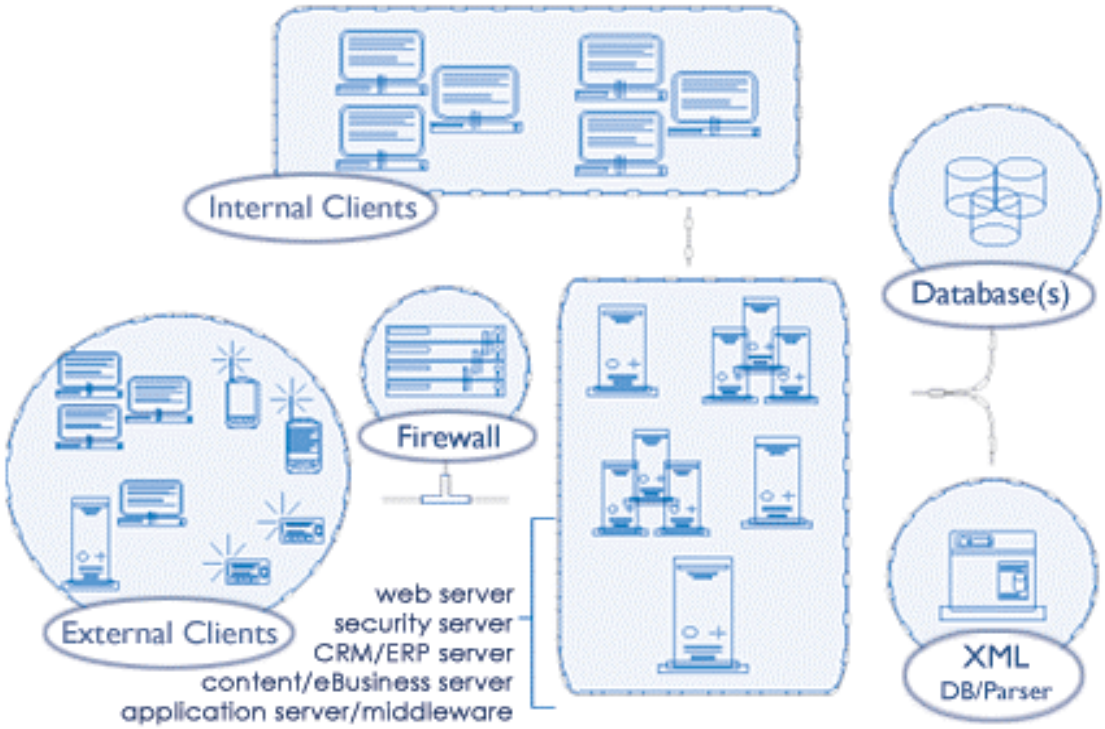
End-to-End Testing of IT Architecture and Applications



Authored by:

Jeffrey Bocarsly, PhD.
Division Manager
Automated Functional Testing
RTTS

Jonathan Harris
Division Manager
Scalability Testing
RTTS

Bill Hayduk
Director
Professional Services
RTTS

55 John Street • 12th Floor • New York, NY 10038-3712 • **phone** 212.240.9050 • **fax** 212.240.9020

20 Park Plaza • 4th Floor • Boston, MA 02116 • **phone** 617.948.2106 • **fax** 617.249.0190

RTTS
www.rttsweb.com

End-to-End Architecture Testing refers to the concept of testing at all points-of-access in a computing environment. For performance and scalability testing these include:

1. hardware
2. operating systems,
3. applications
4. databases and
5. network

In regard to functionality testing, points-of-access include:

1. the front-end client,
2. middle tier
3. content sources and
4. back-end databases

With this in mind, 'architecture' is the term that defines how all of the components in the environment interrelate, how they interact with other components in the environment and how users interact with them. The specific architecture that organizes components defines their strengths and weaknesses. The uncertainty in how an architecture will respond to the demands placed on it creates the need for End-to-End Architecture Testing.

Complex architectures

Typical heterogeneous computing environments today are a complex mix of legacy, home grown, third party and standardized components and code. With the advent of the Web, architectures have increased in complexity, often with a content-tier placed between back-end database(s) and the user-oriented presentation-tier. The content-tier may deliver content from multiple services that are brought together in the presentation-tier, and may also contain business logic that previously would have been found in the front-end of a client-server system.

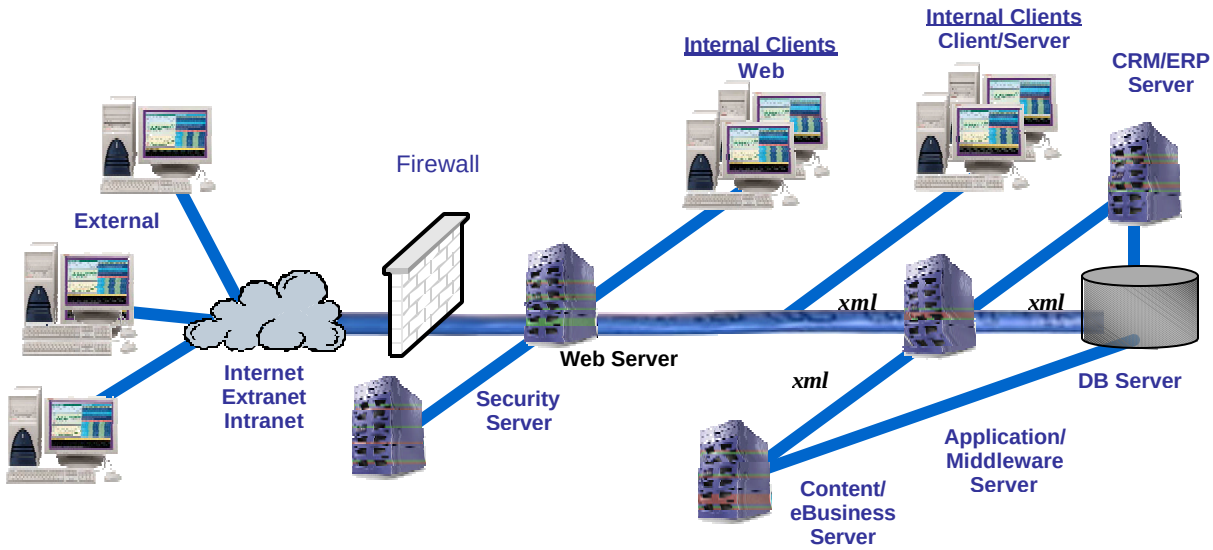


Figure 1 – typical architecture

55 John Street • 12th Floor • New York, NY 10038-3712 • **phone** 212.240.9050
20 Park Plaza • 4th Floor • Boston, MA 02116 • **phone** 617.948.2106

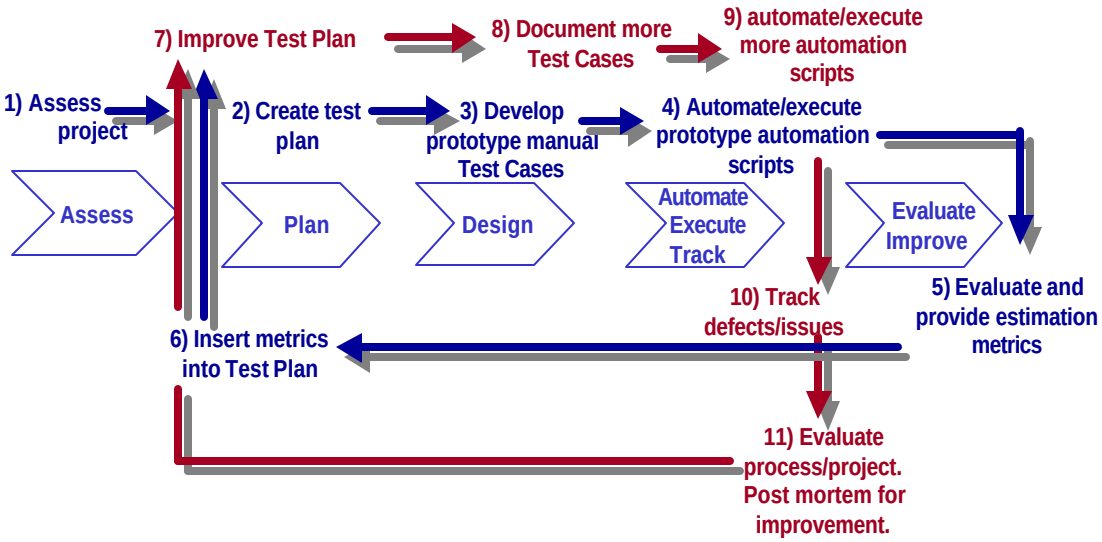
For both functional and scalability issues, industry-standard testing practices, which evolved in response to quality issues facing the client-server architecture, have centered either on the front end (functional) or the back end (scalability/performance). This 'division of labor' derived largely from the fact that the classic client-server architecture, a 2-tier structure, is not complicated relative to current multi-tier and distributed environments. In the standard client-server arrangement, issues are either on the client side or on the database side. The current trend toward architectural complexity makes this simple testing scheme obsolete.

The increase in complexity, overlaid with the problems of integrating legacy and cutting-edge development, can make characterization, analysis and localization of software and system issues (functional *and* scalability/performance) major challenges in development and delivery of software systems.

An overall quality strategy

These moves toward architectural complexity suggest that new, aggressive quality enhancement strategies are necessary for successful software development and deployment. For control of scalability behavior, the most potent strategy combines testing individual components of the environment with testing the environment as a whole.

Figure 2 - RTTS' Test Automation Process (TAP)



These parallel modes of analysis aide in determining the strengths and weaknesses of the architecture, and in pinpointing which components must be involved in resolving the performance- and scalability-related issues of the architecture. An analogous strategy, full data integrity validation, is recommended for the management of functional quality. Because data may now derive from diverse sources, the validation of data integrity across the full architecture is becoming increasingly critical. Assessing data integrity (including any functional transformations of data that occur during processing) both within components and across component boundaries leads to the efficient localization of each potential defect, making the tasks of system integration and defect isolation part of the standard process of development.

For every question that can be asked about how an environment scales or performs, a test can be created to answer that question.

- How many users can access the system simultaneously and still maintain acceptable response time?
- Will my high-availability architecture work as designed?
- What will happen if we add a new application or update the one I currently use?
- What should the configuration be to support the number of users we expect at launch, in 6 months and in 1 year?
- We only have partial functionality – is the design sound?

For questions pertaining to functionality (UI presentation, business logic, component data handling, system integration and system data integrity), an analogous set of testing strategies is available. Whether testing the components individually, as subsystems or as a whole, tests targeting each of these questions can be executed. End-to-end architecture testing exercises and analyzes the environment from the ground up.

The component level

End-to-end architecture scalability testing begins with *Component Testing* where each system within the environment is exercised to determine its transaction (or volume) limitations. Functional testing applied at this level validates the transactions that each component performs. This includes any data transformations the component is required to perform, as well as validations of business logic that apply to any transaction handled by the component. Both functional and scalability testing at the component level are useful as diagnostics when the environment is being built.

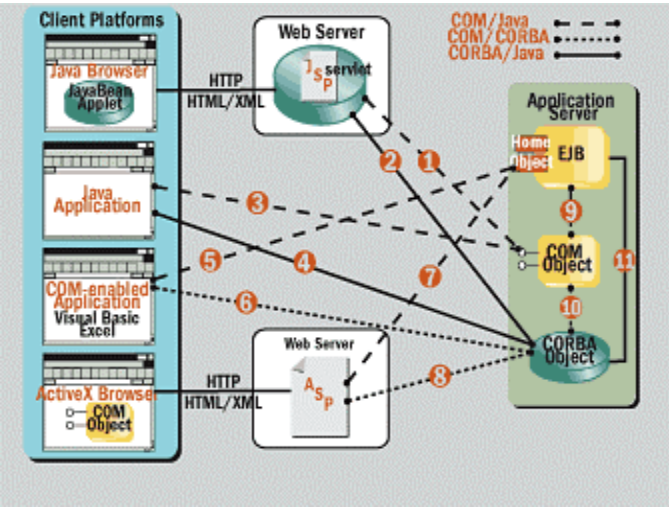


Figure 3 – component interoperability (from ADTmagazine)

As application functionality is written, *Infrastructure Testing* verifies and quantifies the flow of data through the environment. Once enough application functionality exists to create business related transactions, *Resource Testing* and *Transaction Characterization Testing* quantify the system usage by transaction type. Modifying hardware, operating system, software, network, database or other configurations you can achieve optimal performance through *Configuration Testing*. In parallel to these scalability checks as the system is assembled, data integrity must be verified as data begins to be passed between system components.

55 John Street • 12th Floor • New York, NY 10038-3712 • **phone** 212.240.9050
20 Park Plaza • 4th Floor • Boston, MA 02116 • **phone** 617.948.2106

The full environment

When the system has been fully assembled, whole environment testing can be initiated. One of the first issues that must be considered is that of integration.

Integration Testing addresses the broad issue of whether the system is integrated from a data perspective. That is, are the components that should be talking with one another communicating properly? If they are, are the proper data being transmitted between them? In favorable situations, data may be accessed at intermediary stages of transmission between system components. These points may occur, for example, when data is written to temporary database tables, or is accessible in message queues prior to access by target components.

Access to data at these component boundaries can provide an important additional dimension to data integrity validation characterization of data issues; in cases where data corruption can be isolated between two data transmission points, the defective component has been localized between those points.

In the area of system capacity, whole environment testing begins with *Scalability Testing*. Scalability testing is designed to determine the upper limitations of transaction and user volume without exceeding defined performance requirements such as user response time.

Performance Testing and additional *Configuration Testing* may improve the overall performance and will provide tradeoff information of specific hardware or software settings.

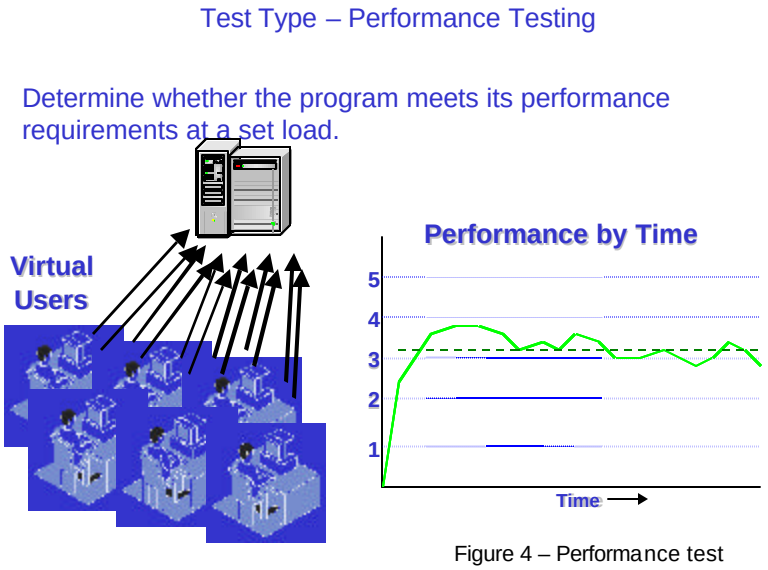


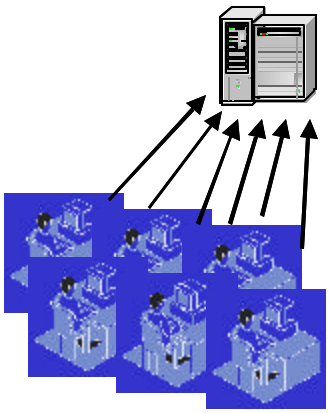
Figure 4 – Performance test

55 John Street • 12th Floor • New York, NY 10038-3712 • **phone** 212.240.9050
20 Park Plaza • 4th Floor • Boston, MA 02116 • **phone** 617.948.2106

Concurrency Testing will profile the effects of database locking and deadlocking as well as single threaded code execution.

Test Type – Concurrence Testing

Determine the effects of locking and deadlocking.



- Row Level Locking
- Page Level Locking
- Table Locking
- Single Threaded Processing
- Critical Path Processing

Figure 5 – Concurrency test

Stress and Volume Testing are performed to test the resiliency of you environment to withstand burst or sustained high volume activity without failing due to memory leaks or queue over-runs.

Test Type – Stress Testing

Perform repetitive high volume transactions in an attempt to break the system (i.e. memory leaks, queue over-runs)

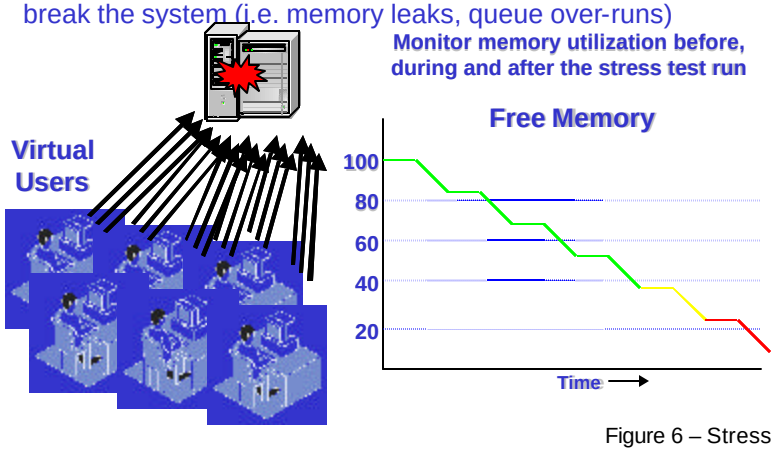


Figure 6 – Stress test

Reliability Testing exercises your environment at a sustained 75% to 90% utilization over an extended time to simulate production.

Lastly as part of whole environment testing, *Failover Testing* is done to test the high-availability functionality within the environment, should specific components fail. Failover testing answers the question of whether users will be able to continue accessing and processing with minimal interruption if a given component fails.

Test Type – Failover Testing

Does the high availability architecture in place operate as designed?

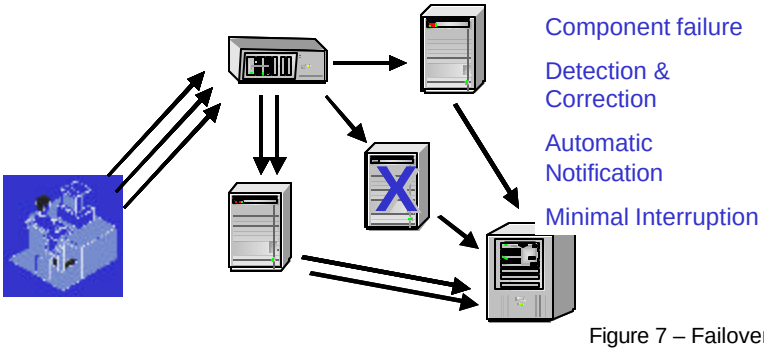


Figure 7 – Failover test

If an environment employs third party software or accepts feeds from outside sources or hosted vendors, then *SLA Testing* (Service Level Agreement) testing can be conducted to ensure end-user response times and inbound and outbound data streams are within contract specifications. Once external data or software sources are in place, monitoring of these sources on an ongoing basis is advisable, so that corrective action can be taken if problems develop, minimizing the effect on end users. There are many tools available to assist in Service Level Management through Web site monitoring that RTTS can assist in implementing.

Modeling the architecture

As part of the process of design, the ability to model different architectures can help make network designs more efficient and less error-prone prior to the acquisition of hardware or the implementation of software. Once the test environment has been optimized and verified but prior to deployment, network infrastructure modeling can help pinpoint inconsistencies and errors in routing tables and configurations. Additionally, application transaction characterizations obtained during testing can be input into the model to identify and isolate application chattiness and potential bottleneck points within the infrastructure.

Conclusion

The end-to-end Architecture Testing strategy described here exercises and analyzes computing environments from a broad-based quality perspective. The scalability and functionality of every component is tested individually and collectively during development and in prerelease quality evaluation to provide both diagnostic information to enhance development efficiency and a high degree of quality assurance upon release. For management of the trend toward architectural complexity and distributed computing, the strategy of end-to-end Architecture Testing provides a robust solution.

55 John Street • 12th Floor • New York, NY 10038-3712 • **phone** 212.240.9050
20 Park Plaza • 4th Floor • Boston, MA 02116 • **phone** 617.948.2106

About the authors:

Bill Hayduk, president and director of professional services, has an excellent reputation in the technology field and is particularly noted for his methodology and automation expertise. Over the last 19 years, Bill has successfully implemented large-scale projects at many Fortune 500 firms. He has worked in various sectors including global banks, brokerage firms, multimedia conglomerates, pharmaceutical, and insurance and reinsurance companies.

Bill holds an MS degree in Computer Information Systems from the Zicklin School of Business (Baruch College) and a BA in Economics from Villanova University. He has been a selected speaker at industry-specific trade conferences, as well as a source of information for corporations and has been referenced in many industry trade publications.

Jeff Bocarsly, the functional testing division manager at RTTS, has successfully implemented automated software testing projects at many Fortune 500 firms. His experience includes project in various sectors including brokerage firms, media, pharmaceutical, software, banking, insurance, and reinsurance companies. He specializes in implementing test automation and methodology solutions for software development groups.

Jeff holds a BS from UCLA and Masters and PhD degrees from Columbia University.

Jonathan Harris, performance and scalability division manager for RTTS, has planned, executed and analyzed data for over 350 performance evaluations within every major market segment. With over 13 years of experience in programming and testing, he has spearheaded the development and implementation of RTTS' proprietary scalability testing.

Previously, Jon worked as a lead testing consultant for *Promark, Inc.* where he developed the front-end interface to the *Promark Robot*, now *Compuware*'s *QA Load*. Jon studied biology and computer science at Carnegie-Mellon University.

About RTTS

RTTS is a professional services organization that specializes in the testing of IT applications and architecture. With offices in New York and Boston, RTTS has been serving Fortune 500 and mid-sized companies nationwide since 1996. RTTS draws on its expertise utilizing best-of breed products, expert test engineers and proven methodology to provide the foremost end-to-end solution that ensures application functionality, reliability, scalability and network performance. To learn more about RTTS, visit www.rttsweb.com or contact:

Ron Axelrod, Director of Business Development
Tel: (212) 240-9050 x17
e-mail: raxelrod@rttsweb.com