Web Services Testing

Mark Lewis-Prazen
Web Services
Fall, 2006

Outline

- Web Services Proliferation
- Exploring Testing Issues
- Web Service Testing Challenges
- Functionality Testing Challenges
- Publish/Find/Bind Testing Challenges
- Security Testing Challenges
- Performance Testing Challenges
- Web Service Testing Tools
- Information Threads

Web Services Proliferation – and the Changing Testing Landscape

- Moving from a relatively small number of large apps to a relatively large number of small apps
- As Web services adoption rises, more developers are doing more testing – early and often in Development vs QA
- The asynchronous nature of the business process services will make the QA job less grunt work and more of an intellectual challenge (a prediction or a QA optimist working on a self promotion campaign?)
- "No matter how easy it is to invoke WSDL, if you don't know what the object was supposed to do, I don't believe you can test it." Advocate for developer testing of Web services.

Exploring Testing Issues

- A key to testing Web services is ensuring their functional quality, because when you string together a set of services, you introduce many more opportunities for error or failure
- Developers are typically poorly versed in security – coding scrutiny driven by performance issues
- Few development organizations within enterprise IT shops understand need for vulnerability testing
- Testing was the last bastion of the waterfall method. But even here the concept of a “freeze” is ending; testing is becoming a continuous activity.

Exploring Testing Issues (cont'd).

Category	Traditional App	Web Service App
Invocation	Testing tools invoke the application and take a look at what it returns	More synchronous invocation of [multiple] services … challenges
Performance Testing	Performance testing typically back loaded exercise (if done at all)	Do performance testing early; as you construct services and deploy them, performance of service is going to roll out across app
QA Skills	Sufficient for running testing of COTS applications - the degree of such expertise in the QA area is typically highly functional in nature	Lack the skills to test; testing spans multiple technologies; the "interface freeze" syndrome
Portfolio Risk & Skills Mis-match	Fairly static environment	Dynamic environment; more small apps requires diverse skill sets

Exploring Testing Issues (cont'd).

Category	Traditional App	Web Service App
White box testing	Code knowledge available	No knowledge; services are just interfaces; white box testing not an option
Mutation testing	Seed code with errors for testing	No access to code; hence no opportunity to seed code; mutation testing not an option
Infrastructure Control (COTS framework)	App integrated into the user system infrastructure	App lives in a foreign infrastructure – implication for testing is to guarantee the SLAs with customers; different stakeholders may want svcs tested
Service Release Control	Service release strategy is known by user and systems integrator	Provider controls service release strategy … doesn't know all users; changes may not be evident from the interface

Exploring Testing Issues (cont'd).

Table 1. Highlights per testing dimension. Needs and responsibilities of each stakeholder are in black, advantages in green, issues and problems in red.

Testing levels	Testing perspectives: Developer	Provider	Integrator	Third-party	User
Service functional testing	White-box testing available Service specification available to generate test cases Nonrepresentative inputs	Needs service specification to generate test cases Limited cost Black-box testing only Nonrepresentative inputs	Needs service specification Needs test suite deployed by service provider Black-box testing only High cost	Assesses only selected services and functionality on behalf of someone else (should be impartial assessment of all services) Small resource use for provider (one certifier tests the service instead of many ntegrators) Only nonrepresentative inputs High cost	Service-centric application self-testing to check that it ensures functionality during runtime Services have no interface to allow user testing
Service nonfunctional testing	Necessary to provide nonfunctional specifications to provider and consumers Limited cost Nonrealistic testing environment	Necessary to check own ability to meet SLA stipulated with user Limited cost Testing environment might not be realistic	High cost Might depend on network configuration Difficult to check whether SLA is met	Assesses performance on behalf of someone else Small resource use for provider (one certifier tests the service instead of many integrators) Nonrealistic testing environment High cost	Service-centric application self-testing to check that it ensures performance during runtime
Integration testing	Can be service integrator on his own	NA	Service call coupling increases because of dynamic binding Must regression test a composition after reconfiguration or rebinding Quality-of-service testing must consider all possible bindings	NA	NA
Regression testing	Limited cost (service can be tested off-line) Unaware of who uses the service	Limited cost (service can be tested off-line) Can be aware that the service has changed but unaware of how it changed	Might be unaware that the service has changed High cost	Retests service during its lifetime only on behalf of integrators, not other stakeholders High cost Lower-bandwidth use than having many integrators Nonrealistic regression test suite	Service-centric application self-testing to check that it works during evolution

Possible Opportunities

- A need for more close and effective collaboration between developers and end users ……. ultimately more robust software ???????

- Opportunities for functional and QA areas to develop more flexible skills more in line with the trends in application development and implementation

New Risks Introduced

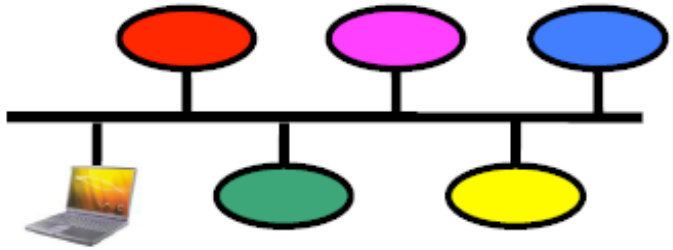
- API risk – Services are being used by many applications
- Service Version risk – In event of service upgrade, if one user of a service does not upgrade, then more than one version needs to be managed/maintained/tested/etc.

Web Service/Intermediary Synopsis

- Developers usually build mock Web service environments
- Emulate the myriad of client requests of server via test scripts including vulnerability tests
- Focus of new vendor test tools is to try to quickly and easily emulate any endpoint (client or server) of a Web service
- Point such tools to a valid WSDL and emulate both the client and server endpoints simultaneously
- Verify that the Intermediary (Soap, security, etc).handles the requests and responses as expected and policies are accurately reflected

Web Service Testing Challenges

- Since Web services are composed of loosely coupled distributed over networks, we must test the application:



- end to end;

- service by service;

- and interface by interface.

Functionality Testing Challenges

- Overall functionality of web services should be easy to test
- BUT, only if we thoroughly trust the applications components (services) before we combine them to complete the application
- Implication is that building from lower defect components should mean a smoother testing process, EXCEPT…….

Functionality Testing Challenges (cont'd).

- Web services have more APIs (one per service) and increased communication paths between those services
- Increased level of integration and interoperability testing
- Who owns this testing? Service provider? Service requester? Both? Others?
- How does trust get established?

Publish, Find and Bind Testing Challenges

- Service providers must advertise their existence to brokers
- Brokers must register the above and provide information of these services through search functions
- Service requesters must find the needed providers and bind to them to consume their services

Publish, Find and Bind Testing Challenges

- Similar to testing 3rd party credit card application processing thru a web app
- Are services able to register themselves?
- Can web app find and bind with services?
- Who owns the tests? Provider? Requestor?
- Trust issue?

Security Testing Challenges

- Web application is collection of independent services which come together to provide some value-added functionality
- Value suggests the need for security; for some authentication of users prior to service access
- Consider the case of an application with a myriad of services, each requiring different authentication procedure and enforcing different security policies ……. a testing challenge of significant dimensions

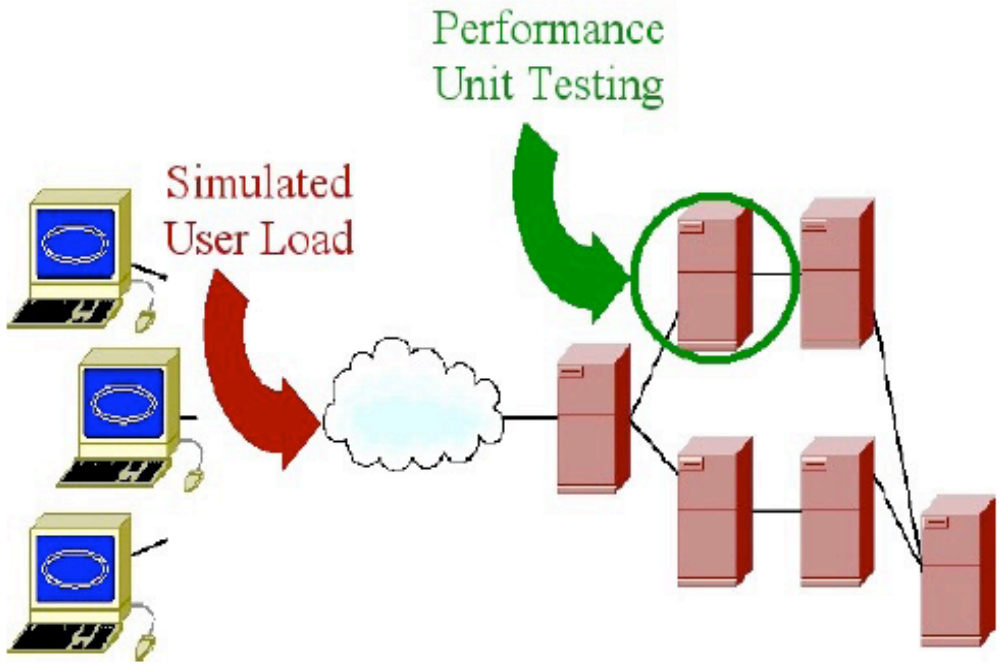
Performance Testing Challenges

- All these loosely-coupled, platform-independent, highly scalable services are not free
- Major performance problems typically are a result of :
  - Large services without adequate hardware support
  - Small services with significant overhead
  - Layer on layer; abstraction on abstraction
  - Services distributed on a network with its own latency

Performance Testing Challenges (cont'd).

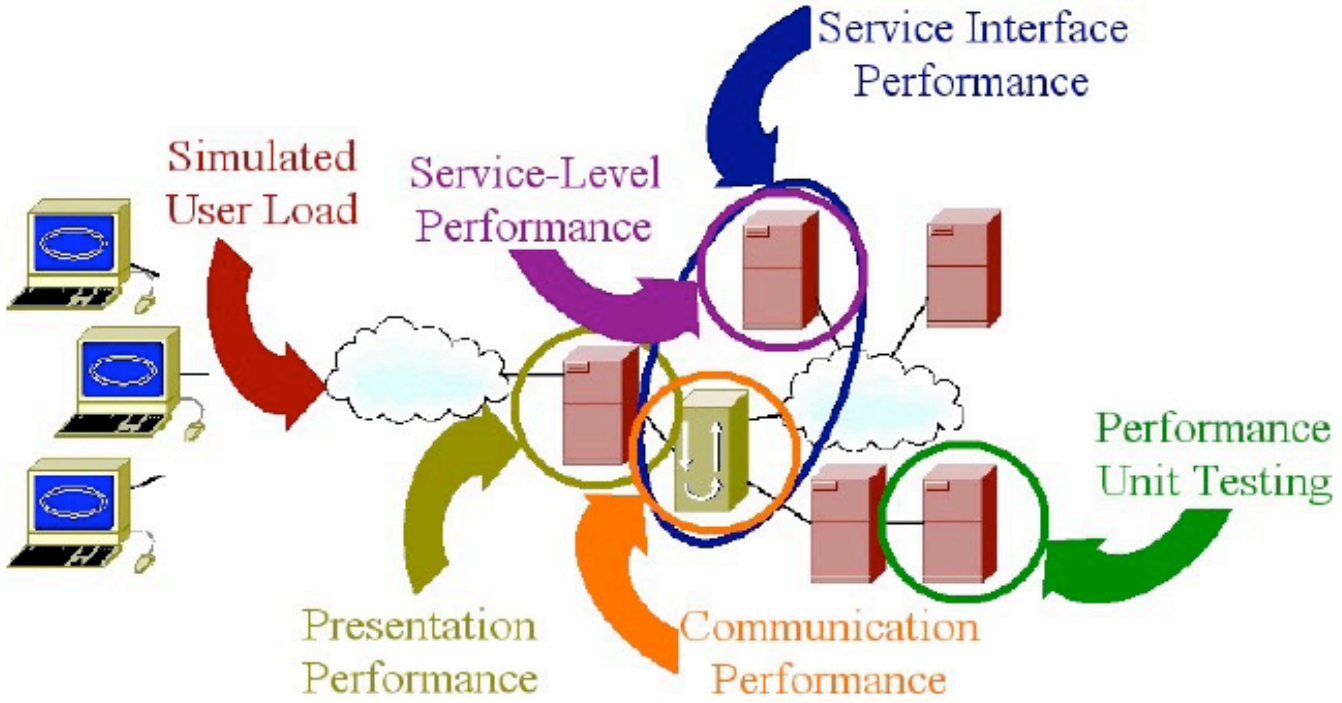
- Application needs to be performance tested in the following manner:
  - End to end from the requester perspective
  - At the unit level during development (by provider)
  - At service level (generally by requester and provider)
  - Interface validation (generally by requester and provider)
  - To ensure functionality under boundary load conditions

Performance Testing Challenges (cont'd).



What we've become accustomed to.

Performance Testing Challenges (cont'd).



What we should have been doing all along...

And with web services have little choice !

In Summary - Web Service Testing Challenges

- Who owns the testing of the services?
  - Requestor
  - Provider
  - Both/Another?
- How well did the vendor test?
  - How do you know?
- How do you establish trust in a service and demonstrate to users that our web app is worthy of their trust?

Some Web Service Testing Tools

- Optimyz - WebServiceTester is an end-to-end product offering automatic test generation; functional, regression, and load testing; conformance testing against WS-I Profiles, BPEL-based orchestration testing; secure Web services testing; and debugging and diagnostics.
- Mercury(now HP) - "end to end" solution for Web services testing in the form of three offerings: LoadRunner, QuickTest Professional and Business Process Testing, its newest tool that sits on top of LoadRunner.
- Empirix Inc. -- e-TEST: e-Manager Enterprise, test management; e-Tester, functional testing; e-Load, scalability testing.
- Parasoft -- SOAPtest, WSDL validation, unit and functional testing of the client and server, performance testing
- IBM Rational Software Co. -- TestStudio, unit, functionality, performance, and load testing; PurifyPlus, runtime analysis tool for detects memory and performance bottlenecks early in the development cycle

Some Example Testing Tools

- MindReef
  - http://www.mindreef.com
- IBM
  - http://demos.dfw.ibm.com/on_demand/Demo/IBM_Demo_Rational_ClearQuest_Test_Management-Jun06.html?S=SWCAT
- Mercury provides a service called ActiveTest that allows it to populate a Web service with real data loads from its server farm. Load runner has been enhanced to send same data to client and server and test for stresses.

Other Information Threads

- http://ieeexplore.ieee.org/iel5/6294/34167/01628907.pdf ; Testing Services and Service-Centric Systems, Canfor and DiPenta, IT Pro, March/April 2006.
- http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci1085779,00.html; Web Services Tools Mature, May 2005.
- http://www.aptest.com/resources.html - a web services test portal.
- http://www.softwaremag.com/L.cfm?Doc=2005-09/2005-09testing. Testing and QAS in a Web Services World, Software Magazine, Sept. 2005.
- www.ibm.com/software/awdtools/tester/clearquest/functionaltest/index.html - IBM Rational ClearQuest and Functional Testing
- The Forrester Wave: Functional Testing Solutions, Q2 2006
- http://www.developers.net/external/1291- Developers Network
- http://sourceforge.net/projects/xmltester
- http://www.soapui.org/index.html