



Evaluating a Service-Oriented Architecture

Paulo Merson
Software Engineering Institute

SBCARS, Campinas, 29-August-2007


 Software Engineering Institute | CarnegieMellon

© 2007 Carnegie Mellon University



Goal of this tutorial:

- Offer *practical information to help the evaluation of the architecture of an SOA system*

 Software Engineering Institute | CarnegieMellon

Evaluating SOA
Paulo Merson
© 2007 Carnegie Mellon University

2

Agenda

Introduction

Architecture Representation, Stakeholders, QA Requirements

SOA Architectural Approaches

SOA Design Questions

Closing



Software Engineering Institute

CarnegieMellon

Evaluating SOA
Paulo Merson
© 2007 Carnegie Mellon University

3

What is Service Oriented Architecture?

SOA is an architectural style where systems consist of service users and service providers



A service is a self-contained, distributed component with a published interface that stresses interoperability, is discoverable and dynamically bound.



Software Engineering Institute

CarnegieMellon

Evaluating SOA
Paulo Merson
© 2007 Carnegie Mellon University

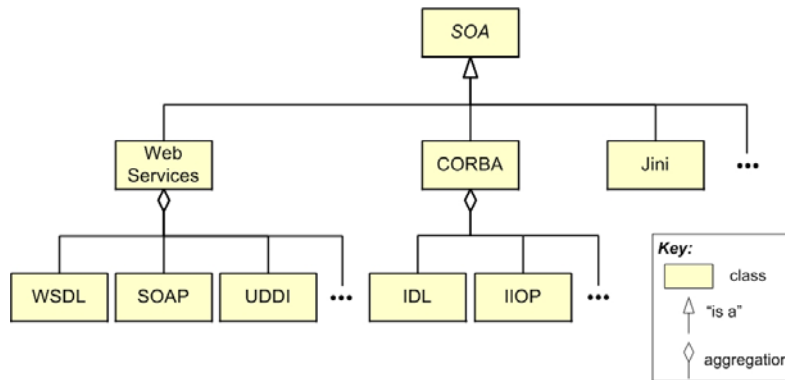
4



SOA and Web Services

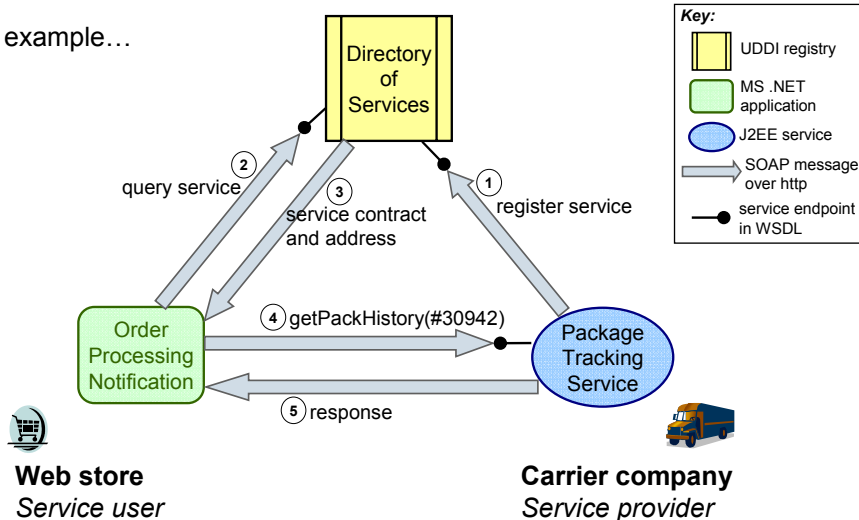
SOA is an architectural style

Web Services is a technology used to implement SOA



How Does It Work?

An example...



Drivers for SOA

Better reuse of IT assets

Corporate mergers and acquisitions

Changing business partnerships (e.g., a new supplier)

Modernization of obsolete systems for financial, functional or technical reasons

Integration with legacy systems



Software Engineering Institute

CarnegieMellon

Evaluating SOA
Paulo Merson
© 2007 Carnegie Mellon University

7

Agenda

Introduction

[Architecture Representation, Stakeholders, QA Requirements](#)

SOA Architectural Approaches

SOA Design Questions

Closing



Software Engineering Institute

CarnegieMellon

Evaluating SOA
Paulo Merson
© 2007 Carnegie Mellon University

8



Stakeholders (1)

Stakeholders are people with a vested interest in a system

A diverse group of stakeholders decreases the risk of overlooking important concerns in the architecture

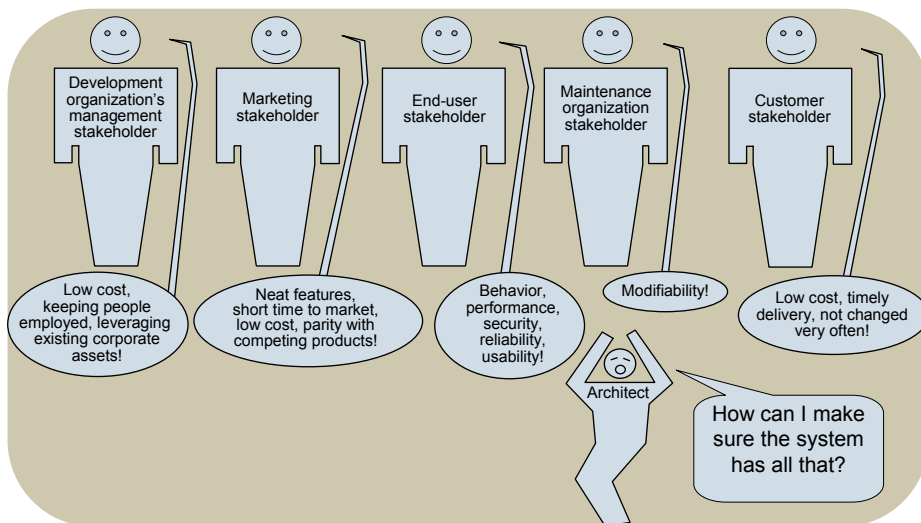
- Architects, developers, testers, integrators and project managers
- Sponsors, end users, business analysts, security officers
- System administrators, DBA, operators, network administrators
- And others

Don't forget:







- *service users if the system is a service provider*
- *service providers if the system is a service user*



Stakeholders (2)



Quality Attributes in SOAs

-  Interoperability
-  Performance
-  Security
-  Availability and reliability
-  Modifiability
-  Testability

Quality attribute requirements (non-functional requirements) can be specified as quality attribute scenarios



SOA General Quality Attribute Scenarios (1)

Interoperability:

- A new business partner that uses platform 'X' is able to implement a service user module that works with our available services in platform 'Y' in two person-days

Performance:

- The service provider can process up to 'X' simultaneous requests during normal operation keeping the response time on the server less than 'Y' seconds

Modifiability:

- The interface of a service that is publicly available changes; the old version is maintained for 12 months and existing service users are not affected in that period



SOA General Quality Attribute Scenarios (2)

Security:

- An unauthorized service user attempts to invoke a protected service provided by the system; the system rejects the attempt and notifies the system administrator

Availability:

- Unscheduled server maintenance is required on server 'X'; the system remains operational in degraded mode during maintenance

Reliability:

- A external service provider is unavailable; modules that use that service respond appropriately regarding the unavailability of the external service and the system continues to operate without failures



Agenda

Introduction

Architecture Representation, Stakeholders, QA Requirements

[SOA Architectural Approaches](#)

SOA Design Questions

Closing



SOA Architectural Approaches

Communication Approaches

Integration Approaches

BPEL

Static vs. Dynamic Web Services



SOA Communication Approaches

How's the communication between service user and provider?

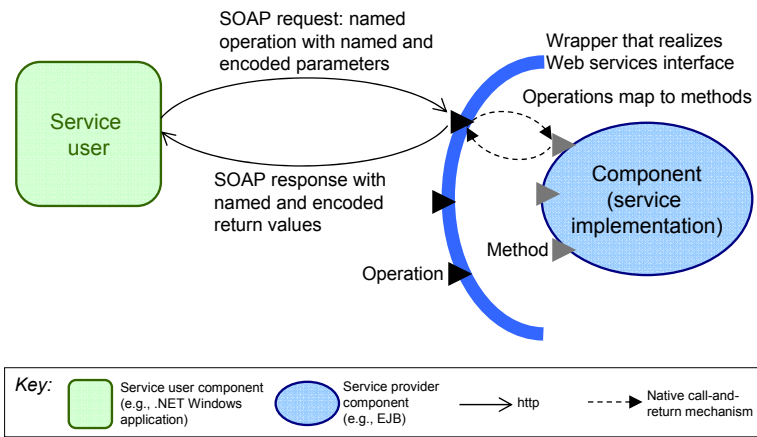
Main alternatives:

- Web Services (SOAP)
- REST
- Messaging systems

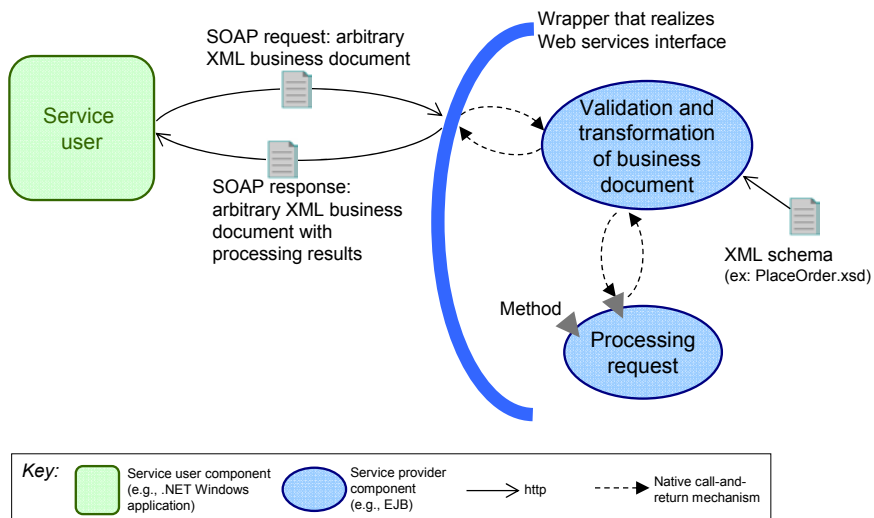
The SOA environment may involve a mix of these along with legacy protocols



Web Services – RPC-Encoded SOAP



Web Services – Document-Literal SOAP



Document-Literal vs. RPC-Encoded

	RPC-Encoded	Document Literal
Interoperability	☹ Less interoperable due to SOAP encoding	☺ Recommended by WS-I
Performance	☹ Processing overhead to encode payloads	☺ No encoding overhead
	☹ Requires DOM parsing	☺ Allows other parsing technologies
Modifiability	☺ Service interfaces closer to programming language	☹ Harder to implement and debug XML schemas, processing and transformation code
	☹ Clients more susceptible to interface changes	☺ More flexibility in changing definition of business documents



Representational State Transfer – REST

Resource. Examples:

- Current weather for zip code 15219
- Temperature averages for city Pittsburgh in May

Resource URI. Examples:

- <http://www.weather.com/current/zip/15219>
- <http://www.weather.com/avg/city/Pittsburgh?month=5>

For each resource, there is a *representation*

- Format is usually XML. Example:

```
<w:Weather>
  <time>Current</time>
  <locationType>zip code</locationType>
  <locationValue>15219</locationValue>
  <temperature unit="Celsius">12</temperature>
</w:Weather>
```

Operations on resources	
Create	http post
Retrieve	http get
Update	http put
Delete	http delete



REST Compared to SOAP-Based Web Services

REST is better:

- Interoperability – requires only http support
- Easier to learn
- Modifiability – only the data contract has to be understood, the interface contract is uniform
- Performance – no intermediaries or marshalling required

SOAP-Based Web Services is better:

- Tool support
- Support for security, reliable messaging and transaction management
- “Network knowledge” and skill base due to widespread adoption

You can always provide both SOAP and REST interfaces for your services



Software Engineering Institute | CarnegieMellon

Evaluating SOA
Paulo Merson
© 2007 Carnegie Mellon University

23

Messaging Systems

Based on IBM WebSphere MQ, Microsoft MSMQ, Oracle AQ, SonicMQ and similar products

Offer asynchronous message exchanges (point-to-point or pub-sub)

Benefits:

- Reliability
- Loose coupling
- Scalability

Challenges:

- Asynchronous model is more complex
- Interoperability – proprietary messaging systems require bridges to interact

WS-ReliableMessaging or WS-Reliability will help with the interoperability problem



Software Engineering Institute | CarnegieMellon

Evaluating SOA
Paulo Merson
© 2007 Carnegie Mellon University

24



Integration Approach

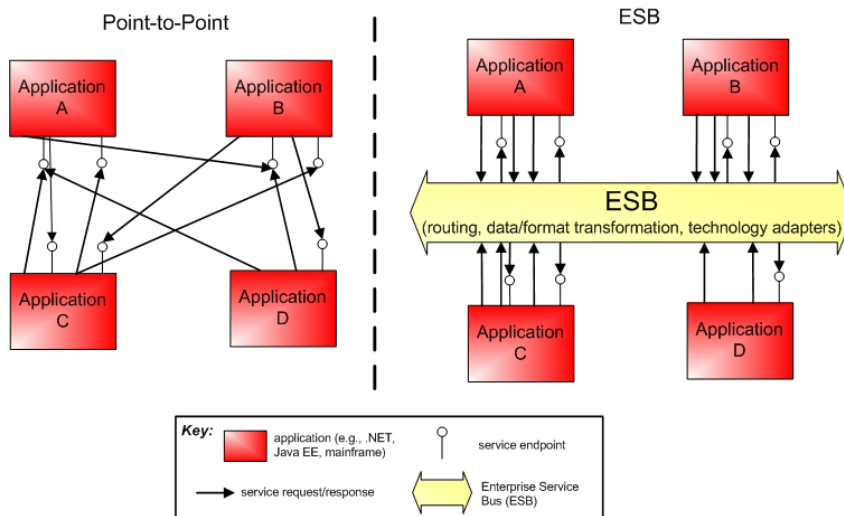
There are multiple possible integration approaches

Commonly divided into:

- Direct point-to-point
- Hub-and-spoke or ESB (Enterprise Service Bus)



Direct Point-to-Point vs. ESB



Direct Point-to-Point

Services are directly connected between applications

Each application must adapt to each connected service:

- Service location
- Interface technology (e.g., asynchronous/synchronous, SOAP vs. REST)
- Security (authentication, authorization and privacy)
- Versioning



ESB

Services connect to a common backbone using Web services, other standards or adapters

Capabilities:

- Web services support
- Connectivity to different technologies through adapters
- Message routing:
 - Fixed A-to-B or based on content, availability, load-balancing or other rules
 - One-to-many or many-to-one (message aggregation)
- Data transformation (format, content based on business rule, splitting and merging)
- Advanced tooling (e.g., graphical routing definitions, runtime monitoring)



Point-to-Point and ESB Tradeoffs

	Point-to-Point	ESB
Modifiability	☹ Changes to a service interface induces change to all connected applications	☺ Service interfaces may change and compatibility is managed in the ESB in many cases
Performance	☺ No transformation and routing overhead	☹ Transformation and routing overhead
Security	☹ Authentication and authorization managed case-by-case by each service	☺ Allows independent management of security for each service
Interoperability	☹ No special support for interoperability between service users and providers	☺ Supports integration of diverse platforms
Cost	☺ No extra cost	☹ Higher cost and complexity of initial implementation



Direct Point-to-Point or ESB?

Point-to-Point is most acceptable in environments that are:

- Small in number of services and applications
- Homogenous in technology
- Low pace of change (business and technology)

ESB is most acceptable in environments that are:

- Large
- Technically diverse
- Rapidly changing



BPEL

Business Process Execution Language (BPEL) is an XML-based standard language to model business processes

Service users and providers are called “partners” and interact primarily via Web Services and messaging

Includes operations like *invoke*, *receive*, *reply*, *throw*, *wait*, *switch*, *while*

BPEL tools allow non-technical developers to create workflows visually

BPEL code is deployed to a BPEL engine (aka BPEL server)

An event triggers the workflow and the BPEL engine coordinates invocation of services using the BPEL code as a script



BPEL Quality Attributes Considerations

Enhanced modifiability

- BPEL allows externalizing process flow logic from source code
- Changing business process workflows is easier in visual BPEL tools

Interoperability

- BPEL engine allows systems in disparate platforms to interoperate
- But BPEL standard is still emerging and interoperability has limitations

Improved reliability due to better constrained sequencing of service interactions and error handling (compared to custom-developed workflow)

Performance overhead due to BPEL code interpretation

Higher cost and complexity in initial implementation



BPM, BPMN and BPEL

BPM (business process management) is a generic term related to:

- methods, techniques and tools to design, execute and monitor business processes

BPMN (business process modeling notation) is a standard graphical notation for drawing business process workflows

BPEL is an XML-based process modeling language

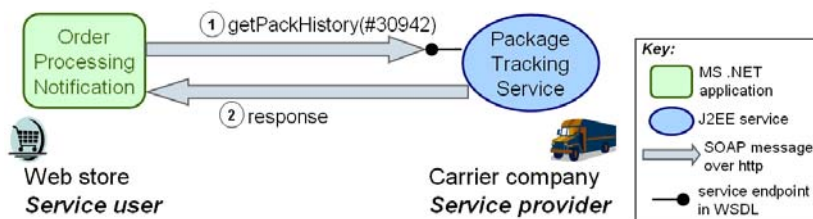
- BPEL has no standard graphical notation
 - Some BPEL tool vendors used their own graphical notations; others adopted BPMN
 - Mapping between BPMN and BPEL is not always straightforward



Static vs. Dynamic Web Services (1)

Static binding of service user and provider

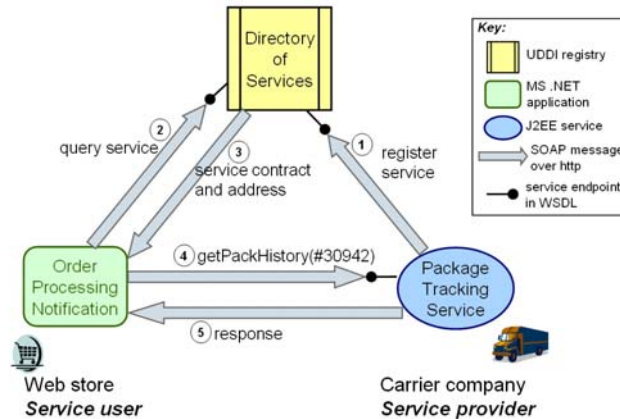
- Service location is known when service user is developed or deployed



Static vs. Dynamic Web Services (2)

Dynamic binding:

- Provider must register the service with a registry



Registry Capabilities

Registration, naming and querying of services

- interface descriptions (WSDL) and XML schemas
- security information
- other metadata

History and versioning

Support for standards (e.g., UDDI v3)



Quality Attribute Considerations

Positive effects of using a registry:

- loose coupling
- support to versioning of services

Negative effects of using a registry

- performance overhead
- increased complexity of service user implementation
- interoperability

Static binding is sufficient for most business scenarios and design solutions



Agenda

Introduction

Architecture Representation, Stakeholders, QA Requirements

SOA Architectural Approaches

[SOA Design Questions](#)

Closing



Design Questions

Can be used to probe the architecture and elicit:

- Strengths
- Weaknesses
- Risks

The list of questions and assertions:

- Is not meant to be exhaustive
- Are common tendencies—there will always be exceptions



Design Questions Covered

Target Platform

Synchronous vs. Asynchronous

Security of Messages

Granularity of Services



What Is Known About the Target Platform?

SOA solutions may involve different platforms

- runtime environment of service providers
- runtime environment of service users
- development environment
- network infrastructure

Many quality concerns are primarily handled or strongly affected by the platform



Target Platform – Evaluation Questions

What standards or WS-I profiles are supported?

What mechanisms are offered to increase:

- availability/reliability
- security (e.g., VPN, declarative authorization)
- throughput (e.g., load-balancing)

What are known issues and technical limitations of the version in use?

What is the support for logging, measuring and monitoring?



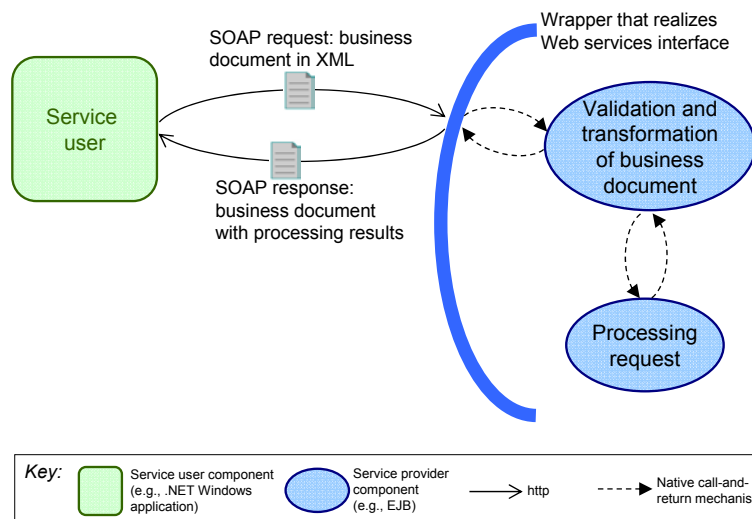
Synchronous or Asynchronous Services?

Choice depends on

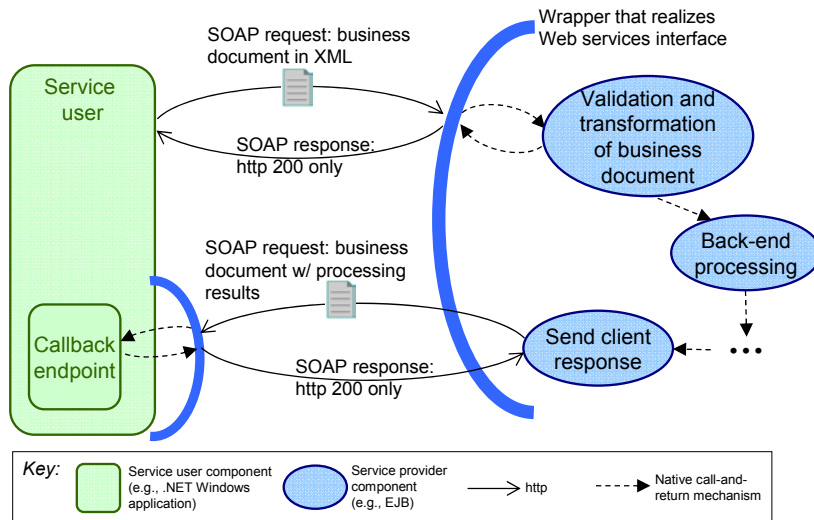
- Business requirements
- QA requirements
- Existing components capabilities



Synchronous SOAP-Based WS



Asynchronous SOAP-Based WS



Synchronous vs. Asynchronous Tradeoffs

	Synchronous Services	Asynchronous Services
Modifiability	☺ Simpler to implement	☹ More complex logic to deal with waiting, callback and correlation
Scalability	☹ Poor for large applications	☺ Best scalability for large applications
Reliability	☹ More susceptible to complex distributed failures	☺ Better independent operation and fault-tolerance
	☺ Simpler error and exception handling designs	☹ More complex error/retry logic



HTTPS or Message-Level Security?

Common threats to distributed systems:

- Disclosure threats
 - XML documents are sent in the clear by default
- Disruption threats
 - Technologies such as SOAP, HTTP and XML are vulnerable to Denial of Service (DoS) attacks



Software Engineering Institute | CarnegieMellon

Evaluating SOA
Paulo Merson
© 2007 Carnegie Mellon University

47

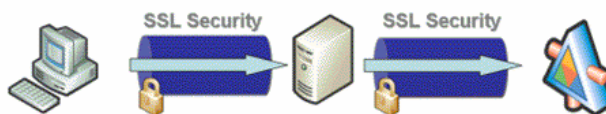
HTTPS

HTTPS is HTTP over SSL

- Entire message encrypted from point-to-point
- Reasonable protection from eavesdroppers and “man-in-the middle” attacks

Problem: message lifecycle may be longer than point-to-point

- Multiple hops
- Intermediaries with different policies and controls
- Messages persisted at various points



Software Engineering Institute | CarnegieMellon

Evaluating SOA
Paulo Merson
© 2007 Carnegie Mellon University

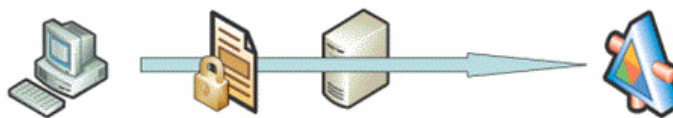
48



Message-Level Security

Service users and providers bind security tokens to messages using the WS-Security standard

- Allows encrypting and signing all or just parts of the message
- Message is protected end-to-end



HTTPS vs. Message-Level Security Tradeoffs

	HTTPS	Message-Level Security
Complexity	☺ Has been around and is well understood	☹ Requires careful management of which parts of a message need to be secured
Interoperability	☺ More interoperable	☹ Emerging standards may not be supported by all parties
Flexibility	☹ Inflexible all or nothing	☺ What parts are encrypted can change; what credentials to use can change
Security	☹ Security is only enforced point-to-point	☺ Security is enforced end-to-end for the entire message lifecycle



Coarse- or Fine-Grained Services? - 1

A fine-grained service interface contains small operations. Example:

- validateCreditCard(CreditCard cc)
- validateCustomer(Customer customer)
- orderItems(List items)

A coarse-grained service has operations that typically comprise a complete transaction. Example:

- placeOrder(Customer customer, CreditCard cc, List items)

Granularity of operations is a critical design factor. Guidelines:

- Operations should be self-contained and comprise a single database transaction
- Limit the number of operations to reduce performance overhead of multiple calls
- Create operations that can be composed as needed today and in a foreseeable future



Service Granularity Tradeoffs

	Coarse Grained	Fine Grained
Performance	☺ Improved by reducing the number of messages	☹ Requires more message exchanges
Testability	☺ Simplifies testing by limiting the number of possible paths	☹ Testing is more challenging because the order of operations is not controlled
Flexibility	☹ Not as flexible	☺ More flexible in assigning authorization for different operations ☺ Gives clients more control over the steps of an operation ☺ Enables service reuse and composition



Service Granularity – Evaluation Questions

What are the bandwidth limitations on the network?

Do operations in the interface map to transactional boundaries?

Are there ordering dependencies between operations?

How stable are the business processes? Is the service likely to change?



Agenda

Introduction

Architecture Representation, Stakeholders, QA Requirements

SOA Architectural Approaches

SOA Design Questions

Closing



Other Important SOA Design Concerns

- Service versioning
- Legacy systems integration
- Service authentication and authorization
- XML optimization
- Exception handling and fault recovery



Important Takeaways

SOAP is not the only option for SOA communication – REST and messaging systems also work

ESB versus direct point-to-point, pros and cons

Consider BPEL when business process is defined and modifiability is more important than performance

Once you understand the importance of each QA requirement, you can weigh the relevance of each design question

The runtime view of the architecture shows service-orientation

SOAs involve **many** technical design considerations



For More Information

“Evaluating a Service-Oriented Architecture”

by P. Bianco, R. Kotermanski, P. Merson. CMU/SEI-2007-TR-015.

(in final review stage)

“Quality Attributes for Service-Oriented Architectures”

by L. O’Brien, P. Merson, L. Bass. SDSOA Workshop, 2007.



Questions – Now or Later

Paulo Merson – pfm@sei.cmu.edu

- *What’s SaaS?*
- *Can I use BPMN to document the SOA architecture?*
- *What are the typical risks found in an SOA evaluation?*
- *Is ESB a product, something I have to develop, an infrastructure service of my application server, or something else?*

