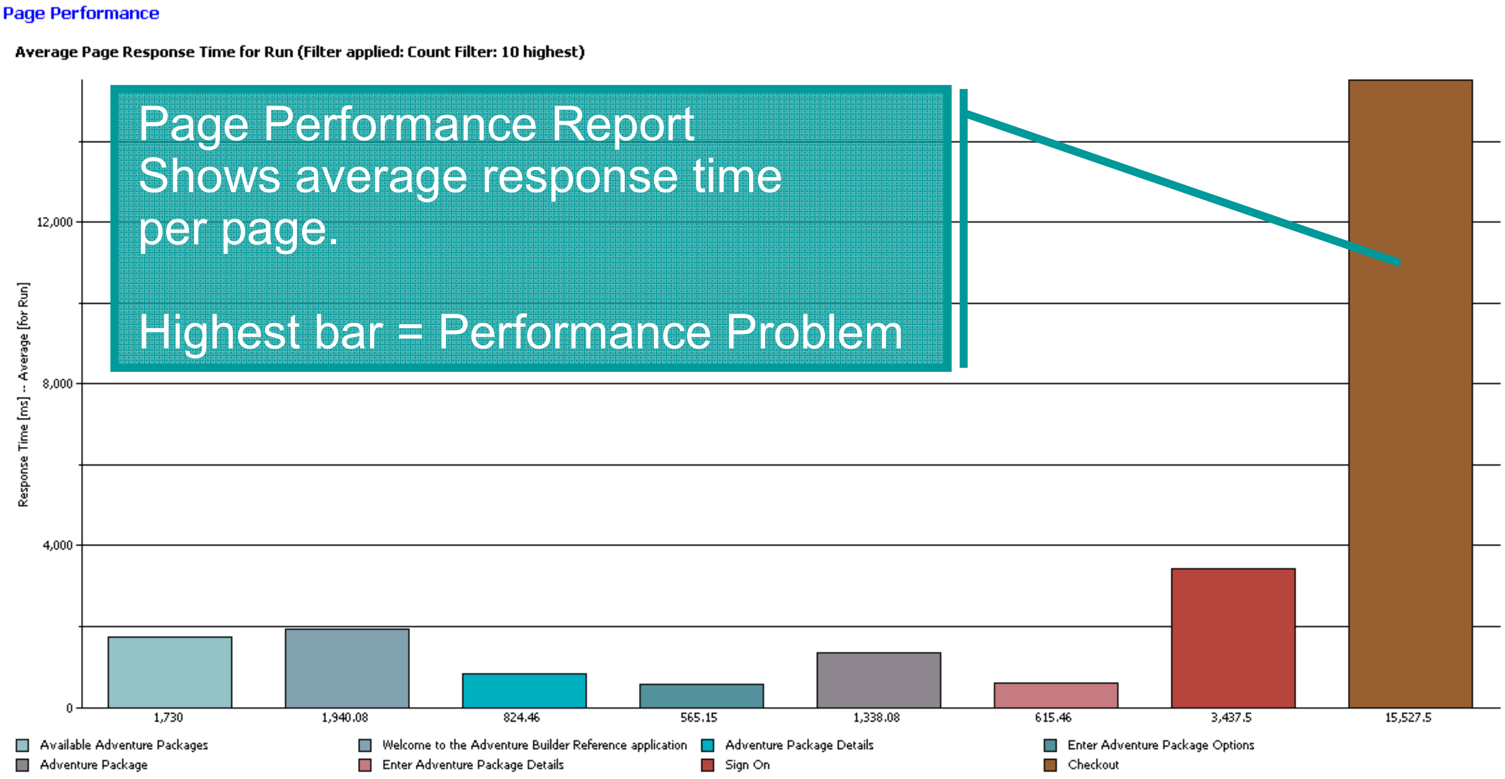
Creating a Performance Test

Creating a performance test is a three step process

Build Scripts **Schedule Workload** **Execute & Analyze**

- **Script Creation Considerations**
  - Visual test editor, varying input data & correlating server responses
- **Scheduling Considerations**
  - Accurately representing a true user workload
- **Execute and Analyze Considerations**
  - Validating responses & finding the bottleneck

Performance Problem Identification During Test



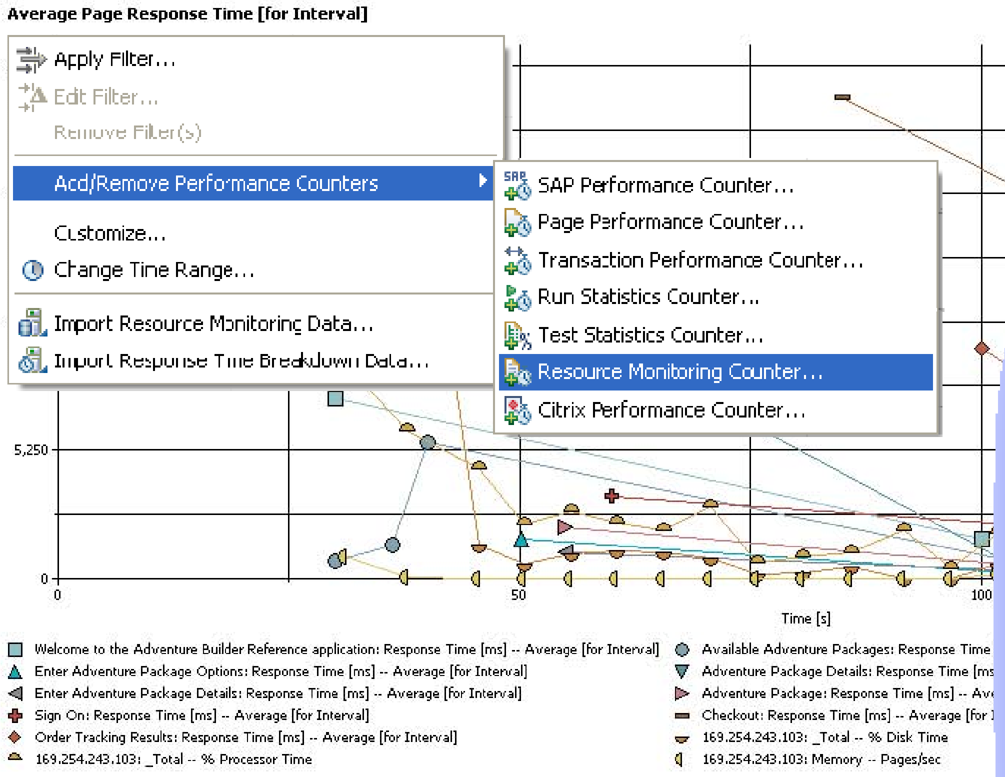
Page Performance Report
Shows average response time
per page.

Highest bar = Performance Problem

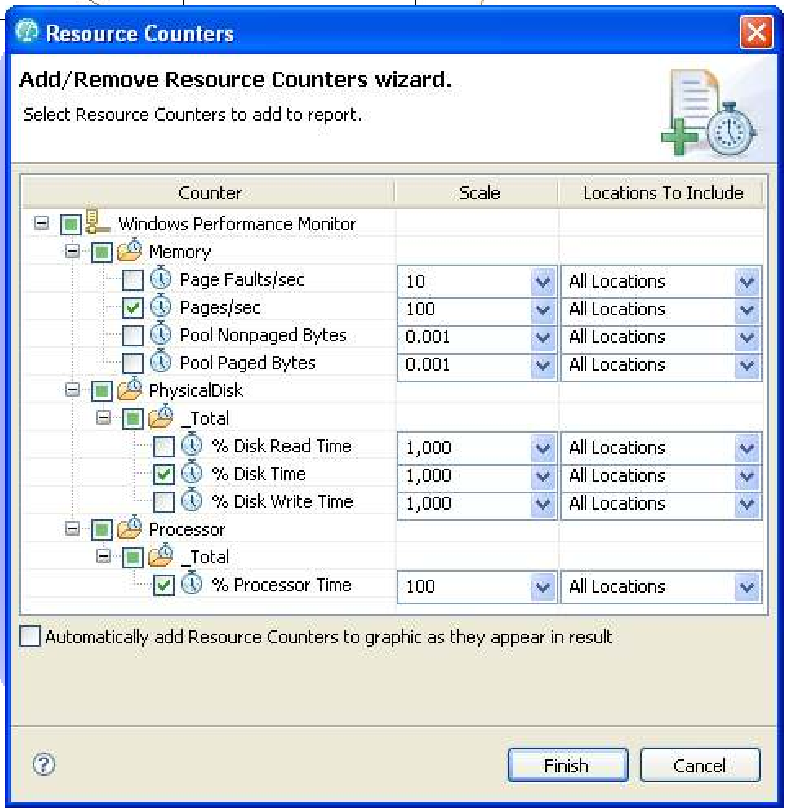
- Performance Testing finds bottlenecks
  - Next logical question is Why?
  - Root Cause Analysis provides to tools to answer this question

Performance & Resource Statistic Report Overlay

Identifying hardware related performance problems


Average Page Response Time [for Interval]
Apply Filter...
Edit Filter...
Remove Filter(s)
Add/Remove Performance Counters
Customize...
Change Time Range...
Import Resource Monitoring Data...
Import Response Time Breakdown Data...
SAP Performance Counter...
Page Performance Counter...
Transaction Performance Counter...
Run Statistics Counter...
Test Statistics Counter...
Resource Monitoring Counter...
Citrix Performance Counter...
5,250
0
0
50
100
Time [s]
Welcome to the Adventure Builder Reference application: Response Time [ms] -- Average [for Interval]
Enter Adventure Package Options: Response Time [ms] -- Average [for Interval]
Enter Adventure Package Details: Response Time [ms] -- Average [for Interval]
Sign On: Response Time [ms] -- Average [for Interval]
Order Tracking Results: Response Time [ms] -- Average [for Interval]
169.254.243.103: _Total -- % Processor Time
Available Adventure Packages: Response Time
Adventure Package Details: Response Time [ms
Adventure Package: Response Time [ms] -- Ave
Checkout: Response Time [ms] -- Average [for I
169.254.243.103: _Total -- % Disk Time
169.254.243.103: Memory -- Pages/sec

- Data from resource monitoring can be displayed on same graph as response time data
- Single view to visually correlate system resource and system response data for faster problem solving



Resource Counters

Add/Remove Resource Counters wizard.

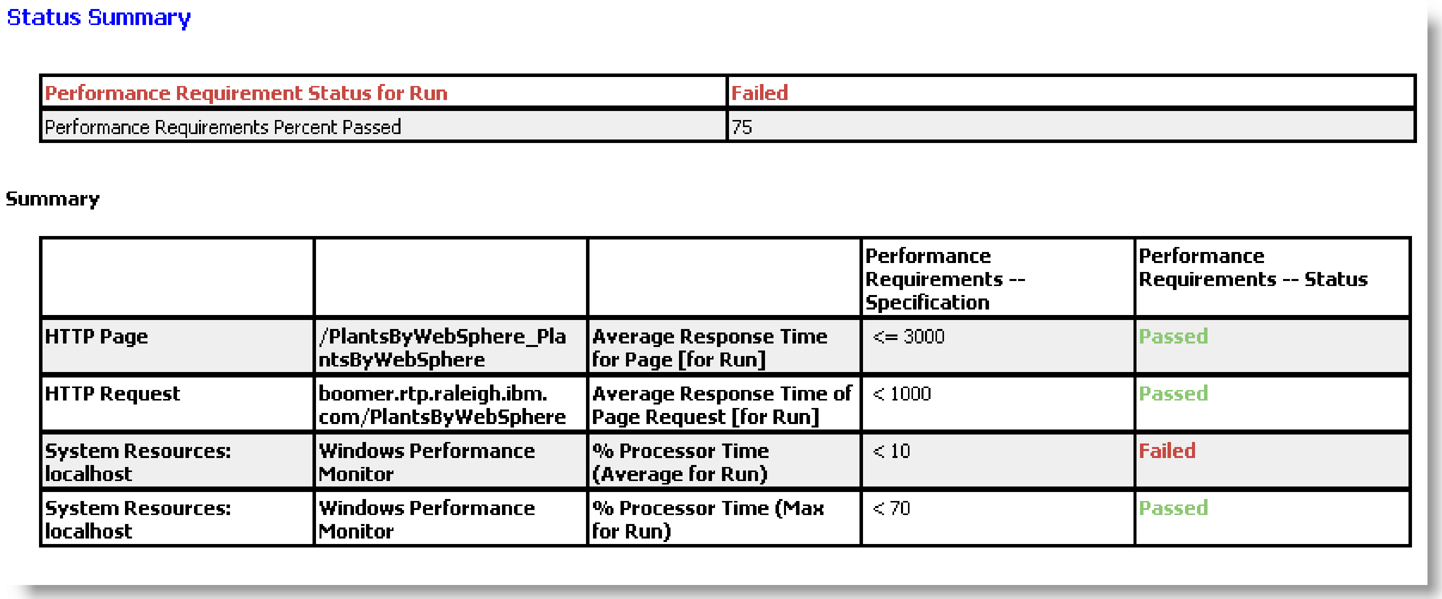
Select Resource Counters to add to report.

Counter	Scale	Locations To Include
⊟ ▣ Windows Performance Monitor		
⊟ ▣ Memory		
□ Page Faults/sec	10	All Locations
☑ Pages/sec	100	All Locations
□ Pool Nonpaged Bytes	0.001	All Locations
□ Pool Paged Bytes	0.001	All Locations
⊟ ▣ PhysicalDisk		
⊟ ▣ _Total		
□ % Disk Read Time	1,000	All Locations
☑ % Disk Time	1,000	All Locations
□ % Disk Write Time	1,000	All Locations
⊟ ▣ Processor		
⊟ ▣ _Total		
☑ % Processor Time	100	All Locations

□ Automatically add Resource Counters to graphic as they appear in result

Business SLA Reporting

Linking performance results to business objectives



Status Summary

Performance Requirement Status for Run	Failed
Performance Requirements Percent Passed	75

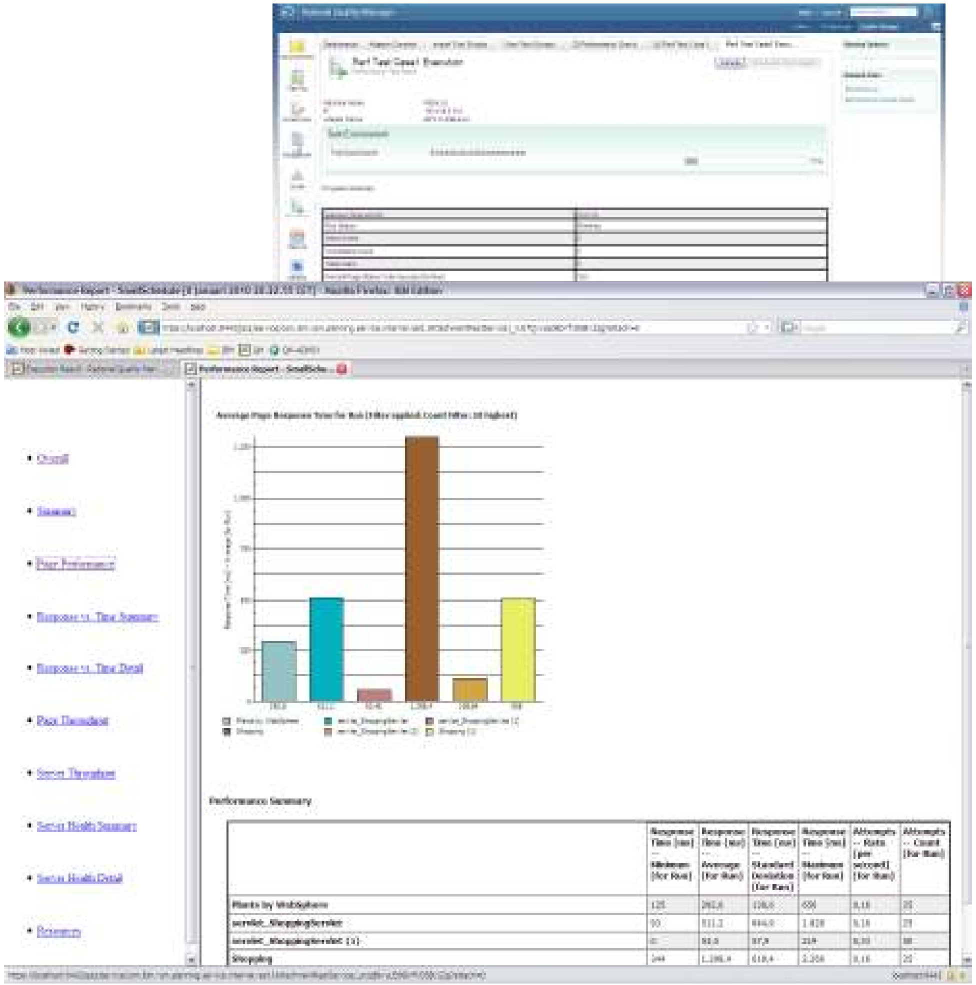
Summary

			Performance Requirements -- Specification	Performance Requirements -- Status
HTTP Page	/PlantsByWebSphere_PlantsByWebSphere	Average Response Time for Page [for Run]	<= 3000	Passed
HTTP Request	boomer.rtp.raleigh.ibm.com/PlantsByWebSphere	Average Response Time of Page Request [for Run]	< 1000	Passed
System Resources: localhost	Windows Performance Monitor	% Processor Time (Average for Run)	< 10	Failed
System Resources: localhost	Windows Performance Monitor	% Processor Time (Max for Run)	< 70	Passed

- Define detailed performance requirements in Rational Performance Tester
- Communicate results against performance criteria
- Results automatically rolled up and reported against user-defined SLA
- Results and reports are passed to RQM for wide visibility

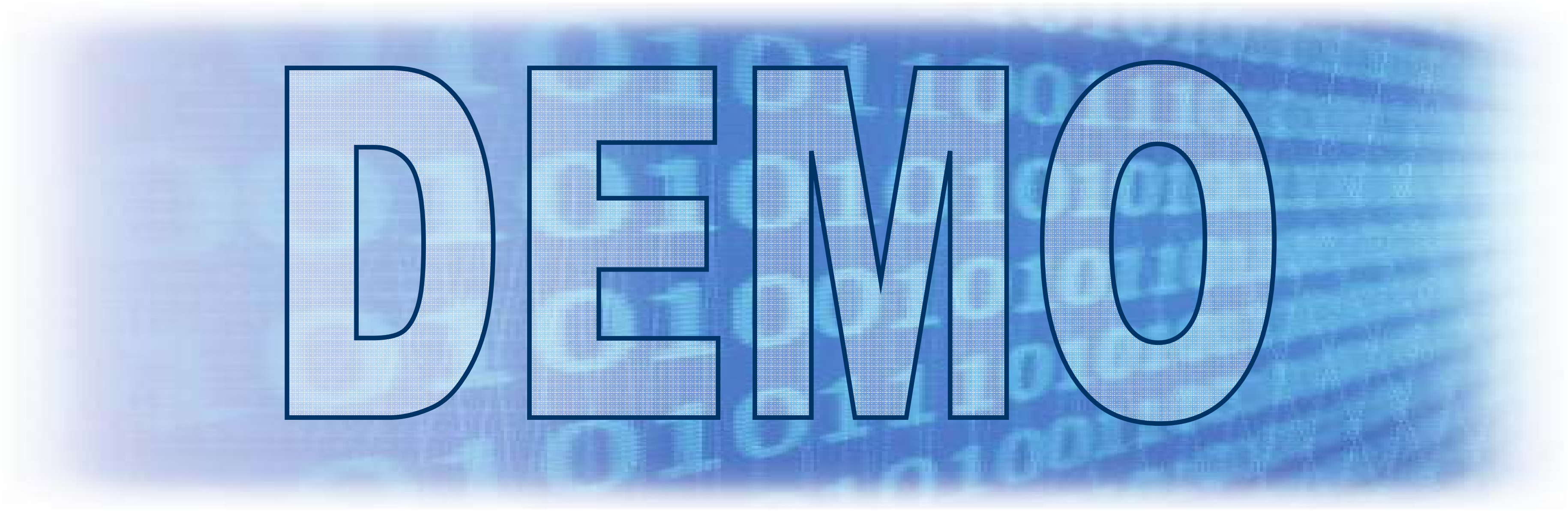
Run Performance Test from Rational Quality Manager

- Utilize any RQM browser to start Performance Test Case
- Utilizing power of RQM
  - Schedule daily at 01:00
- Follow progress while executing
- Results are communicated back to RQM


Perf Test Case1 Execution
Performance Report - SmallSchedule
Overall
Summary
Page Performance
Response vs. Time Summary
Response vs. Time Detail
Page Throughput
Server Throughput
Server Health Summary
Server Health Detail
Resources
Average Page Response Time for Run [Filter applied: Count filter: 10 highest]

Performance Summary

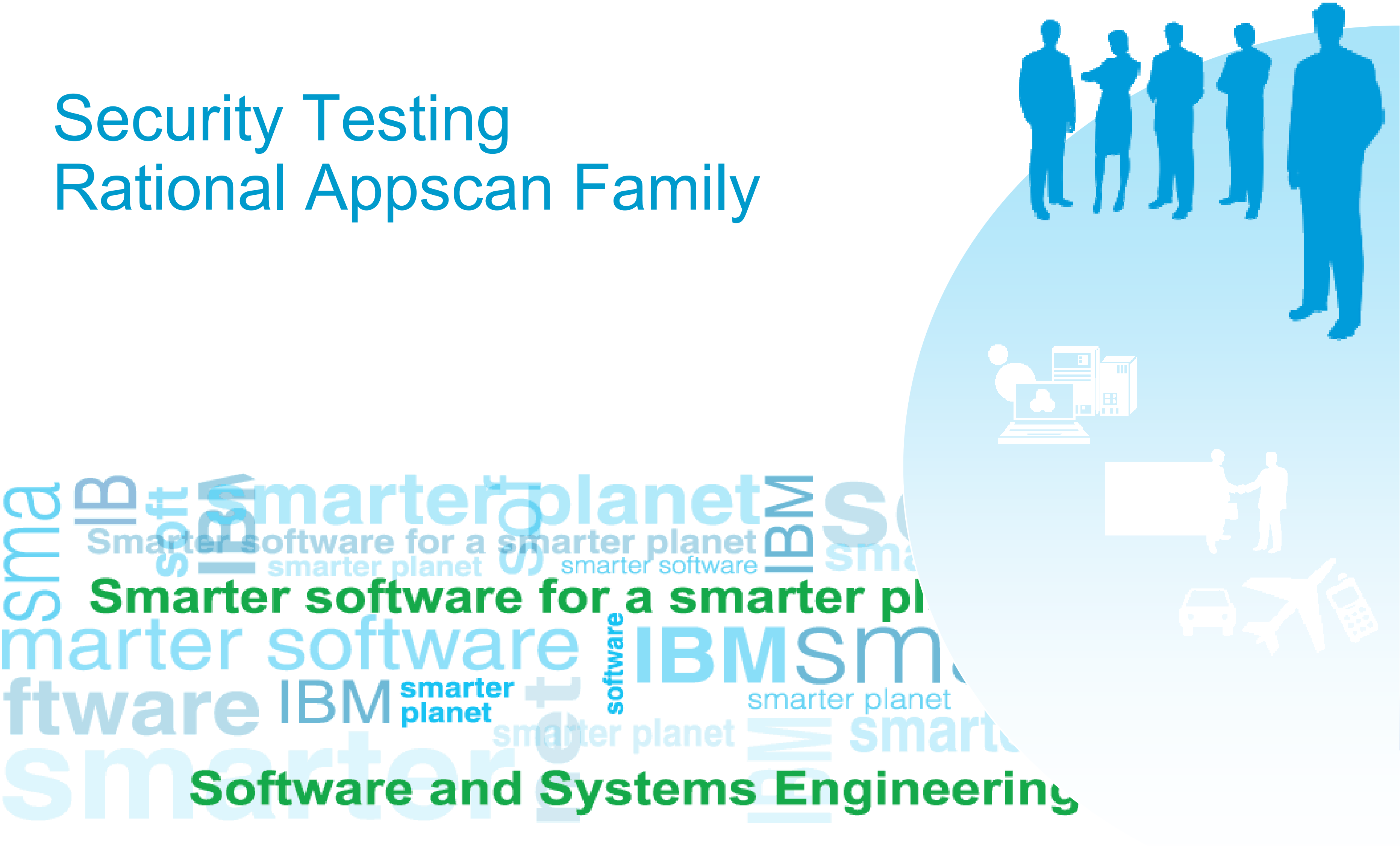
	Response Time [ms] -- Minimum [for Run]	Response Time [ms] -- Average [for Run]	Response Time [ms] -- Standard Deviation [for Run]	Response Time [ms] -- Maximum [for Run]	Attempts -- Rate [per second] [for Run]	Attempts -- Count [for Run]
Plants by WebSphere	125	282,6	138,6	656	0,16	25
servlet_ShoppingServlet	50	511,2	444,0	1.828	0,16	25
servlet_ShoppingServlet (1)	0	51,0	57,9	219	0,33	50
Shopping	344	1.206,4	618,4	2.356	0,16	25



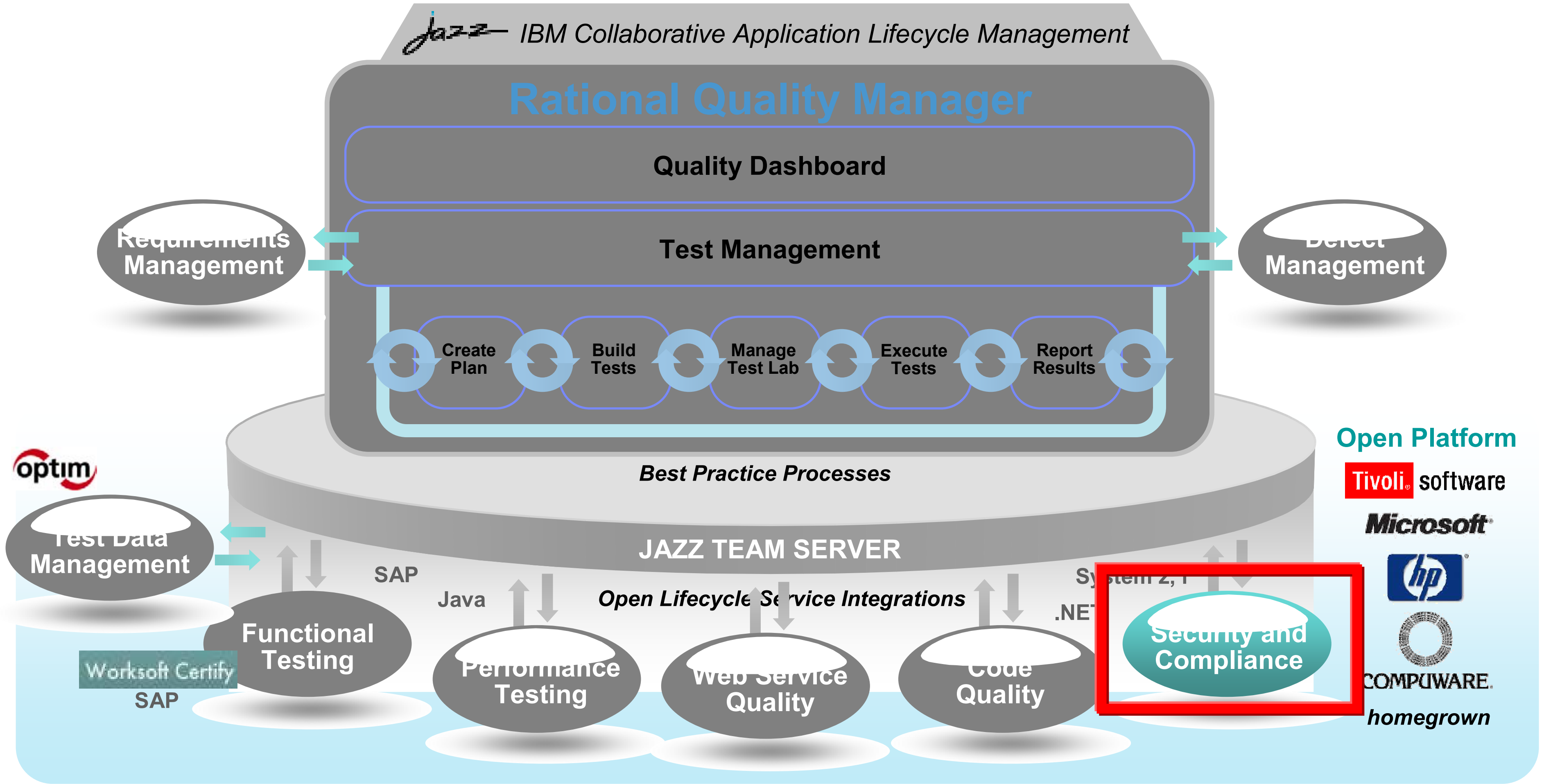
What You'll See:

- Rational Performance Tester

Security Testing
Rational Appscan Family


Smarter software for a smarter planet
smarter planet
smarter software
Smarter software for a smarter pl
marter software
IBM
smarter planet
Software and Systems Engineering

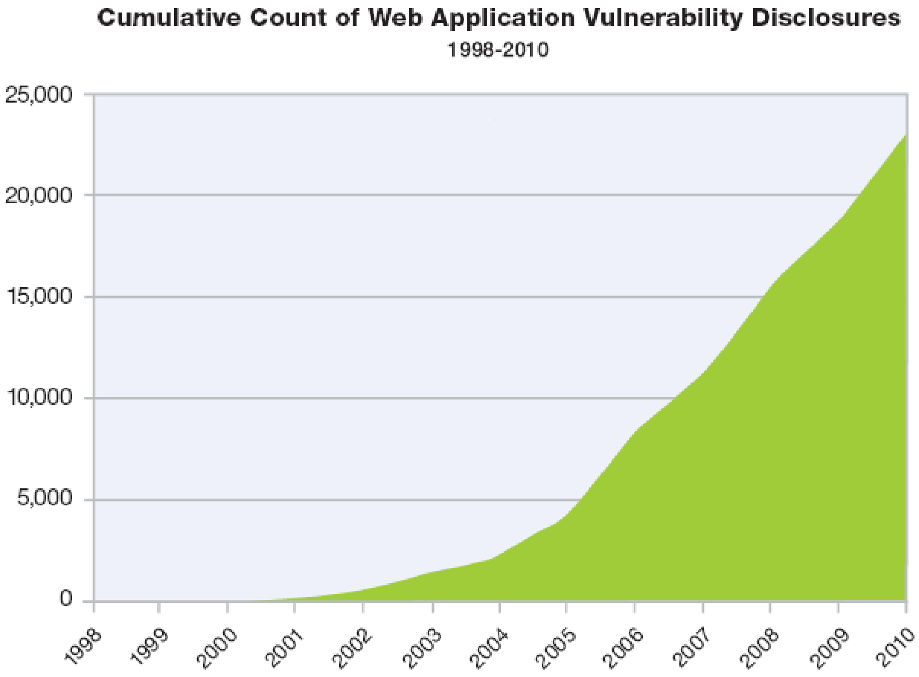
Centralized test management offering allowing full lifecycle support across all types of testing and platforms

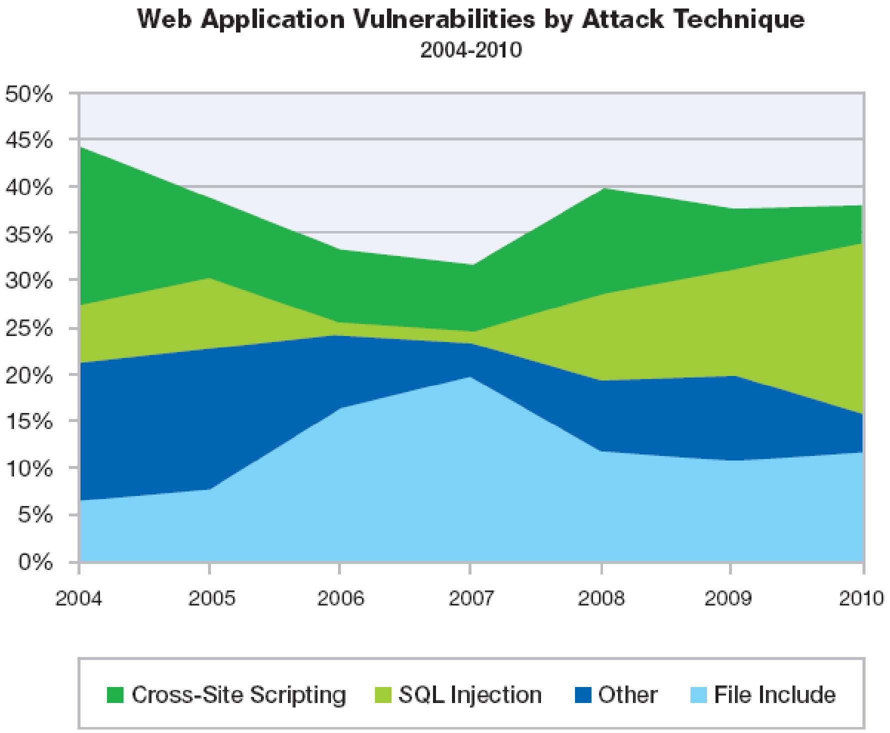


Application security challenges: vulnerabilities

Web application vulnerabilities dominate enterprise threat landscape

- **49%** of all vulnerabilities are in web applications*
- Cross-Site Scripting & SQL injection vulnerabilities continue to dominate





* IBM X-Force 2010 Trend & Risk Report

Application security challenges: security-development disconnect fails to prevent vulnerabilities in production applications

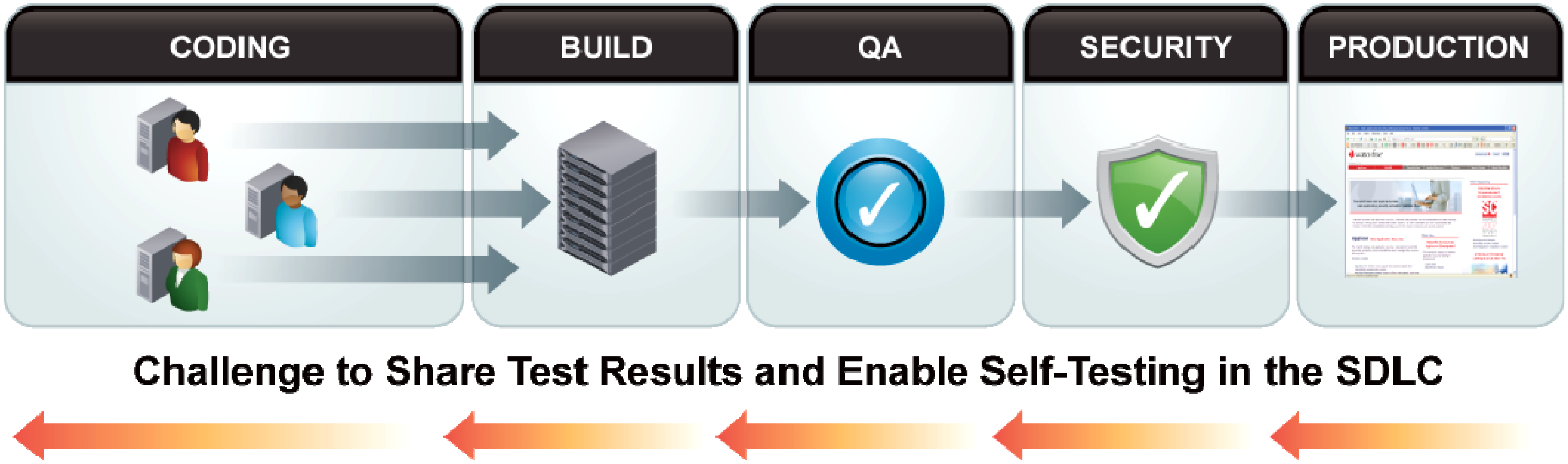
▪ Developers Lack Security Insights

(or Incentives to Address Security)

- Mandate to deliver functionality on-time and on-budget – but not to develop secure applications
- Developers rarely educated in secure code practices
- Product innovation drives development of increasingly complicated applications

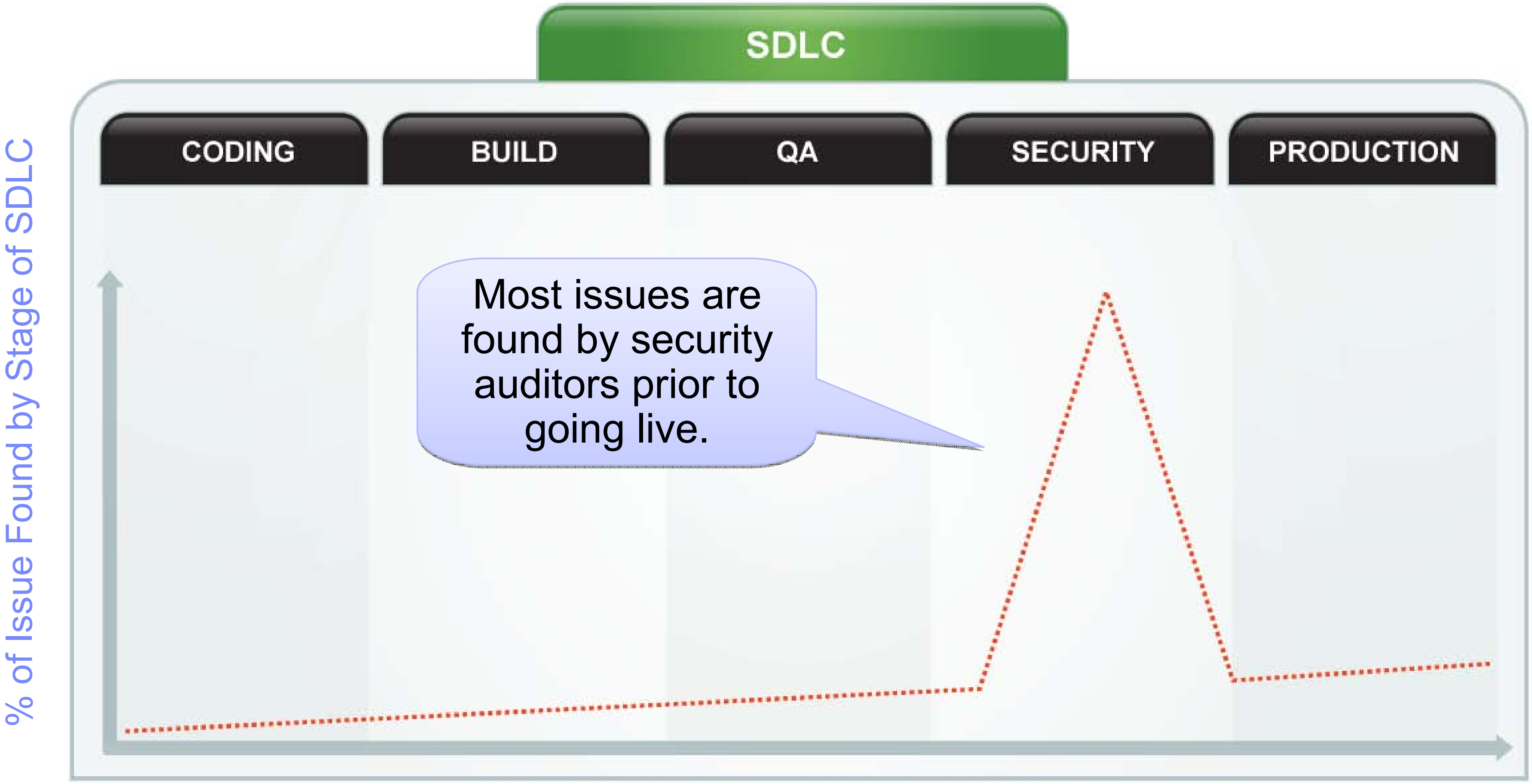
▪ Security Team = SDLC Bottleneck

- Security tests executed just before launch
  - Adds time and cost to fix vulnerabilities late in the process
- Growing number of web applications but small security staff
  - Most enterprises scan ~10% of all applications
- Continuous monitoring of production apps limited or non-existent
  - Unidentified vulnerabilities & risk



Challenge to Share Test Results and Enable Self-Testing in the SDLC

Security testing within the application life cycle



Most issues are found by security auditors prior to going live.

Prevention...

A little bit every day

- Low cost
- Low pain
- Low disruption

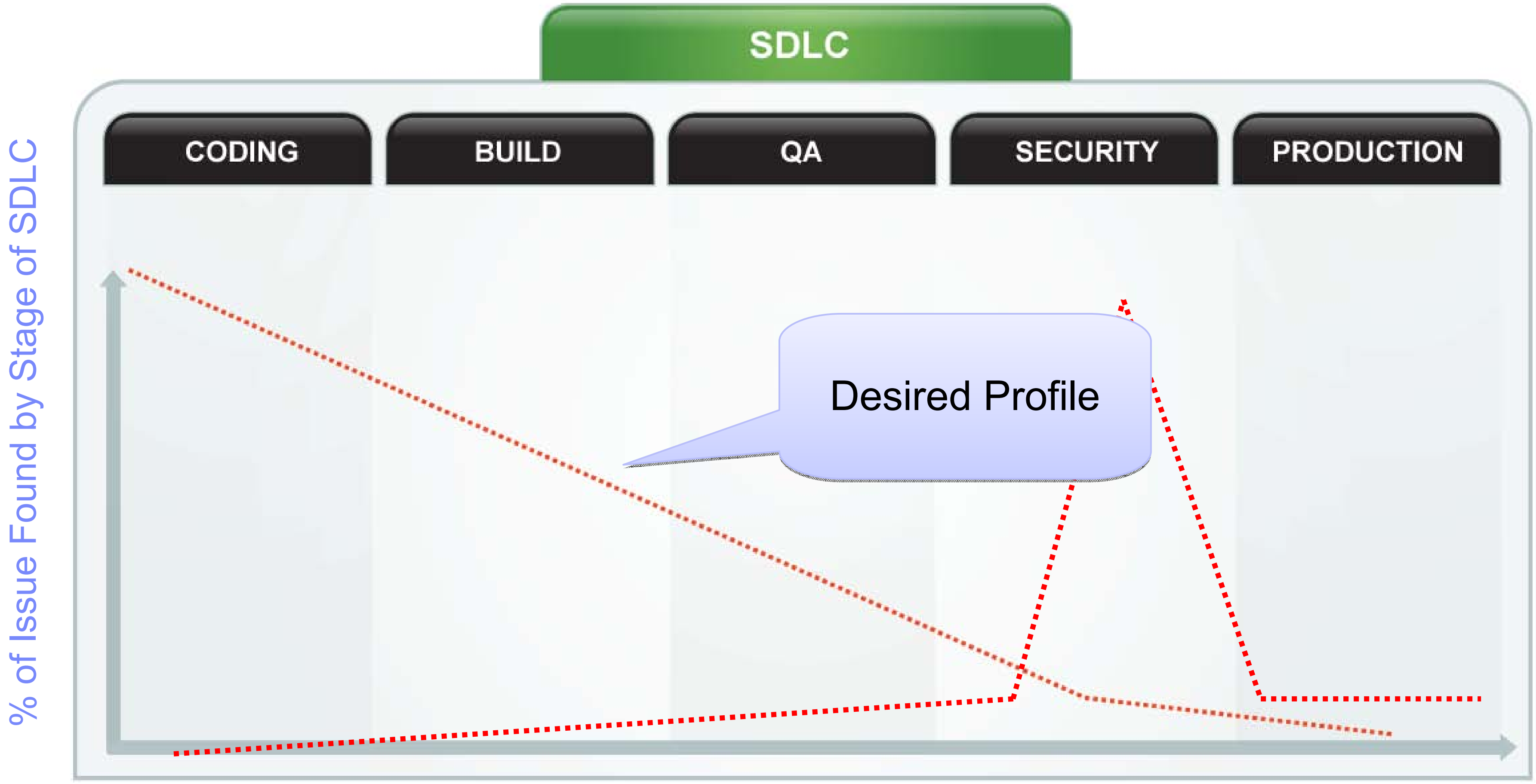
This?

Or This?

Ignore the issue until...

- High cost
- High pain
- High disruption

Security testing within the application life cycle



Desired Profile

Make applications secure, by design

Cycle of secure application development

- **Design**
- Consider security requirements of the application & apply threat models
- Issues such as required controls and best practices are documented on par with functional requirements
- Secure code libraries maintained for reusable secure code

- **Development**
- Create work items that map to security requirements
- Use secure code libraries
- Software is checked during coding for:
  - Implementation error vulnerabilities
  - Compliance with security requirements

- **Build & Test**
- Map test plan to security requirements
- Testing begins for errors and compliance with security requirements across the entire application
- Applications are also tested for exploitability in deployment scenario

- **Deployment**
- Configure infrastructure for application policies
- Deploy applications into production

- **Operational**
- Continuously monitor applications for appropriate application usage, vulnerabilities and defend against attacks

Cost is a significant driver

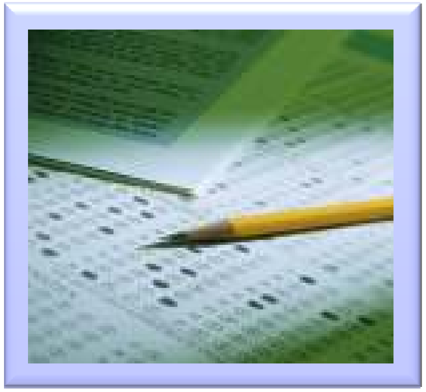
80% of development costs are spent identifying and correcting defects!*

During the
CODING phase

$80/defect

During the
BUILD phase

$240/defect



During the
QA/TESTING
phase

$960/defect

Once released
as a product

$7,600/defect

+

Law suits, loss of customer trust, damage to brand

*National Institute of Standards & Technology
Source: GBS Industry standard study
Defect cost derived in assuming it takes 8 hrs to find, fix and repair a defect when found in code and unit test.
Defect FFR cost for other phases calculated by using the multiplier on a blended rate of $80/hr.

Solution requirements: advanced security testing + collaboration & governance through application lifecycle

Advanced Security Assessments

Dynamic Analysis

- Analysis of a running/ deployed application
- Key requirements
  - Threat coverage: WASC, OWASP Top 10, etc
  - Web Services/ SOA
  - Web 2.0 & Rich Internet Applications

Static Analysis

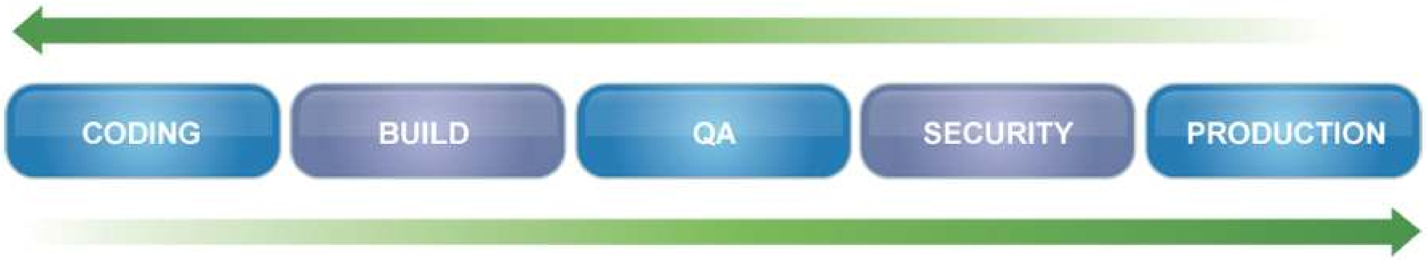
- Scanning source code for security issues
- Key requirements
  - Application/language support
  - Ease of use for non-security users (developers and build managers)

Runtime & Hybrid Analysis

- Glass box testing with runtime analysis
- Automated correlation of static & dynamic results
- Key requirements
  - Precise & Actionable results
  - Broad threat coverage

Collaboration & Governance in Application Lifecycle

Security testing, shared results, assign ownership



Track corrections and integrate with development systems

Solution Requirements: Static, Dynamic and Runtime Analysis

	Static Analysis (White Box testing)	**Dynamic Analysis** (Black Box testing)	**Runtime Analysis** (Glass Box testing)
Scan input	Scans source code and bytecode for security and quality issues. Requires access to source or bytecode	Scans running web applications. Requires starting point URL, and login credentials where relevant	Similar to black box to scan running web applications with an agent installed on the application
Assessment techniques	Uses "taint analysis" and pattern matching techniques to locate issues	Tampering of HTTP messages to locate application and infrastructure layer issues	Agent monitors application performance during a black box scan for expanding threat coverage and greater detail
Role in application development lifecycle	**Development:** Scan code and work remediation from IDE **Build:** Scan nightly or weekly build to highlight defects for developers to correct **Security:** Define & customize security best practices for developers; Execute pre-production scans and audits	**Build:** Scan as part of build acceptance tests before releasing build to testing team **Test:** Execute security test scripts as part of quality plan **Security:** Define test scripts for quality plan; Execute pre-production scans and audits	**Build:** Provides added layer of vulnerability detail that assists developers with security de-bugging **Security:** Expands threat coverage for hard-to-identify vulnerabilities (including all OWASP Top 10)
Results & Output	Results are presented by line of code, source to sink functions flow	Results are presented as HTTP messages (exploit requests)	Results are presented as a combination of HTTP messages (exploit requests) and the line of code

Application Security: Where do I start?

- **First time conducting in-house application security assessments**
- **Most clients start with dynamic testing**
  - Dynamic analysis (black box testing) allows security groups to assess application risk in both development and production apps
  - Easy to roll out & automate work previously done with outsourced penetration testing
  - Select a solution that combines ease of use, advanced security analysis and results that can be shared outside of security
- **Application security testing confined to security team**
- **For deployments led and executed only by security teams, start with dynamic and later consider static (white box)**
  - Dynamic analysis is executed against compiled applications in lab environments, so security teams can control & execute the application security program
  - Select a solution that allows you to share results with development, cover all of your applications in both development and production, and later scale program with static analysis
- **Development & security teams integrate security testing in the SDLC**
- **Most clients evolve to this level of application security program with various use cases of dynamic and static analysis that fit their development processes**
  - Developers execute static analysis from their IDE or at least access static results from IDE
  - Build: Static analysis of each build and dynamic analysis before releasing build
  - Test: Dynamic testing included in test plan and executed from testing tools
  - Security: Conduct advanced dynamic and static testing before launch (benefit from early testing that eliminates the common security defects like SQL Injection and Cross-Site Scripting
  - Select a solution that delivers governance and collaboration while empowering non-security users

IBM AppScan: Advanced research drives precise security testing that integrates with application development lifecycle

Legacy of Security Innovation

Advanced testing technologies

- Dynamic Analysis (black box); IBM holds the original patent for dynamic web app security scans (US6584569)
- Static Analysis (white box)
- Runtime Analysis (glass box); patent filed 2008
- JavaScript Security Analyzer (static scans of client-side JavaScript)

Broad application support

- Web applications
- Packaged applications (SAP)
- Legacy applications (COBOL)

Broad technology coverage

- Web 2.0 and Rich Internet Applications
- Web Services/ SOA/SOAP

Governance and Collaboration in Application Development Lifecycle

Code

- Scan code, manage work items and remediate vulnerabilities from the IDE

Build

- Integrate security testing as a natural extension of build extension testing
- Find & fix defects before releasing a build

Test

- Include security testing in quality plan
- Execute basic security test scripts from quality management platform

Security

- Build security test scripts for non-security experts
- Focus pre-production audits on most advanced threats
- Manage test policies and scan permissions
- Collaborate with development to triage findings and assign ownership

AppScan Standard: **Desktop solution combines advanced security testing, broad technology coverage and ease of use**

Web Application Assessments for Pen-Testers and Security Practitioners

Dynamic Analysis (black box)

- **Covers all relevant OWASP & WASC TCv2 threat classes**
  - SQL Injection
  - Cross-Site Scripting
  - HTTP Response Splitting
  - OS Commanding
  - LDAP Injection
  - XPath Injection
  - Buffer Overflows
  - ***1000s more***
- **Web 2.0 and Rich Internet Applications**
  - JavaScript & Ajax
  - Adobe Flash & Flex
- **Malware analysis**
  - Scan site with malware analysis from IBM X-Force Security Research
- **Web Services/ SOA**
  - SOAP/XML parser issues (External entities, XML blowup, etc.)
  - Application-layer issues
  - Infrastructure issues

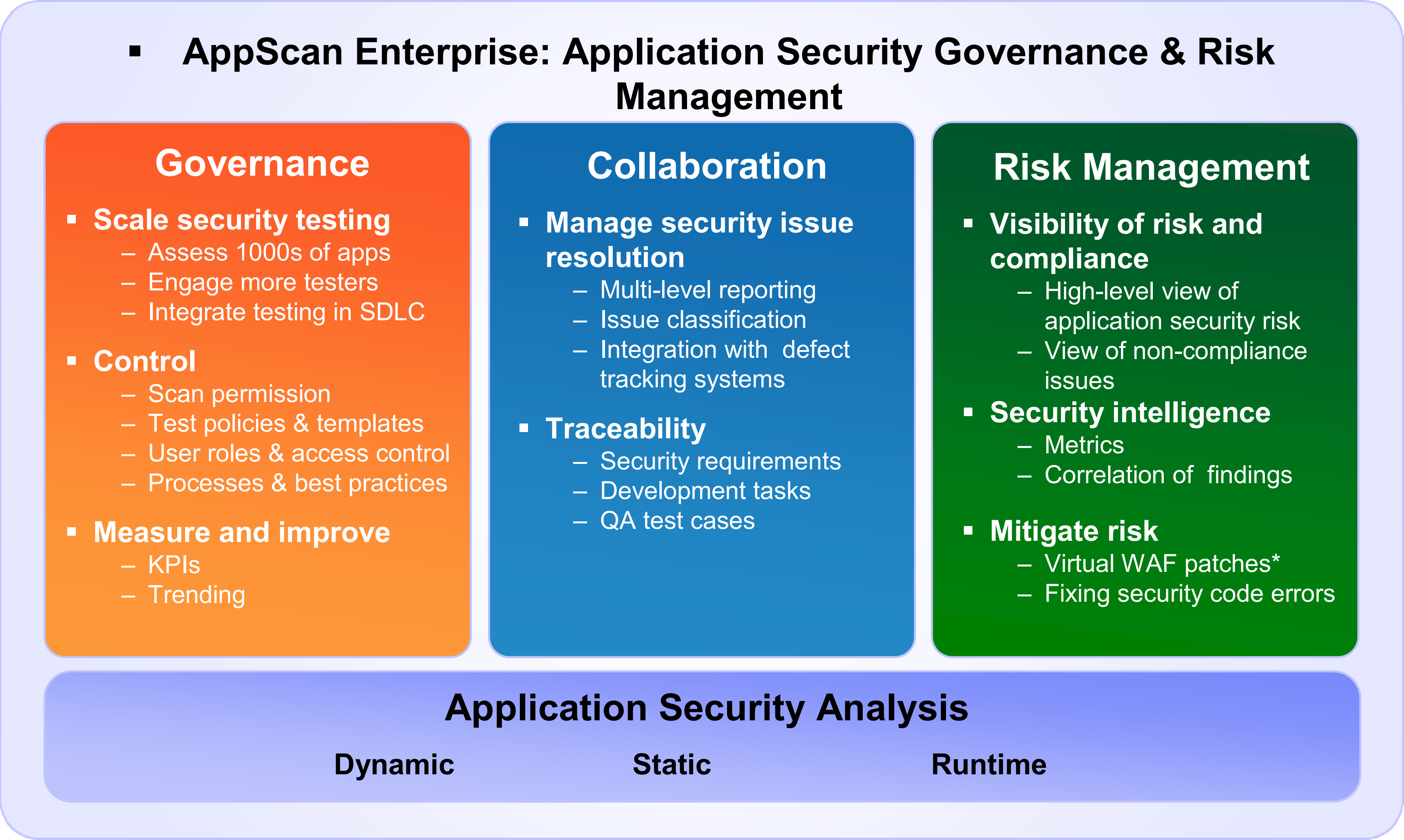
Hybrid Technology

- **Runtime Analysis (glass box testing)**
  - Expanded threat coverage with less configuration
  - Precise results (line of code) assist remediation
- **JavaScript Security Analyzer**
  - Static taint analysis of client-side JavaScript

Ease of Use

- **Configure & test**
  - Scan Expert provides recommended settings based on your apps
- **Details & guidance to correct the vulnerability**
  - Explanation of threat and recommended fix
- **Integrate with Defect Tracking Systems**
  - Rational® ClearQuest
  - HP Quality Center
- **Compliance & Reporting**
  - 40+ compliance reports
  - Executive-level summaries
  - Guidance for development

AppScan Enterprise



- **AppScan Enterprise: Application Security Governance & Risk Management**

Governance

- **Scale security testing**
  - Assess 1000s of apps
  - Engage more testers
  - Integrate testing in SDLC
- **Control**
  - Scan permission
  - Test policies & templates
  - User roles & access control
  - Processes & best practices
- **Measure and improve**
  - KPIs
  - Trending

Collaboration

- **Manage security issue resolution**
  - Multi-level reporting
  - Issue classification
  - Integration with defect tracking systems
- **Traceability**
  - Security requirements
  - Development tasks
  - QA test cases

Risk Management

- **Visibility of risk and compliance**
  - High-level view of application security risk
  - View of non-compliance issues
- **Security intelligence**
  - Metrics
  - Correlation of findings
- **Mitigate risk**
  - Virtual WAF patches*
  - Fixing security code errors

Application Security Analysis

Dynamic **Static** **Runtime**

AppScan Enterprise: **Security testing and visibility throughout the SDLC for enterprise-wide application risk management**

CODING | BUILD | QA | SECURITY | PRODUCTION

Information Security	▪ Schedule and automate assessments ▪ Manage test policies and scan permissions ▪ Collaborate with development and QA by publish findings for remediation ▪ Build protection strategies based on known vulnerabilities
Development & Build Automation	▪ Analyze source code for security issues in applications, projects or files from IDE or automatically trigger scans in Build system ▪ Remediate vulnerabilities with details and recommended fixes available in IDE ▪ Execute source code scans Execute dynamic test of compiled applications to identify and remediate issues before passing build to QA
Quality Assurance	▪ Create security test plans & test scripts in Rational Quality Manager ▪ Manage open issues via defect tracking systems
Management	▪ Enterprise-view of application security risk ▪ Trending and reporting with key performance indicators
Compliance Officers	▪ Review compliance reports ▪ Audit vulnerability resolution

AppScan Enterprise + AppScan Source: **Static analysis (white box) security & quality testing in the collaborative application lifecycle**

Source Code Analysis for Security Testing in Development & Build Automation

Broad Application Support

- **Out of the Box for Security Testing**
  - Java
  - JSP
  - C
  - C++
  - Classic ASP (VB6)
  - COBOL
  - SAP ABAP*
  - .NET
    - C#
    - VB.NET
    - ASP.NET
  - PHP
  - HTML
  - Perl
  - ColdFusion
  - Client-Side JavaScript
  - Server-Side JavaScript
  - VBScript
  - PL/SQL
  - T-SQL

Code Quality Static Analysis

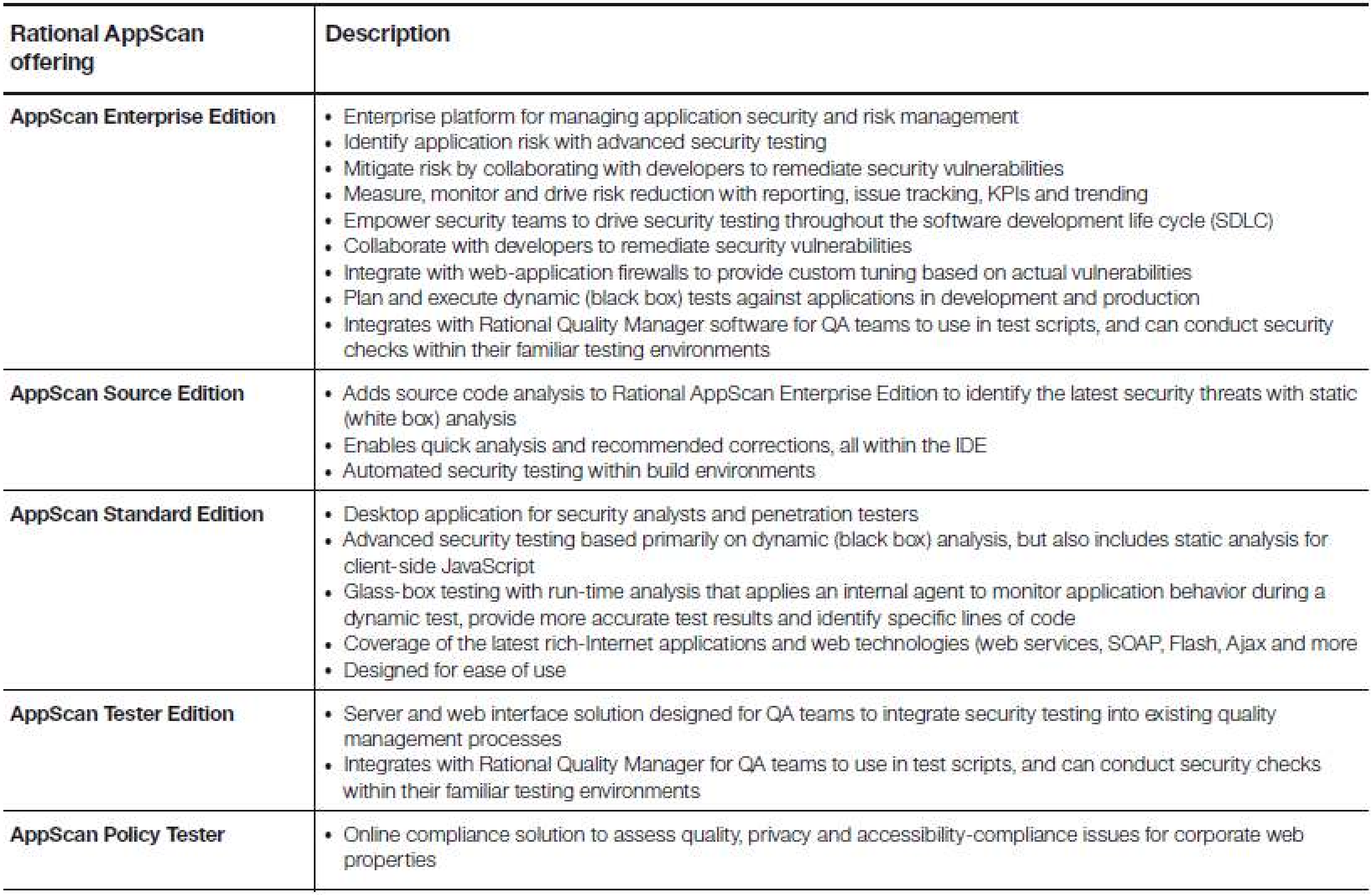
- Identify code-level quality defects within IDE
- Automate code quality analysis as part of the build process for centralized software code scanning
- Key Performance Indicators (KPIs) to help developers learn best practices
- Languages: Java, C, C++

Application Lifecycle Integrations

- **Develop**
  - IDE plug-ins to remediate identified issues (*Source for Remediation*)
  - Options to scan code locally from IDE (*Source for Developer*)
- **Build**
  - Automatically trigger security scans with each build (*Source for Automation*)
  - Review results from IDE or Security user & create work items for remediation
- **Security**
  - *Source for Security* power user creates SAST scans executed from IDE or in build automation
  - Executes advanced scans in pre-production security audits

* Requires Virtual Forge CodeProfiler for AppScan Source Edition

IBM Rational Appscan portfolio summary



Rational AppScan offering	Description
AppScan Enterprise Edition	• Enterprise platform for managing application security and risk management • Identify application risk with advanced security testing • Mitigate risk by collaborating with developers to remediate security vulnerabilities • Measure, monitor and drive risk reduction with reporting, issue tracking, KPIs and trending • Empower security teams to drive security testing throughout the software development life cycle (SDLC) • Collaborate with developers to remediate security vulnerabilities • Integrate with web-application firewalls to provide custom tuning based on actual vulnerabilities • Plan and execute dynamic (black box) tests against applications in development and production • Integrates with Rational Quality Manager software for QA teams to use in test scripts, and can conduct security checks within their familiar testing environments
AppScan Source Edition	• Adds source code analysis to Rational AppScan Enterprise Edition to identify the latest security threats with static (white box) analysis • Enables quick analysis and recommended corrections, all within the IDE • Automated security testing within build environments
AppScan Standard Edition	• Desktop application for security analysts and penetration testers • Advanced security testing based primarily on dynamic (black box) analysis, but also includes static analysis for client-side JavaScript • Glass-box testing with run-time analysis that applies an internal agent to monitor application behavior during a dynamic test, provide more accurate test results and identify specific lines of code • Coverage of the latest rich-Internet applications and web technologies (web services, SOAP, Flash, Ajax and more • Designed for ease of use
AppScan Tester Edition	• Server and web interface solution designed for QA teams to integrate security testing into existing quality management processes • Integrates with Rational Quality Manager for QA teams to use in test scripts, and can conduct security checks within their familiar testing environments
AppScan Policy Tester	• Online compliance solution to assess quality, privacy and accessibility-compliance issues for corporate web properties