Assurance in Service-Oriented Environments

Soumya Simanta

Research, Technology, and System Solutions (RTSS) Program
Software Engineering Institute
Carnegie Mellon University
Pittsburgh 15232

28th October, 2009

Assurance in a Service-oriented World

Assurance of service-oriented systems is similar to assurance of any distributed systems

- Existing assurance approaches still apply at a component level

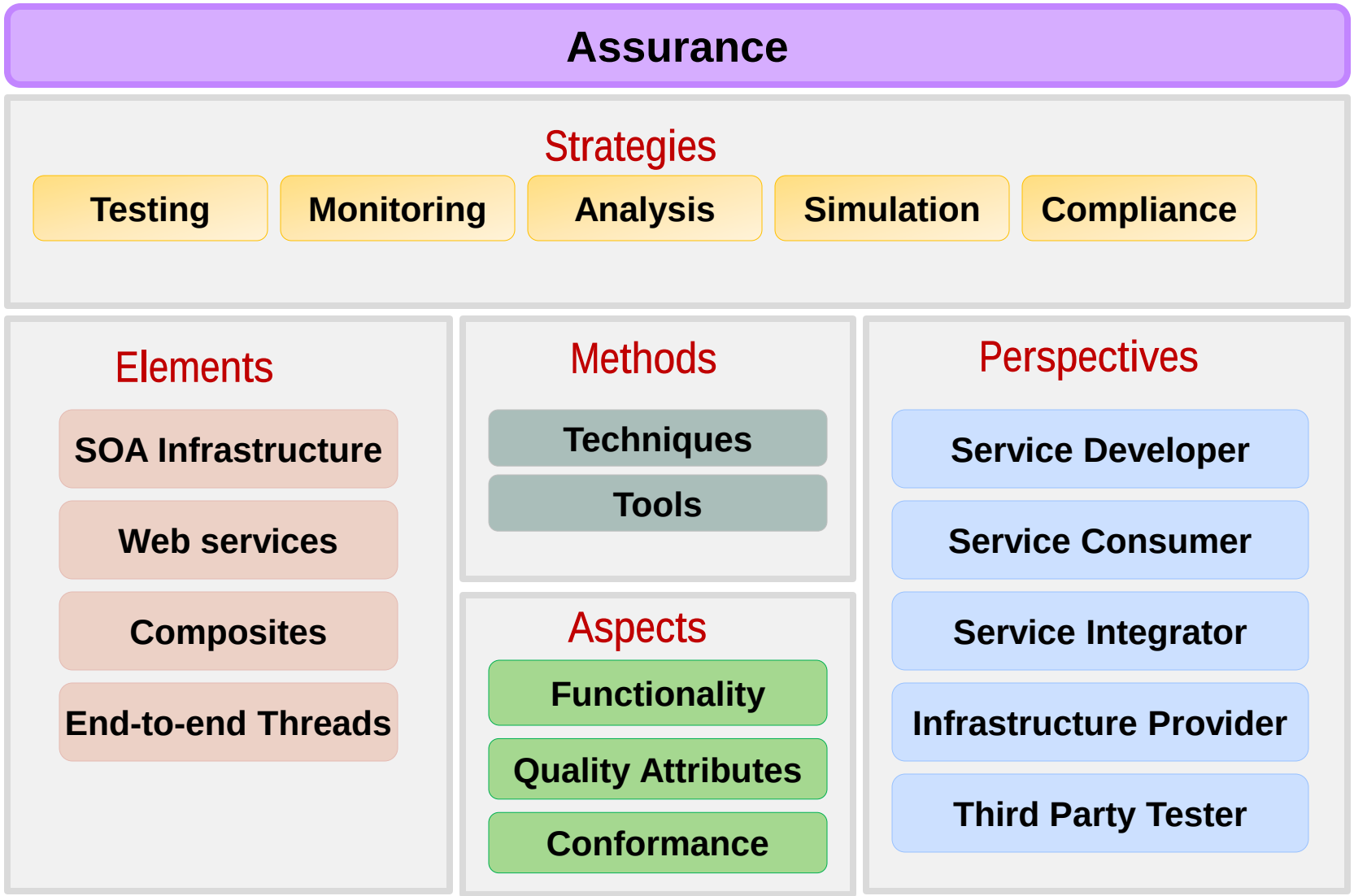
However, service-oriented environments bring new challenges because of their unique characteristics

- Reduced control
- Reduced observability and visibility
- Reduced trust
- Increased coordination and collaboration

Therefore, assuring SOA-based systems requires

- A new mindset
- Additional assurance methods, techniques, and tools
- Successful collaboration and coordination between participants

Dimensions of Assurance in Service-Oriented Systems



Assurance Strategies$_1$

Testing

- Popular and effective strategy
- Not exhaustive but often provides good confidence
- Both automated and manual
- Works on the actual implementation

Monitoring

- Essential for providing runtime assurance for dynamic nature of SOA environments
- Complementary to other strategies
- Automated with manual intervention
- Works on the actual implementations

Analysis

- Limited applicability and scalability. Techniques such as model checking and static analysis are not always applicable
- When applicable can provide high assurance and confidence
- Almost always automated
- Works mostly on abstractions (models)

Assurance Strategies$_2$

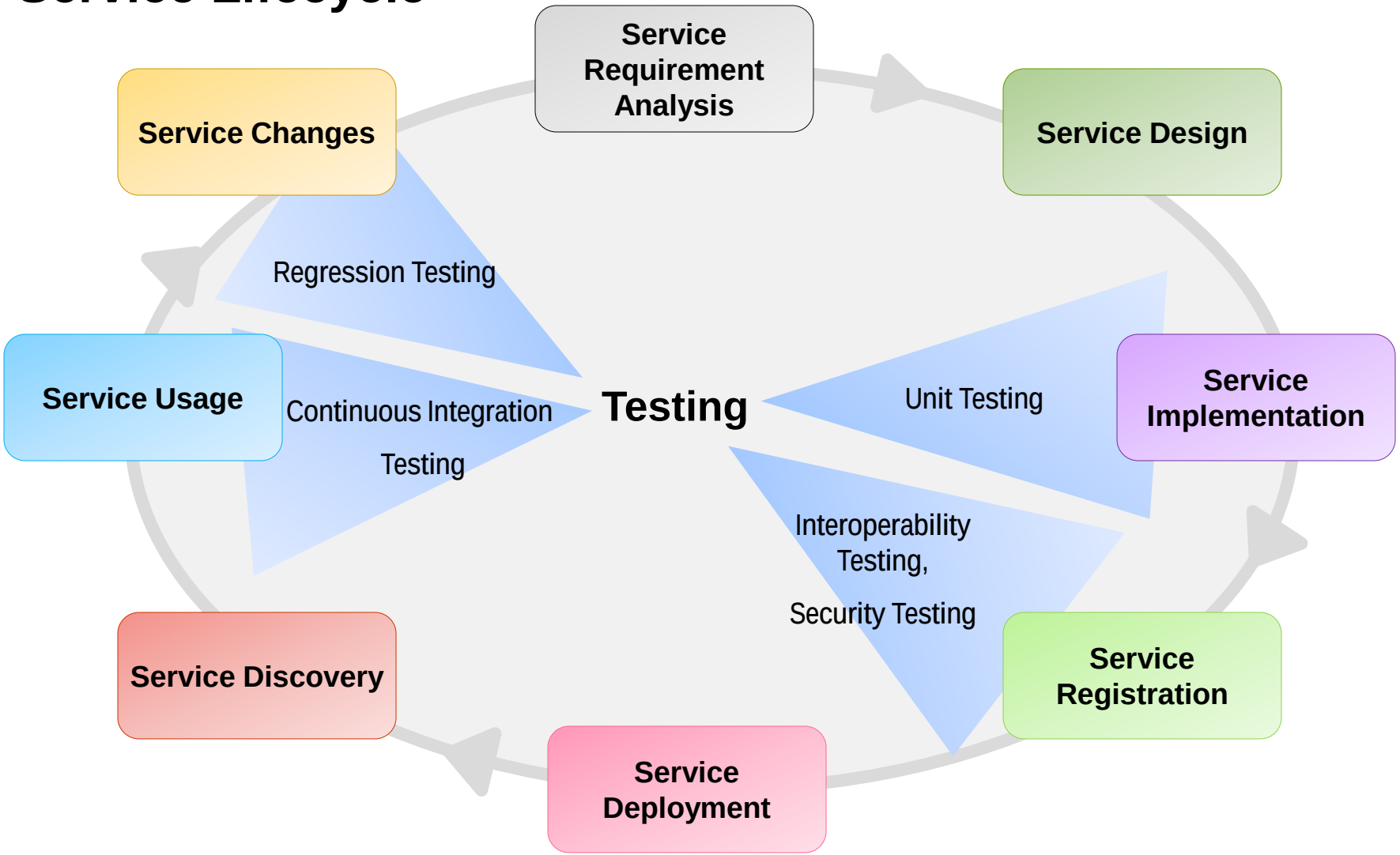
Simulation

- **Limited applicability and use**
- **Can be useful to obtain insights when all elements are not ready**
- **Works when actual implementations are really complex and/or expensive to test**
- **Often requires substantial modeling and results are only as good as the models**

Compliance

- **Third party assurance**
- **Cannot be applied to all properties**
- **Difficult to achieve because of distributed, loosely coupled, and dynamic nature of SOA**
- **Weaker than certification**

Example: Assurance Strategy (Testing) Applied to Service Lifecycle



Assurance Strategies: Summary

Strategies are not mutually exclusive

- *Conformance* often requires a combination of **testing**, **simulation** and **analysis**
- *Simulation* can be used to perform **unit testing** of web services

Strategies have strengths and weaknesses. Therefore, a combination of strategies is required to get better assurance.

Not all strategies are applicable to all contexts and perspectives.

SOA Elements that Require Assurance

SOA Infrastructure

- Consists of business independent capabilities
- Capabilities common across multiple services
- Example: service discovery, managing metadata, provide security, and message delivery

Web Services

- Provide business-level capabilities
- The elements of a web service from a testing perspective are the service interface, service implementation, message format, message payload, and service level agreement (SLA).

End-to-end Threads/ Business Processes

- End-to-end threads or business processes are composites of humans, applications, services, back-end applications, and databases that utilize the SOA and network infrastructure to perform a mission or business task.
- End-to-end threads include services along with other interacting components (human, functional, infrastructural), along with the operating environment.

SOA Infrastructure: Assurance Challenges

Limited internal and technical information

- Testers may not have access to source code and design information necessary for testing all components used in a SOA infrastructure

Complex configurations

- Infrastructure often consists of complex components that are configured for a particular infrastructure

Rapid release cycles

- Commercial components are upgraded and patched frequently, requiring a rapid assurance cycle

Cross infrastructure variation

- Different product implementations and versions across multiple SOA infrastructures may result in variations, making assurance difficult

Web Services: Assurance Challenges$_1$

Unknown contexts and environments

- From a service developer's perspective it is not easy to anticipate and unit test for all usage scenarios.
- Even when the service developer is aware of all contexts, it may be difficult and expensive to create a test environment that addresses each context.

Lack of source and binary code

- The source code and binary code of the web service are unavailable to the service integrator and service tester.
- White box testing and analysis techniques such as static analysis are impossible.
- This challenge is problematic for organizations that maintain high information assurance standards.

Web Services: Assurance Challenges$_2$

Unanticipated demand

- Specific usage of the service (e.g., load, network and infrastructure delay, data) is unknown at development time, making it difficult to test and verify QoS expectations.

Standards conformance

- Web services should conform to standards, if they are to provide syntactic interoperability. Testers have to ensure that web services comply and conform with standards, if they have to provide syntactic interoperability.

End-to-end Threads: Assurance Challenges$_1$

Decentralized ownership and lack of centralized control

- Distributed ownership that makes it difficult to set up a test environment
- Data and process contexts that are often outside the control of the end-to-end tester
- Services that participate in an end-to-end thread are loosely coupled

Complexity

- End-to-end threads that require interaction of services are black boxes and can be recursive
- Service implementations

Long-running business activities

- End-to-end threads are often long running business processes. This increases the testing time if all conditions have to be tested.
- It may not be possible to indentify all conditions that need to be tested.

End-to-end Threads: Assurance Challenges$_2$

Cascading failures

- It can be difficult to identify the cause of failure, assign blame, and mandate an appropriate patch when many nodes in the pathway are outside the control of the organization performing end-to-end testing.

Regression testing

- Changes at any service, node, or component along the pathway exercised in support of an end-to-end thread may indicate a need for regression testing of the thread. Maintaining awareness of these changes requires agreements regarding what types of changes require notification, when such changes are allowed to occur, and how affected parties are notified.

Dynamism

- End-to-end threads are not static. Changes in one node may affect many threads. In some cases these changes may be unknown and therefore difficult to test.

Perspectives$_1$

Service Developer

- **Creates the interface of an individual service and its underlying implementation by using an existing component and wrapping it as a service or creating the service implementation "from scratch"**

Service Provider

- **Provides services. A service provider may or may not be the developer of the service; however, a service developer can also be a service provider.**

Service Integrator (Service Consumer)

- **Uses existing services (individual or composite) to either create composite services or to create an end-user application**

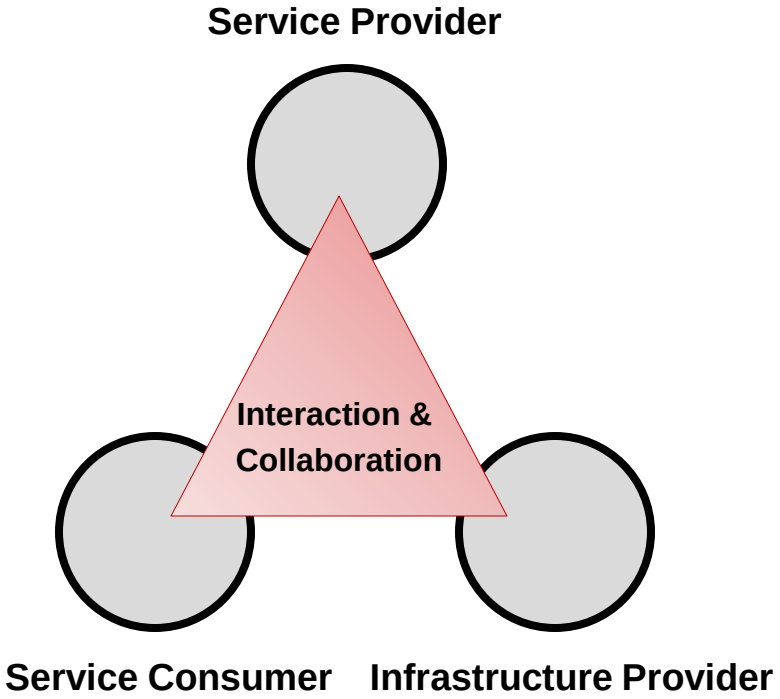
Perspectives$_2$

Infrastructure Provider	• Provides the necessary SOA infrastructure middleware (e.g., enterprise service bus [ESB]) and infrastructural mechanisms such as service discovery to service providers, service consumers, and service integrators
Third-party Service Tester or Certifier (Service Consumer)	• Validates and potentially certifies whether a service (individual or composite) works as expected
End User (Service Consumer)	• Uses applications that directly or indirectly use services • Participates in beta-testing and report errors and faults

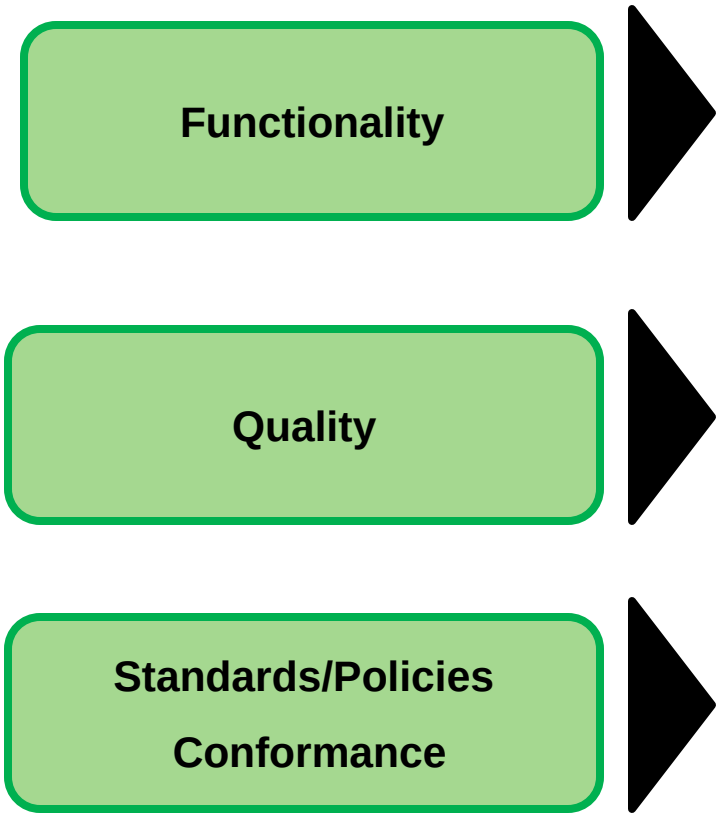
Collaborative Assurance

Successful assurance in an SOA-environment requires collaboration between various perspectives

- An SOA requires participation by service provider, service consumer, and infrastructure provider
- Some assurance strategies are more collaborative than others—such as group testing or collaborative verification and validation (CV&V)



What Aspects for Service-oriented Elements Require Assurance



- **The functional capability provided by the SOA element – SOA infrastructure (e.g., service registration), service (e.g., get customer data), and end-to-end threads/business process (e.g., order book)**
- **The quality of the functionality provided by the SOA elements. Example qualities are performance, security, reliability etc.**
- **The standards set the SOA elements must conform to. These standards can be open or proprietary.**

Assurance Challenges - Interoperability

WS standards (e.g., WS-*) are currently limited to enabling syntactic interoperability

Assuring syntactic interoperability is difficult

- Customized vendor implementations of standards
- Large number of evolving standards
- Only some standards are meant to be interoperable

Assuring interoperability at higher levels (semantic and organizational) is more challenging

- Difficult to standardize due to large number of agreements required
- Specific needs associated with business processes and data models
- Testers have to understand the semantics and the business processes to verify interoperability at these levels

Assurance Challenges - Security

Services are autonomous and black boxes to the service consumers

- White box testing and static analysis techniques cannot be used by service consumers
- Sandboxing techniques to isolate a service implementation cannot be used

Services may be provided on a untrusting network by service providers

- An application developer must establish trust across a large number of distributed nodes having varying degrees of trust

Service composition is recursive

- A service invoked by the application may invoke other services with their own set of distributed nodes, any of which could be untrustworthy

Unknown and dynamic attack surface

- New services are added and old services are retired
- New service consumers are added
- Late binding of services

Assurance Challenges - Reliability

Service implementations are hidden from service consumers

- Difficult to debug if the implementation has crashed or just failed because of specific input(s)
- The developer often finds it difficult to design tests to stress boundary conditions of the service through fault injection or other techniques

Long-running business processes and transactions

- Testing reliability of long-running processes is difficult

Difficult to implement a central coordinator

- Hard to detect root causes of faults and recover from them when services are autonomous
- Not easy to implement transactional services

Assurance Challenges - Performance$_1$

Evaluation, verification, and monitoring of performance properties such as the latency and throughput of web services is similar to that of other distributed computing technologies

- Unavailability of source or binary code makes the empirical verification of performance even more challenging
- It becomes almost impossible to trace execution paths unless owners of all participating elements agree to collaborate

Difficult to pinpoint performance bottlenecks

- Services are black boxes to service consumers
- No single authority controls all the services in a composition
- Composite web services are more difficult to analyze for performance because the elements of a composite service are not known until runtime

Runtime monitoring of these services needs to ensure that services do not cross the acceptable limits set for them and applications can react when these are crossed

Assurance Challenges - Performance$_2$

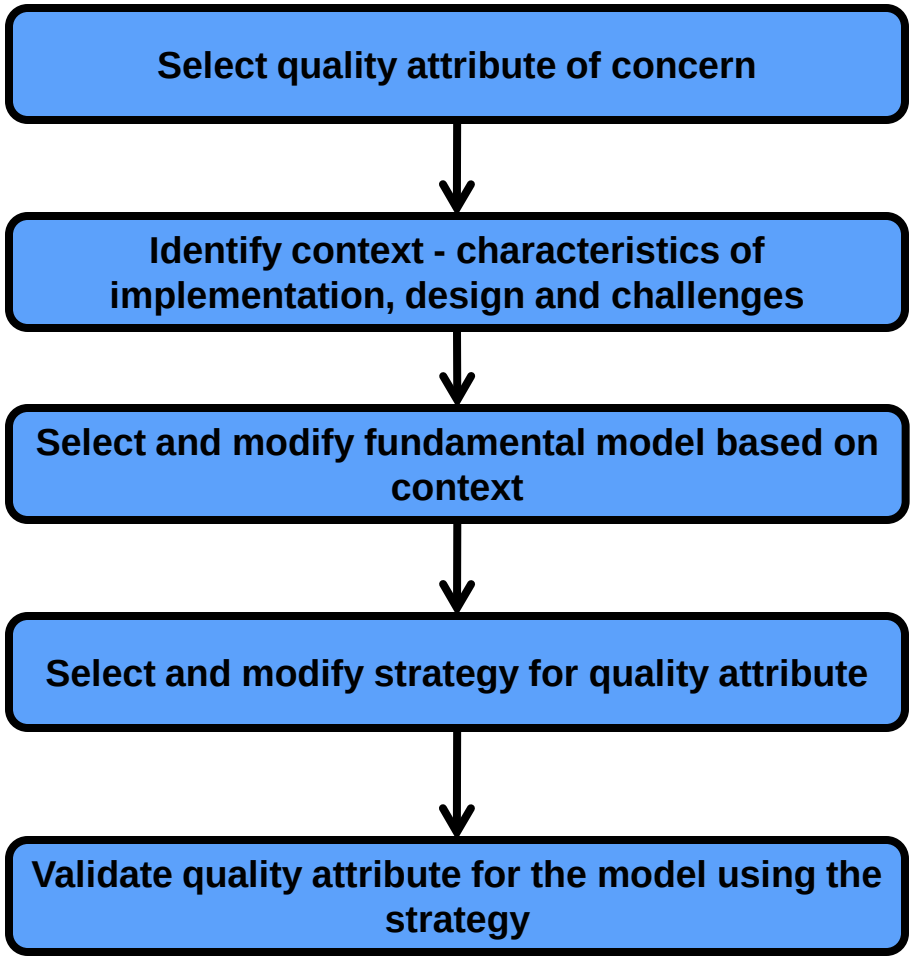
Web services are vulnerable to changing loads

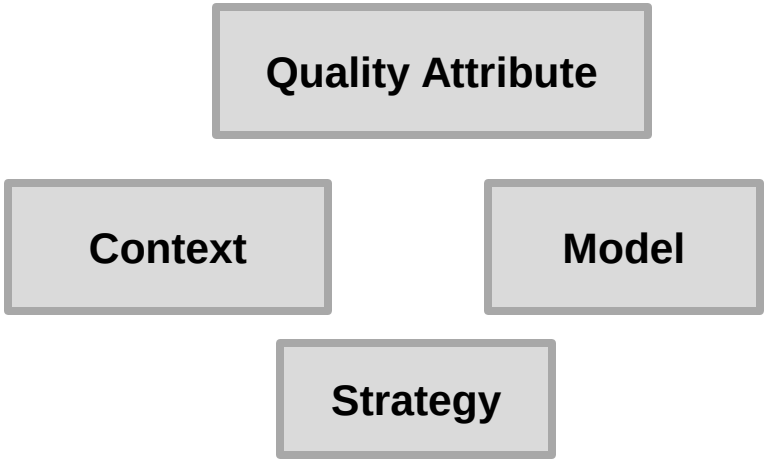
- The effect of unpredictable loads can be much more drastic in the case of shared web services and composite web services
- If a web service has multiple consumers, the increase in load from one service consumer can degrade performance of all other consumers, unless the services are specifically designed for the load
- It is difficult to identify the specific source of a bottleneck, where source code is not available

In the case of composite web services, it may not be clear if they are under-performing due to

- Heavy load on the service
- Heavy load on the network
- A performance bottleneck in another service invoked by the composite service

Assuring Quality Attributes





Assuring Quality Attributes

Quality Attribute	Base Model Elements	Strategies
Interoperability	Levels of Interoperability Set of Standards	Compliance – For WS-* based web services check conformance to WS-I profiles Testing – Identify critical interactions (business processes and mission threads) and perform integration testing Compliance and Monitoring – Test services for conformance to standards at the time of time of registration and publishing
Reliability	Fault Model	Fault injection – check the response of the service to different types of faults Empirical evidence using test execution Monitoring and adaptive correction

Assuring Quality Attributes

Quality Attribute	Base Model Elements	Strategies
Security	Threat Model	Testing - Fault injection, bad input generation at unit testing level
	Attack Surface	Testing - Penetration testing
		Analysis - Static analysis
		Compliance - Conformance to security standards
		Monitoring - Monitoring for malicious code and attacks
Performance	Latency	Testing – Continuous load testing
	Throughput	Monitoring – Monitoring for increased load
		Testing – Testing for impact of other quality attributes (e.g., security) on performance

Evaluating Tool Capabilities$_1$

Usability

- Is the tool easy to use? How steep is the learning curve?
- Example: Intuitive GUI

Protocol and standard support

- What standards and protocols does the testing tool support?
- Example: Support for both REST and SOAP-based web services

Interoperability

- Is the testing tool interoperable with other tools?
- Example: Can test cases from one tool be reused with another tool?

Automation

- What level and type of automation does the tool provide?
- Example: Capture and replay; generating test cases from models

Evaluating Tool Capabilities$_2$

Monitoring
- Does the tool provide service monitoring capabilities?
- Example: Monitoring a service for increased load or denial of service attacks

Simulation
- What kind of simulation support does the tool provide?
- Example: Mocking services clients and instances

Static analysis
- Does the tool provide static analysis support for checking service code?
- Example: Taint analysis of a service for checking security

QoS Testing
- What qualities of services testing can be performed by the tool?
- Example: Load testing, security analysis

Formal models
- Does the tool support formal models for exhaustive checking?
- Example: Model checking

SOA Testing Tools

SoapUI

TestMaker

WebInject

SOAPSonar

Qengine

iTKO LISA

SOAPscope

SOAtest

SOA Test Governance

Should be part of the overall SOA governance strategy

Shared governance framework between service providers, service consumers, and infrastructure providers

- Service-orientation often hides information that may be relevant or sometimes necessary for assurance
- Shared governance mechanisms can allow sharing this information resulting in better assurance

Assurance Decisions

Decision makers should consider

- How much should be invested in quality assurance?
- When should assurance be performed? (i.e., how do assurance activities integrate with the phase of a service life cycle?)
- Who should participate in assurance activities?
- What are the risks and cost associated with not doing proper assurance?
- What polices and governance should be place for collaborative assurance?
- What tools, frameworks, and mechanisms should be used for assurance?
- What types of assurance strategies are appropriate for the context?

Takeaways and Recommendations

Characteristics of service-oriented systems (distributed, loosely coupled) pose serious challenges from an assurance perspective.

Testability should be an important concern from the start when engineering a service-oriented system. Introducing assurance later in the cycle can be expensive.

Often a combination of complementary strategies will be required to achieve acceptable assurance in service-oriented environments.

Existing assurance approaches are still valid below the service level.

As a service-oriented systems becomes more distributed and loosely coupled, it becomes harder to provide assurance because of decreasing control.

As service-oriented becomes widely accepted, more focus has been given to assurance issues. The service-oriented assurance field is evolving, with many open issues that are still under research.

Contact Information

Soumya Simanta

ssimanta@sei.cmu.edu

Research, Technology, and System Solutions (RTSS)

Software Engineering Institute

Carnegie Mellon University

4500 Fifth Avenue

Pittsburgh, PA 15213

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at .

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.