



IBM Software Group

# Testing SOA Applications: A Guide for Functional Testers

*Brian Bryson, IBM*  
*bbryson@ca.ibm.com*

**Rational.** software

[Go to IBM](#)

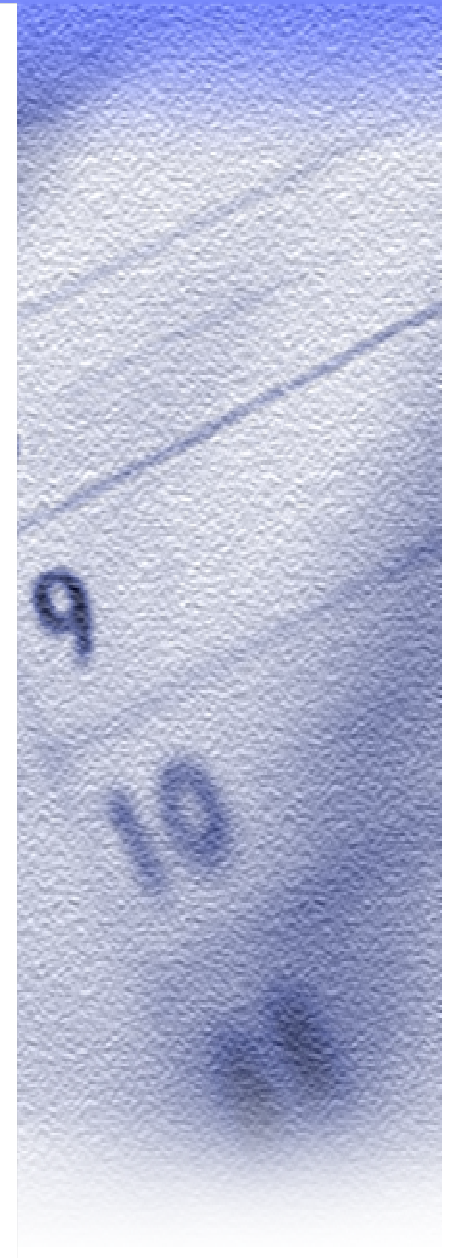
## Session Objective

- ▶ Understand implications of SOA architecture on QA and Test professionals
- Explore strategies for testing SOA based applications



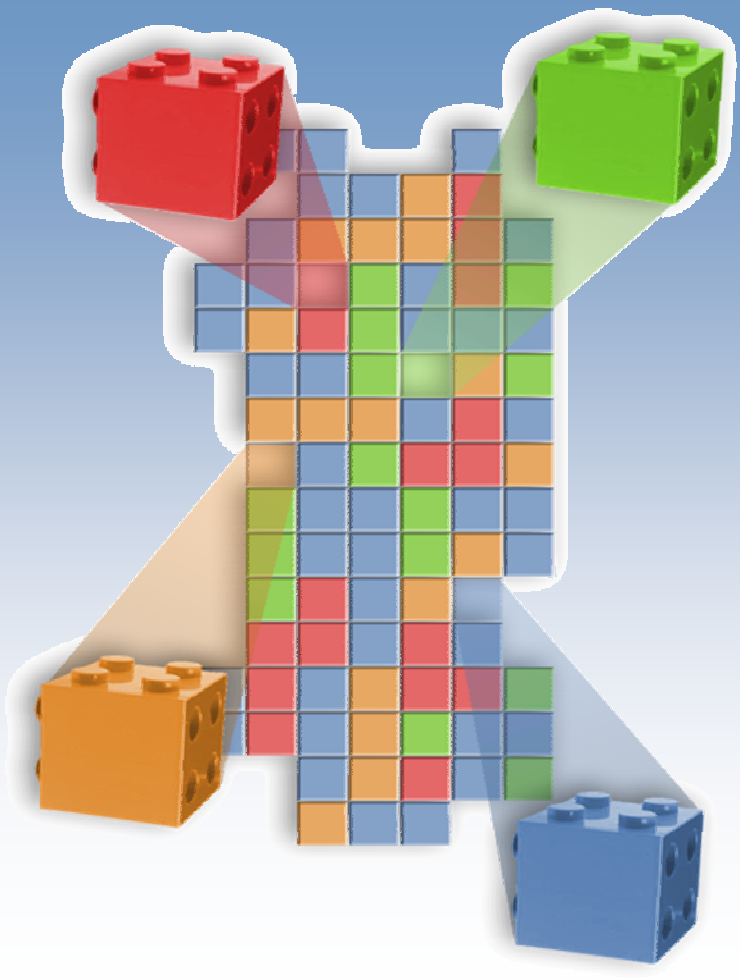
# Agenda

- **SOA Architecture Overview**
- **Demo:**  
**Building and Deploying a Web Service**
- **Testing SOA Applications**
  - ▶ **Challenges**
  - ▶ **Strategies**
- **Demo:**  
**Testing a Web Service**
- **Summary Lessons Learned in the Field**





# 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.**

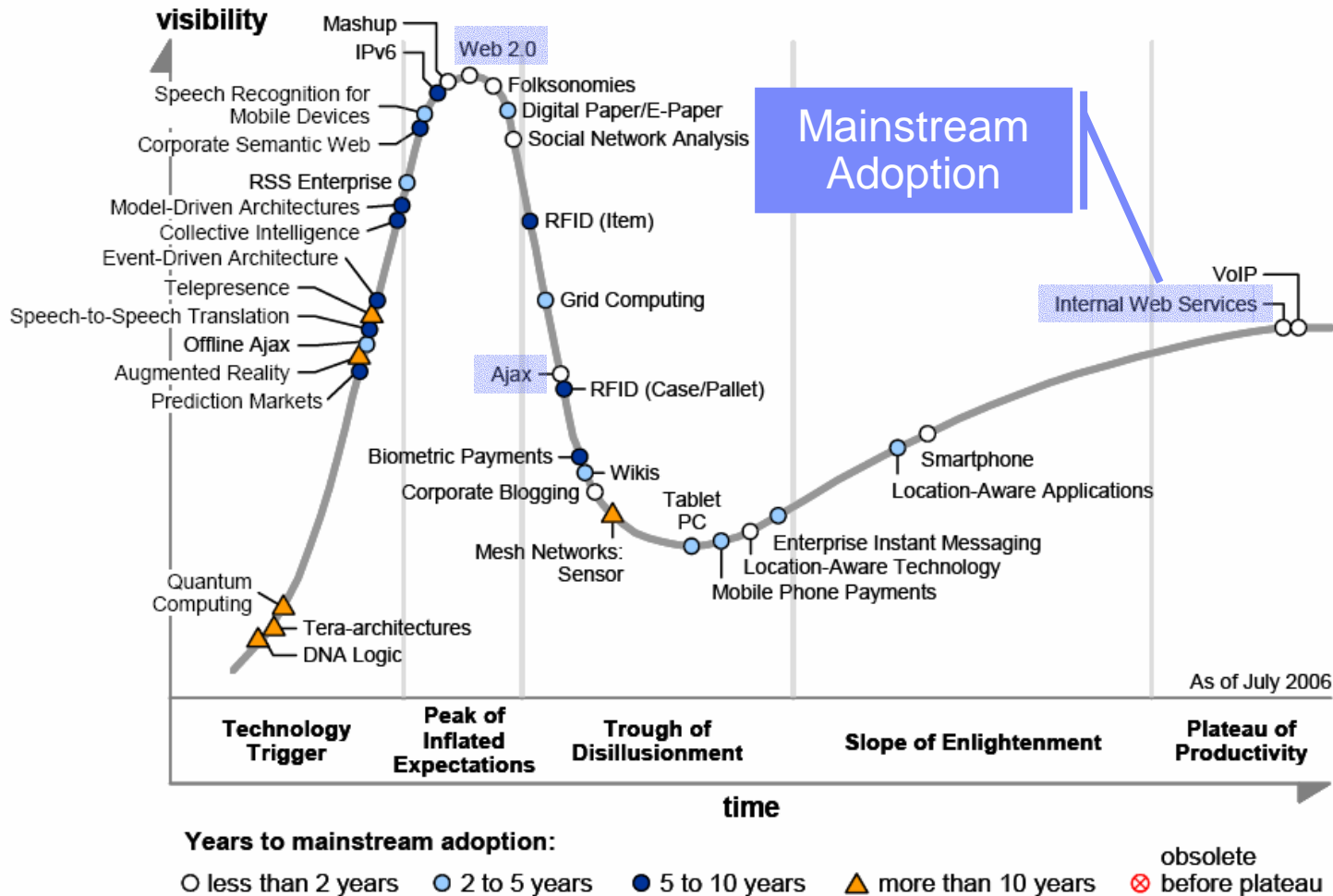
Period.





# SOA Adoption: Where are we?

Figure 1. Hype Cycle for Emerging Technologies, 2006



Source: Gartner (July 2006)



# Fact-or-Fiction: SOA in early stages?

Online Banking



Teller  
FSM Desktop  
Cheque Imaging

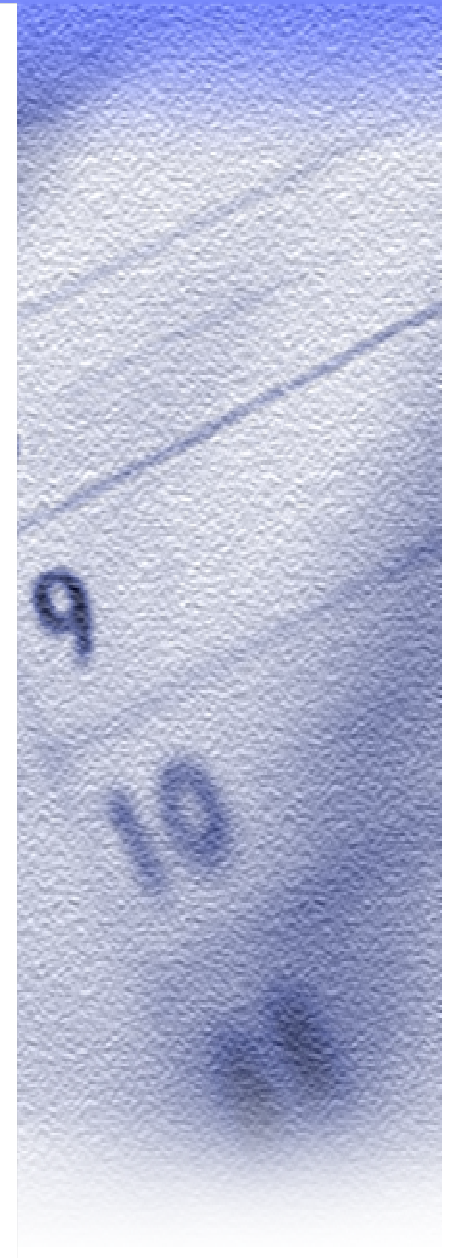


Order Management



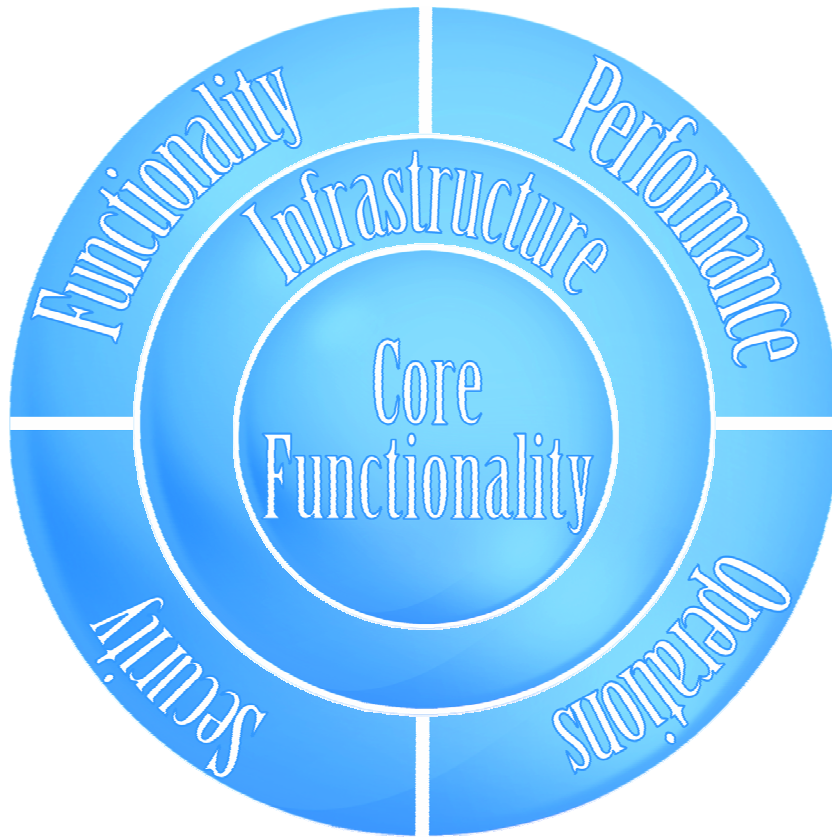
# Agenda

- **SOA Architecture Overview**
- **Demo:**  
**Building and Deploying a Web Service**
- **Testing SOA Applications**
  - ▶ **Challenges**
  - ▶ **Strategies**
- **Demo:**  
**Testing a Web Service**
- **Summary Lessons Learned in the Field**





If you only remember one thing...

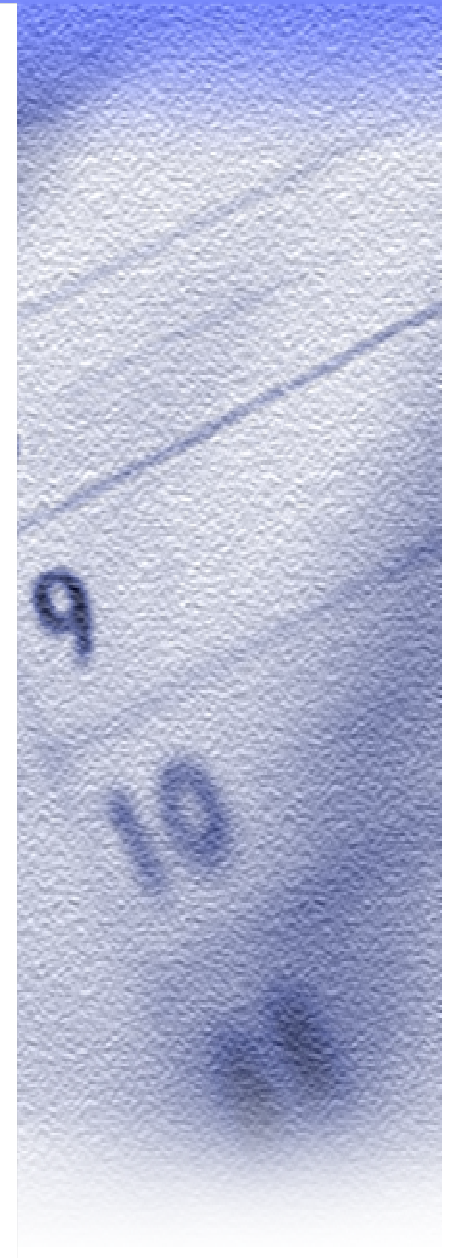


Web Services are like Ogres – They Have Layers



# Agenda

- **SOA Architecture Overview**
- **Demo:**  
**Building and Deploying a Web Service**
- **Testing SOA Applications**
  - ▶ Challenges
  - ▶ Strategies
- **Demo:**  
**Testing a Web Service**
- **Summary Lessons Learned in the Field**



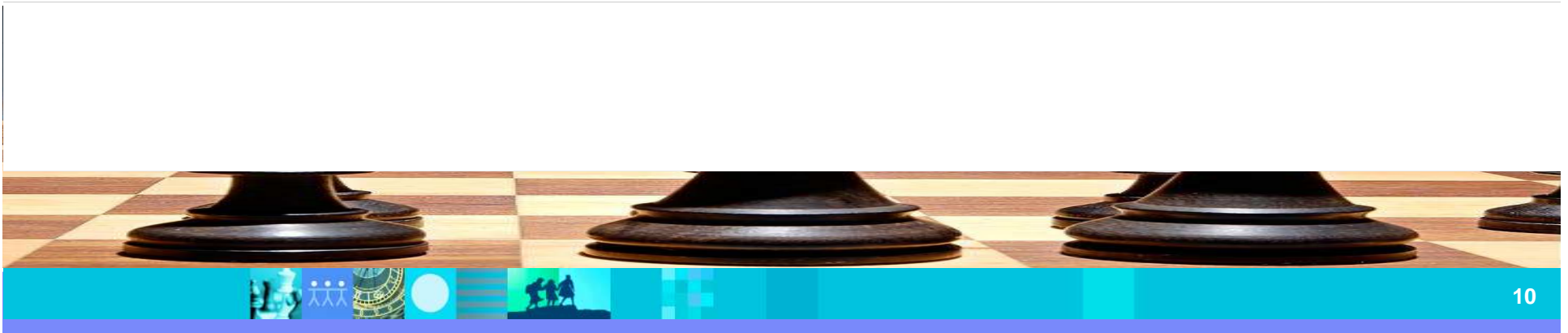
# Unique Challenges of SOA Testing

- **Headless Testing / GUI-less Testing**

- ▶ No client interface to interact with
- ▶ Similar to testing via API

- **Re-Use: Intended and Unintended**

- ▶ A single low-quality service can have broad impact
- ▶ No single entity owns the end-to-end flow, implies greater need for SMEs on test team
- ▶ Important to incorporate maximum data permutations and combinations
- ▶ Re-Use means Re-Test





## Web Service Testing vs. Functional GUI Testing

The screenshot shows two parts of the website. On the left, the 'outfitmycar' logo is displayed above a text box that reads: 'Select your car, and we'll take you to a customized homepage featuring: > gear that'll fit in or work with your car, truck or van > parts, accessories, and tools > vehicle-specific info and installation guides. You'll find everything from stereos and speakers to iPod adapters and exhaust systems.' On the right, the 'CRUTCHFIELD Car Selector' interface is shown, featuring a car icon and a dropdown menu labeled 'Select Year'. Below the dropdown, there is a section titled 'Can't find your exact car?' with text: 'Please [tell us about it](#). Even with the world's most extensive car installation database (over 12,500 vehicles), we haven't covered them all. If it's manufactured for sale in the United States, we'll add your car to our to-do list. Although our experts don't have specific info on your car, they may be able to help you brainstorm options. Our Sales Advisors can be reached at 1-888-955-6000, 8am - midnight, Eastern.'

- This is an online car audio vendor site
- The car selector lets you define your year, make and model so the site can determine which stereo equipment will fit into your vehicle.



## Web Service Testing vs. Functional GUI Testing

**outfitmycar<sup>SM</sup>**

Select your car, and we'll take you to a customized homepage featuring:

- › gear that'll fit in or work with your car, truck or van
- › parts, accessories, and tools
- › vehicle-specific info and installation guides.

You'll find everything from stereos and speakers to iPod adapters and exhaust systems.

**CRUTCHFIELD Car Selector**

Year: 2003

Select: Acura

Available models filtered by year

Can't find

Please tell us if you have a car model that isn't covered here. With the world's most extensive car installation database (over 12,500 models), we haven't covered them all. If it's manufactured for sale in the United States, we'll add your car to our to-do list.

Although our experts don't have specific info on your car, they may be able to help you brainstorm options.

Our Sales Advisors can be reached at **1-888-955-6000**, 8am - midnight, Eastern.

- Available models is filtered by year selected
- You cannot select a model that doesn't exist for a given year



## Web Service Testing vs. Functional GUI Testing

**outfitmycar<sup>SM</sup>**

Select your car, and we'll take you to a customized homepage featuring:

- › gear that'll fit in or work with your car, truck or van
- › parts, accessories, and tools
- › vehicle-specific info and installation guides.

You'll find everything from stereos and speakers to iPod adapters and exhaust systems.

**CRUTCHFIELD Car Selector**

Year: 2003  
Make: Acura  
Select: MDX

**Available makes filtered by model**

Model  
3.2CL  
3.2TL  
3.5RL  
MDX  
NSX  
RSX

**Can't find your car?**

Please [tell us](#) what you're looking for. Even with the world's most extensive car installation database (over 12,500 models), we haven't covered them all. If it's manufactured for sale in the United States, we'll add your car to our to-do list.

Although our experts don't have specific info on your car, they may be able to help you brainstorm options.

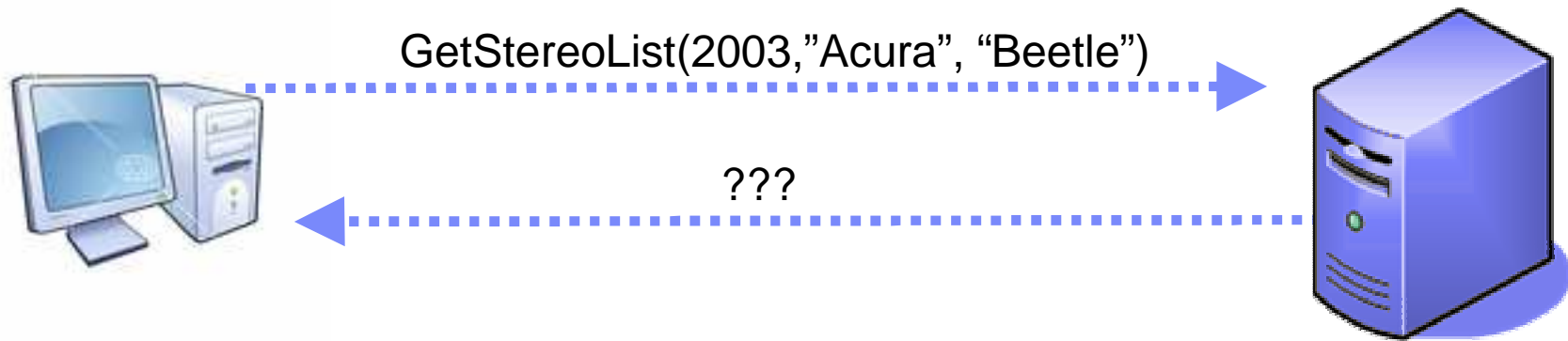
Our Sales Advisors can be reached at **1-888-955-6000**, 8am - midnight, Eastern.

- Available makes is filtered by model selected
- You cannot select a make that doesn't exist for a given model, ie: Impossible to select Acura Beetle





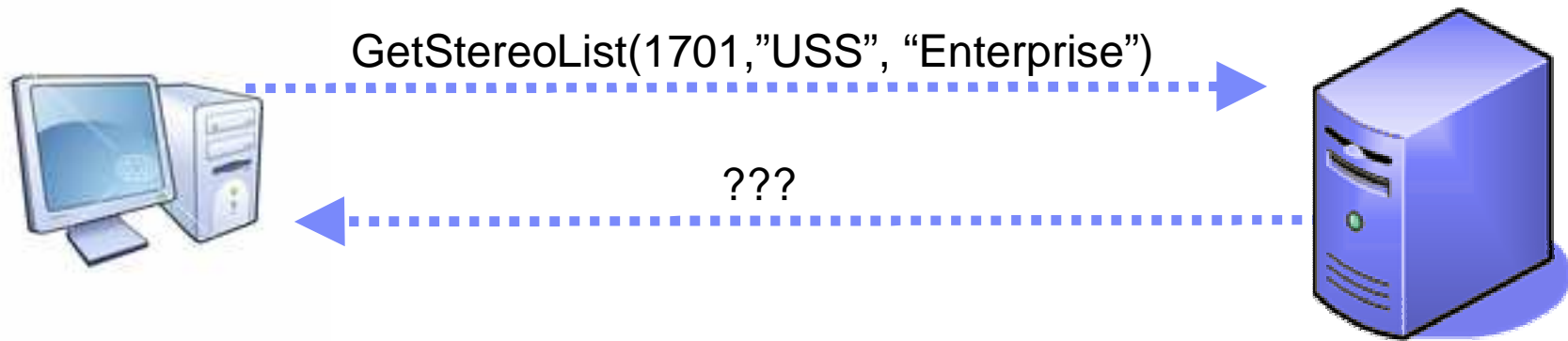
## Web Service Testing vs. Functional GUI Testing



- A web service has no preventative input validation
  - ▶ Nothing prevents you from sending invalid combinations of data



## Web Service Testing vs. Functional GUI Testing



- A web service has no input validation
  - ▶ Nothing prevents you from sending invalid combinations of data
  - ▶ Nothing prevents you from sending invalid data of any type
- Implication for testers
  - ▶ You need to ensure your web service is bulletproof



# Unique Challenges of SOA Testing

- **Open**
  - ▶ Security test scenarios take center stage
  
- **Ease of Access**
  - ▶ Can lead to spiky volume, requires rigorous performance testing
  
- **Loosely coupled**
  - ▶ The use of intermediaries such as ESB, Gateways/Proxies requires additional test scenarios and hardware resources/configurations.



# SOA Testing Challenges – Some Things Never Change

- **Automation is a must**

- ▶ **SOA accelerates pace of change**

- Manual testing can't keep up

- ▶ **SOA conceals use cases**

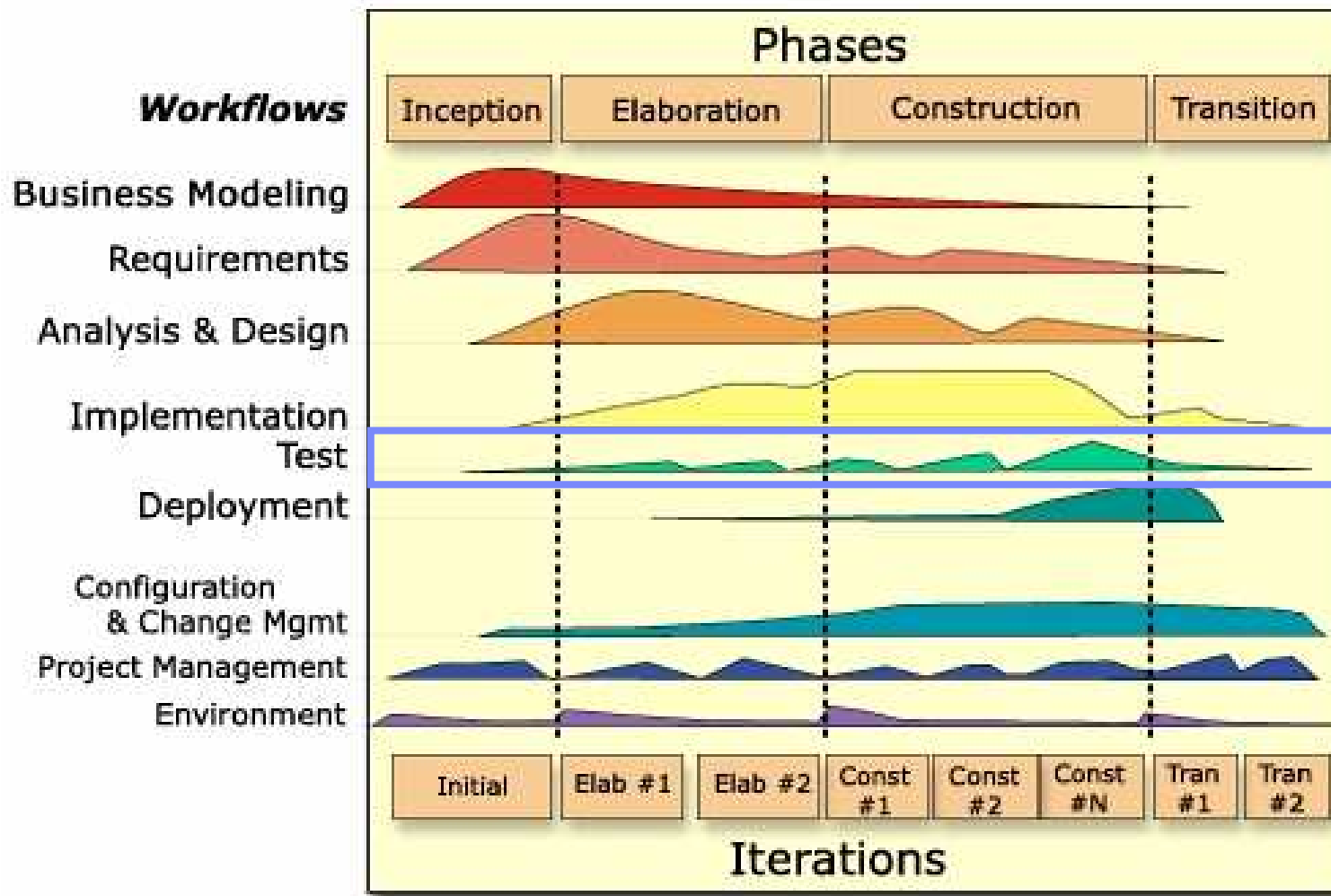
- Need for high volume of data permutations and combinations means multiple datasets per web service
    - Performance testing crucial to address unpredictable usage patterns
    - Security testing crucial to ensure transaction integrity

- ▶ **SOA encourages and enables early testing (yay!)**





# SOA Testing in Various Phases



# SOA Testing Activities by Phase

<b>Workflows</b>	<b>Phases</b>			
	Inception	Elaboration	Construction	Transition
Test				
Process Design				
Structure				
Function				
Security				
Performance				
Operations				



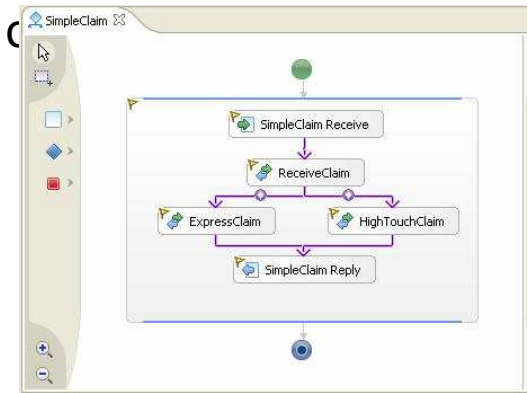
## Process: Design and Implementation

### ■ Elaboration / Construction

- ▶ Leverage information contained in BPEL model if available
- ▶ Use Process simulation to uncover “holes” in your process flow
- ▶ Test every fault and compensation handler
- ▶ Automate human tasks to allow regression testing

### ■ Transition

- ▶ Ensure interruptible flows can be resumed and failed c



## SOAP fault payload is platform-dependent

```
<soapenv:Fault>
  <faultcode>soapenv:Server.generalException</faultcode>
  <faultstring></faultstring>
  <detail>
    <ns1:DetailedFault xmlns:ns1="http://faulttest.test.ibm.com">
      <ns1:detailedInfo>myErrorDetails</ns1:detailedInfo>
    </ns1:DetailedFault>
    <ns2:exceptionName xmlns:ns2="http://xml.apache.org/axis/">
      com.ibm.test.faulttest.DetailedFault
    </ns2:exceptionName>
    <ns3:hostname xmlns:ns3="http://xml.apache.org/axis/">
      saturn
    </ns3:hostname>
  </detail>
</soapenv:Fault>
```



```
<soapenv:Fault>
  <faultcode xmlns:p25="http://faulttest.test.ibm.com">
    p25:DetailedFault
  </faultcode>

  <faultstring>
    <![CDATA[com.ibm.test.faulttest.DetailedFault]]>
  </faultstring>

  <detail encodingStyle="">
    <p25:DetailedFault xmlns:p25="http://faulttest.test.ibm.com">
      <p25:detailedInfo>myErrorDetails</p25:detailedInfo>
    </p25:DetailedFault>
  </detail>
</soapenv:Fault>
```




## Structure: Service Interface Design Testing



- Inception

- ▶ Verify ws-i compliance (soap stacks) at <http://www.ws-i.org/deliverables/workinggroup.aspx?wg=testingtools>

- Elaboration / Construction

- ▶ Verify ws-i compliance at wsdl / schema level
- ▶ Verify use of interoperable schema constructs
- ▶ Verify meta-information (e.g. restrictions) in schema is accurate
- ▶ If industry schema is used, test for compatibility with SOAP stacks





## Function: Service Implementation

### ■ Elaboration / Construction

- ▶ Test for "unhappy" path - null, empty string, empty arrays, fault, uncaught exceptions
  - *Client can send anything across the wire, including XML data that are non-compliant with the WSDL and Schema.*
  
- ▶ Test for boundary conditions (e.g. outside of restriction range)
  - *Most SOAP stacks do not perform schema validations !!!*
    - E.g., mandatory fields are not really mandatory
    - minLength, maxLength is not enforced
  
- ▶ Confirm functionality and interoperability of advanced WS-\* capabilities (e.g. attachments, transaction, reliable messaging)
  - *Don't assume interoperability until it is tested*
  
- ▶ Test your implementation for ESB compatibility



## Function: Service Implementation

- Functional testing hierarchy for Web Services
  1. jUnit Core Functionality Testing
    - Developer runs jUnit Test to validate core functionality, independent of web service infrastructure
  2. Web Service Client Testing
    - Developer or tester uses generated / temporary web service client to access and test web service to validate functionality running on infrastructure
    - Ex: Web Services Explorer, [www.soapclient.com/soaptest.html](http://www.soapclient.com/soaptest.html)
  3. Automated Web Service Client Testing
    - Developer or tester uses test automation tool to automate testing of temporary web service client
    - Ex: Rational Functional Tester automation of [soapclient.com/soaptest.html](http://soapclient.com/soaptest.html)
  4. Automated SOA Testing
    - Tester uses tool specifically designed for SOA applications to validate functionality, performance
    - Best solution for multiple services, with multiple data combinations to validate
    - Ex: IBM Rational Tester for SOA Quality



## Service Performance

### ■ Elaboration

- ▶ Conduct single user round-trip analysis to measure XML payload size and overhead
  - XML Serialization/Deserialization ≠ Slow

### ■ Construction / Transition

- ▶ Perform load simulation test with proper mix of successful and exception scenarios
  - Response time and CPU processing need for SOAP Fault processing and BPEL exception handler may surprise you!



# Security



- Elaboration
  - ▶ Verify Authentication and Authorization mechanism
- Construction
  - ▶ **Vulnerability discovery: E.g. WSDL scanning.**
    - Similar to a thief searching for an open window or unlocked door, revealing internal weaknesses and exposures.
  - ▶ **Probing attacks: E.g. Parameter Tampering and Replay Attacks.**
    - Similar to a thief trying random combinations on locks
  - ▶ **Coercive Parsing: E.g. Recursive Payloads, Oversize Payloads and Denial of Web service Attacks.**
    - Similar to a thief cutting the wires to a core system of a house – the XML parser –in order to gain access.
  - ▶ **External Reference Attack: E.g. External URI Reference.**
    - Similar to letting a stranger into your house who you think is a friend.
  - ▶ **Malicious Content: E.g. Schema Poisoning and SQL Injections.**
    - Similar to a thief delivering a misleading package that results in stolen identities, information leaks and fraudulent transactions.



# Operations: Continuous Service Monitoring

## ■ Construction / Transition

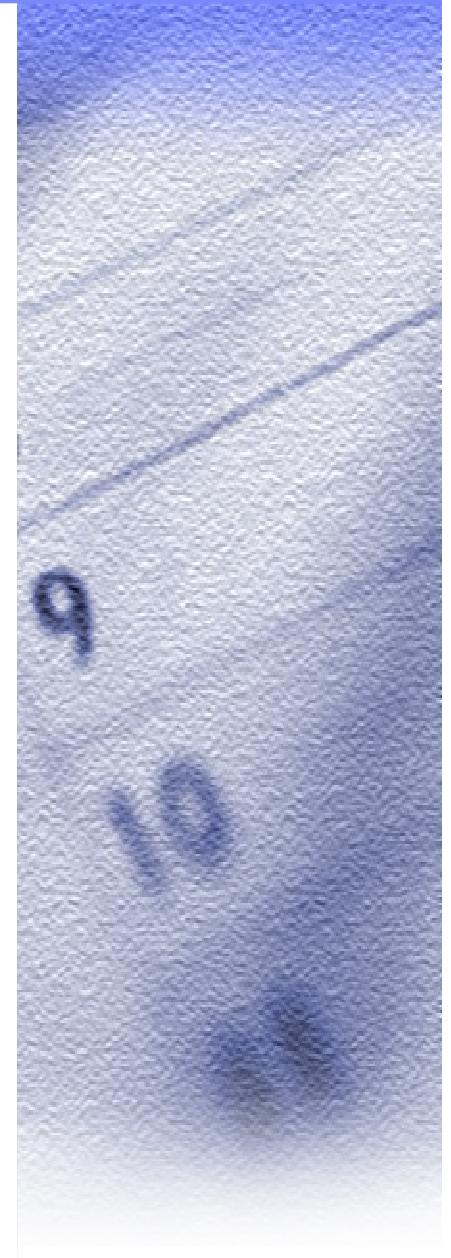
- ▶ Continuous monitoring is the only way to know how your service is being used
- ▶ Test event emission (CBE) and monitoring mechanism
- ▶ Test for proper “housekeeping” of resources, especially under exception situations
  - Proper timeout of stateful services or macro-flows
  - Resources are always returned to the pool (jms, jdbc, jca)





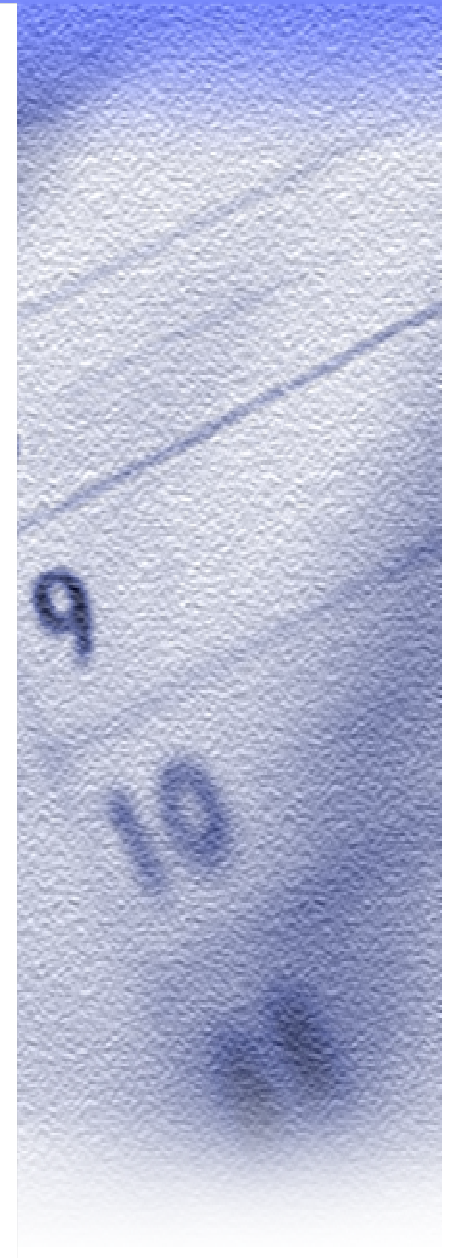
# Agenda

- **SOA Architecture Overview**
- **Demo:**  
**Building and Deploying a Web Service**
- **Testing SOA Applications**
  - ▶ **Challenges**
  - ▶ **Strategies**
- **Demo:**  
**Testing a Web Service**
- **Summary Lessons Learned in the Field**

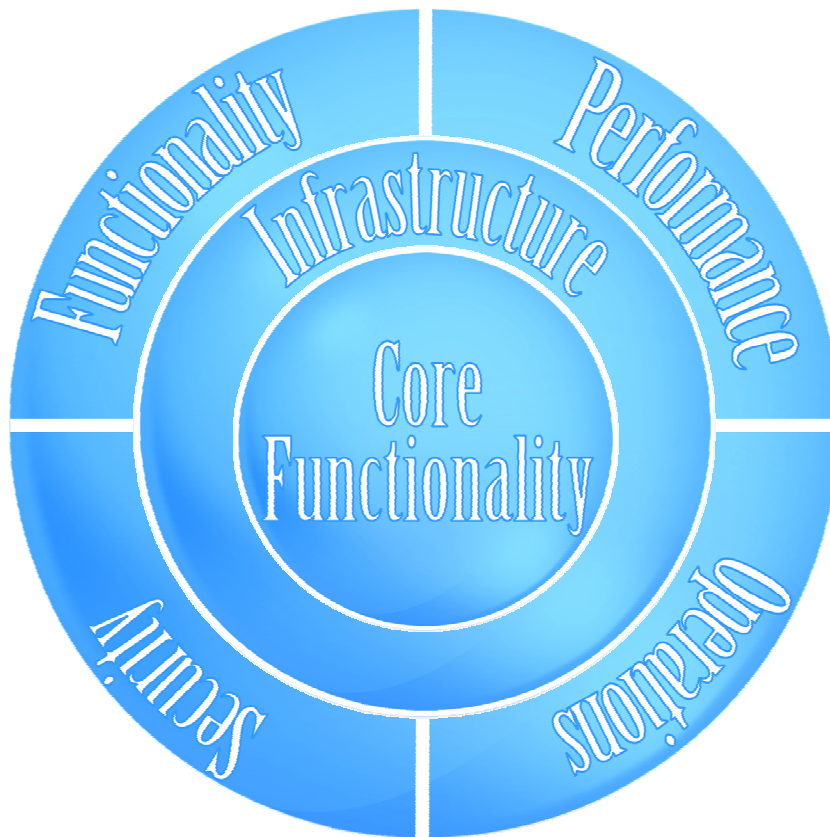


# Agenda

- **SOA Architecture Overview**
- **Demo:**  
**Building and Deploying a Web Service**
- **Testing SOA Applications**
  - ▶ **Challenges**
  - ▶ **Strategies**
- **Demo:**  
**Testing a Web Service**
- **Summary Lessons Learned in the Field**



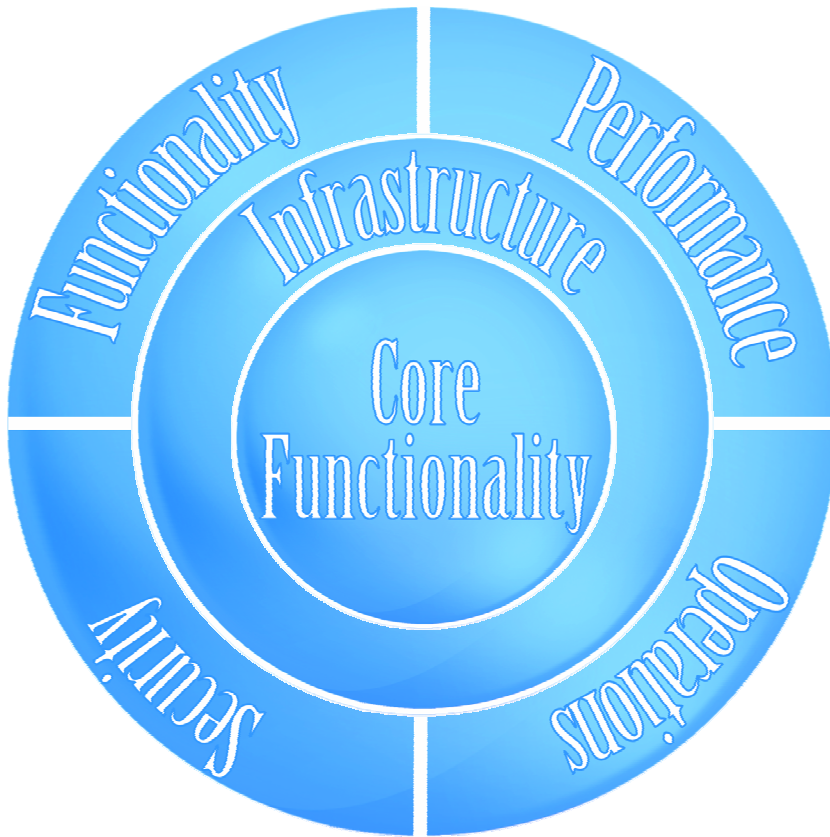
# Testing Web Services: QA Checklist



## ■ To Do:

- ▶ Ensure the functionality of the core service itself
- ▶ Validate the operability / interoperability of the web service infrastructure
- ▶ Ensure performance on service and infrastructure
- ▶ Continuously monitor deployed services for new trends that impact the way a service is being used
- ▶ Focus on data, data, data
- ▶ Focus on bad data, bad data, bad data for security purposes

# Testing Web Services: QA Checklist

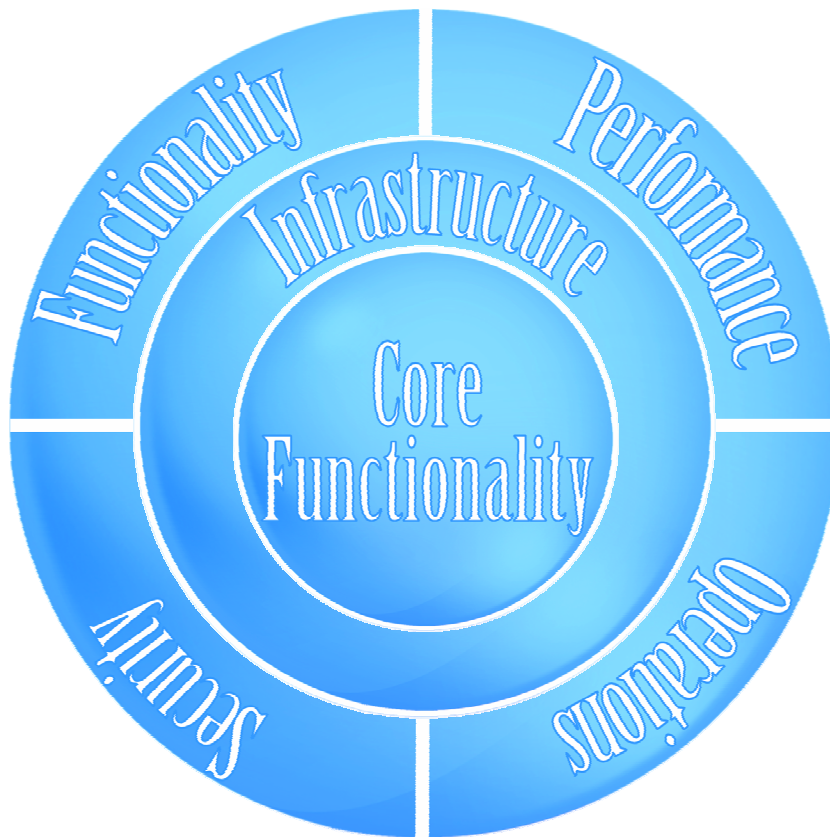


- **What's New Here**

- ▶ API/Headless testing paradigm
- ▶ Web service infrastructure
- ▶ XML structure will be new for many
- ▶ Emphasis on security and performance
- ▶ Additional tooling required



# Testing Web Services: QA Checklist



- **What's Not**

- ▶ **Focus on data**

- However heavy emphasis on data might be considered new – especially in security case

- ▶ **Focus on use cases**

- However more than ever will you have to battle ambiguity as often use cases aren't known





## Top 5 Lessons Learned from the field

1. Learn enough XML to be able to read & understand a WSDL
2. A well-annotated, expressive WSDL contract helps testing significantly
3. Know your schema dependencies. Changes to shared schemas often have cascading effects to testers. Know what to retest.
4. Conduct early interoperability testing with known platforms
5. Automate as much as possible. Leverage developer jUnit tests where available.





IBM Software Group

# Testing SOA Applications: A Guide for Functional Testers

*Brian Bryson, IBM*  
*bbryson@ca.ibm.com*

**Rational.** software

[Go to IBM](#)