Evaluating a Service-Oriented Architecture

Paulo Merson, SEI
with
Phil Bianco, SEI
Rick Kotermanski, Summa Technologies

May 2007

Software Engineering Institute | Carnegie Mellon

Goal:

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

Agenda

Introduction

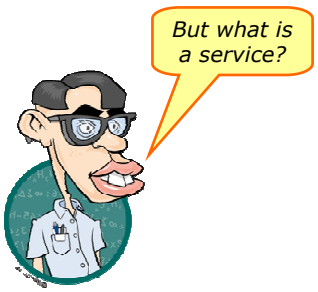
SOA Architectural Approaches

SOA Design Questions

Closing

What is Service Oriented Architecture?

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

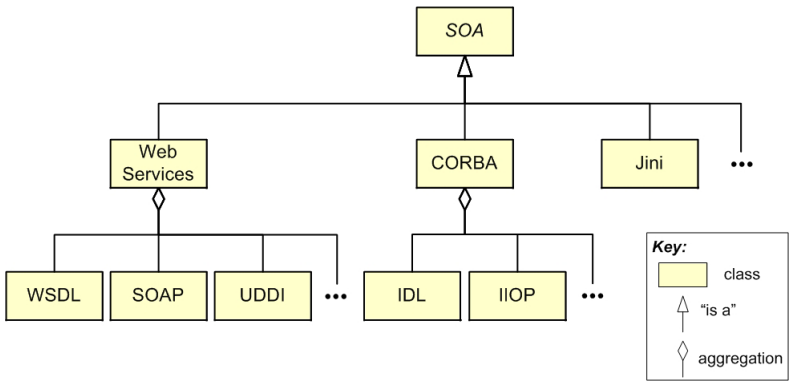


A service is
a self-contained, distributed component with a published interface that stresses interoperability, is discoverable and dynamically bound.

SOA and Web Services

SOA is an architectural style

Web Services is a technology used to implement SOA

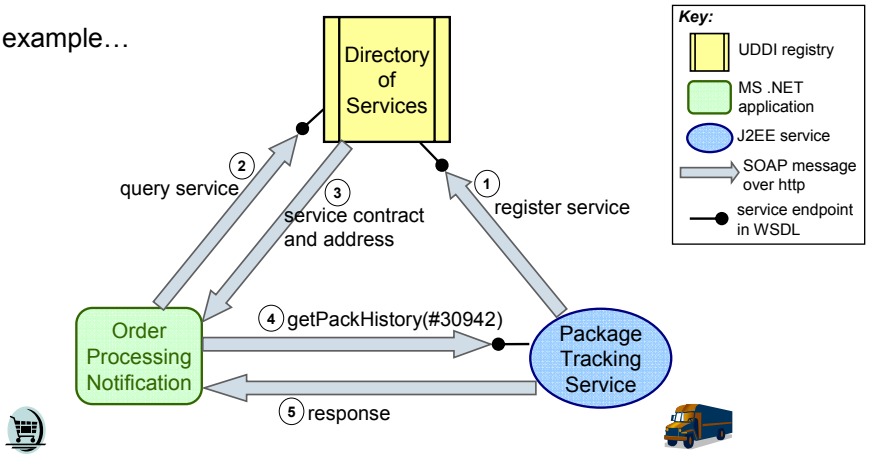


Software Engineering Institute | Carnegie Mellon

Evaluating SOA
Merson, Bianco, Kotermanski 5
© 2007 Carnegie Mellon University

How Does It Work?

An example…



Web store
Service user

Carrier company
Service provider

Software Engineering Institute | Carnegie Mellon

Evaluating SOA
Merson, Bianco, Kotermanski 6
© 2007 Carnegie Mellon University

ATAM

In the analysis, the evaluation team:

- identifies architectural approaches
- asks quality attribute questions about the design decisions
- identifies and records risks and tradeoffs

In SOA systems,
- What architectural approaches could be used?
- What quality attribute questions could the evaluators ask?

Agenda

Introduction

SOA Architectural Approaches

SOA Design Questions

Closing

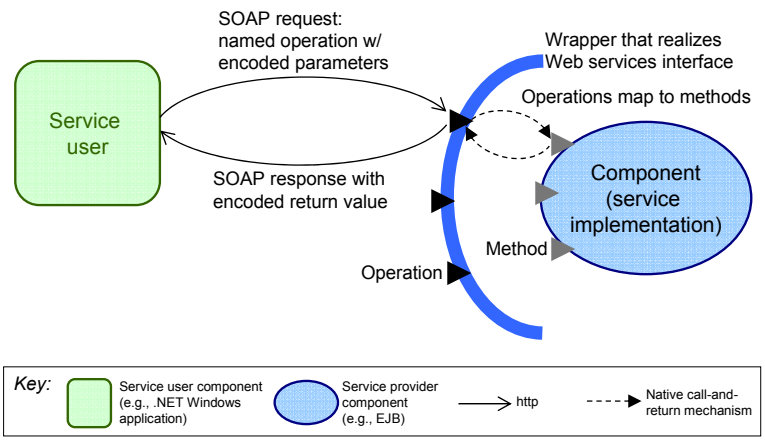
SOA Communication Approaches

How's the communication between service user and provider?

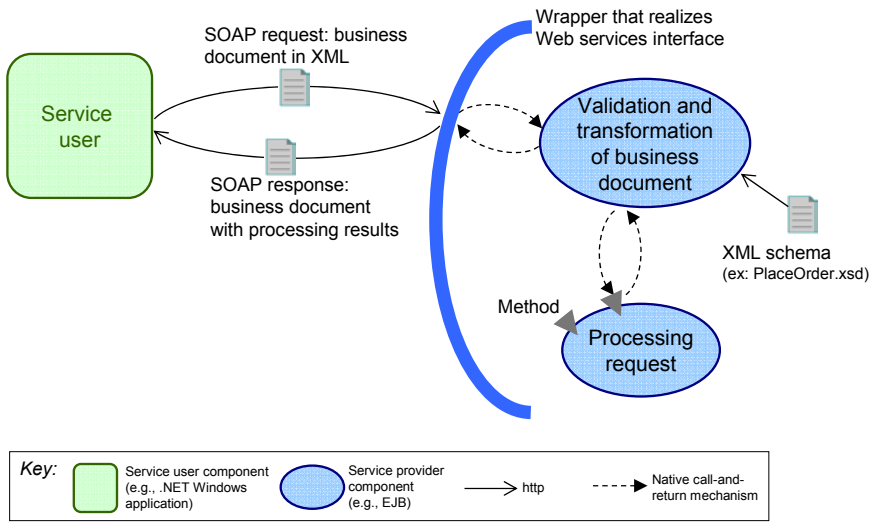
Main alternatives:

- Web Services (SOAP)
- REST
- Messaging systems

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

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

Messaging Systems

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

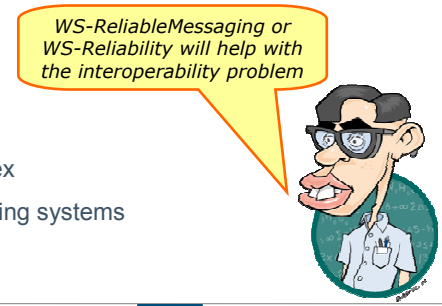
Offer asynchronous message exchanges (point-to-point or pub-sub)

Benefits:

- Reliability
- Loose coupling
- Scalability

Challenges:

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

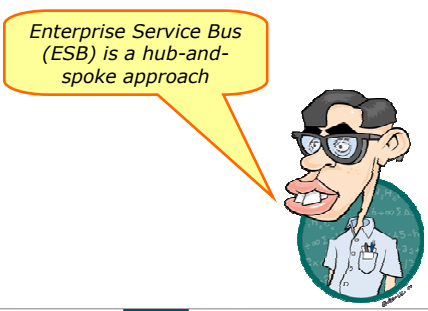


Integration Approach

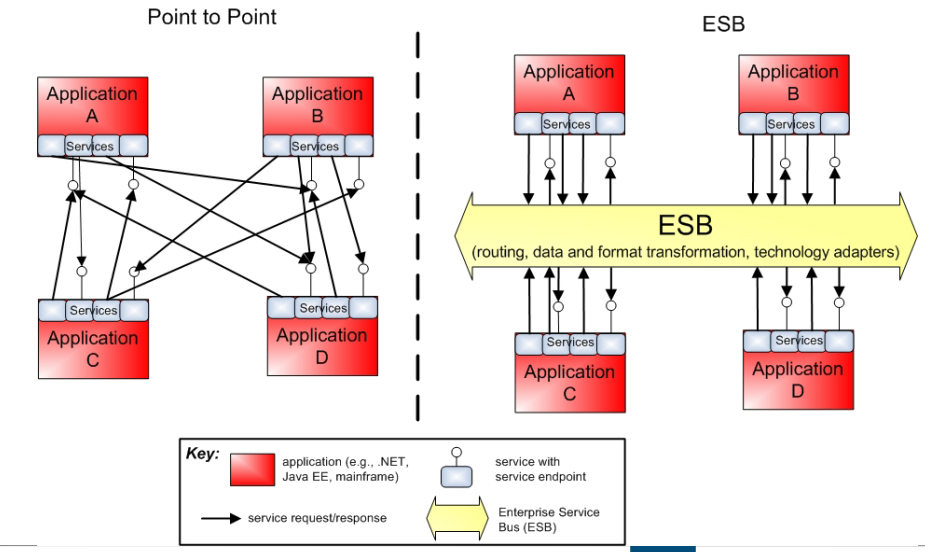
There are multiple possible integration approaches

Commonly divided into:

- Point-to-point
- Hub-and-spoke



Point-to-Point vs. ESB



When to Use 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)

ESBs are most acceptable in environments that are:

- Large
- Technically diverse
- Rapidly changing

Point to Point vs. ESB Tradeoffs - 1

	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

Point to Point vs. ESB Tradeoffs - 2

	Point-to-Point	ESB
Serviceability	☹ Problem determination spread across applications—no central point to manage connectivity	☺ Centralized service management allows centrally log/audit of interactions
Reliability	☹ Strong coupling may result in complex failure modes and unintended dependencies	☹ Additional components add complexity and introduce failure modes ☺ Loosely coupled approach improve overall reliability (ESB may be deployed to avoid SPOF)
Interoperability	☹ Each service to service connection must be compatible	☺ Designed to support diverse connectivity

Agenda

Introduction

SOA Architectural Approaches

SOA Design Questions

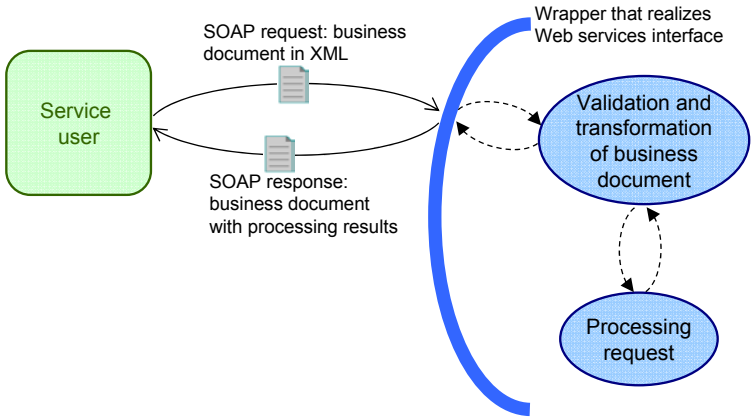
Closing

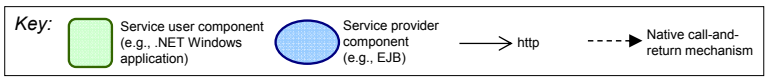
Synchronous or Asynchronous Services?

Choice depends on

- Business requirements
- QA requirements
- Existing components capabilities

Synchronous SOAP-Based WS

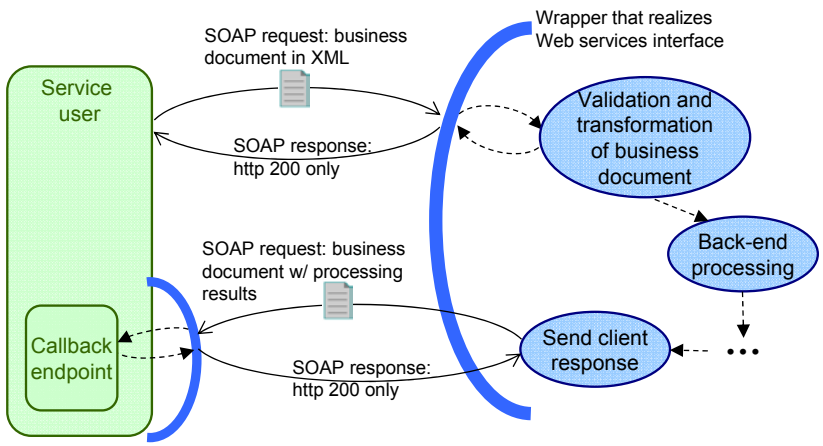


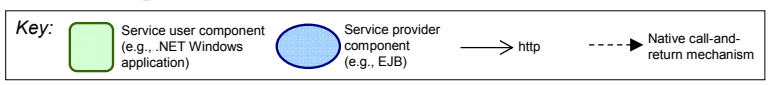


Software Engineering Institute | Carnegie Mellon

Evaluating SOA
Merson, Bianco, Kotermanski 23
© 2007 Carnegie Mellon University

Asynchronous SOAP-Based WS





Software Engineering Institute | Carnegie Mellon

Evaluating SOA
Merson, Bianco, Kotermanski 24
© 2007 Carnegie Mellon University

Synchronous vs. Asynchronous Services - 1

	Synchronous Services	Asynchronous Services
Modifiability	☺ Simpler to implement	☹ More complex logic to deal with waiting, callback and correlation
	☹ Behavior (e.g. timing) dependencies beyond interface syntax make replacement more difficult	☺ Lower coupling (components can be more easily replaced)
	☹ More difficult to insert an ESB because of performance or behavior dependencies	☺ Ease of inserting ESB or other brokering into conversations
	☺ Easier control of serialization of parallel requests	☹ Control of sequencing drives complex correlation, exception management and timeout designs

Synchronous vs. Asynchronous Services - 2

	Synchronous Services	Asynchronous Services
Performance	☺ Designed to achieve better responsiveness	☹ Overhead of messaging
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?

Main difference:

- HTTPs allows point to point security
- Message-level allows end to end security

One doesn't exclude the other

HTTPS

HTTPS is HTTP over SSL

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

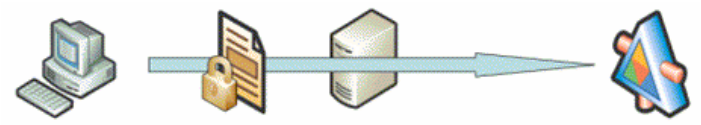
Problem: message lifecycle usually is longer than point to point

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

Message-Level Security

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

- Allows encrypting and signing all or just parts of the message
- Tokens represent claims made by the sender (e.g., authentication, authorization, confidentiality, integrity)
- WS-Security does not address security infrastructure such as key management



HTTPS vs. Message Level Security

	HTTPS	Message Level Security
Performance	☺ Some performance overhead but generally faster response times	☹ Overhead parsing tokens may increase response time
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 from point to point	☺ Security is enforced for the entire message lifecycle.

Coarse- or Fine-Grained Services?

Coarse-grained service typically consists of a complete business process

Fine-grained service usually performs small functions

The following should influence service interface design:

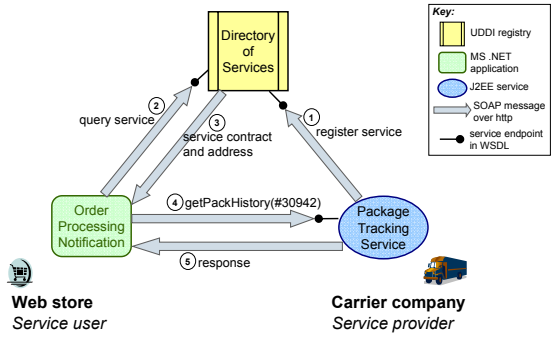
- Transactions and state
- QA requirements

Coarse- vs. Fine-Grained Services

	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 ☺ Give clients more control over the steps of an operation ☺ Enables service reuse and composition

Static or Dynamic Web Services?

Dynamic WS:



Static WS:

Static vs. Dynamic Web Services? - 2

	Static	Dynamic
Performance	☺ Less overhead because service location is known during design ☺ No WSDL processing	☹ Service lookup overhead ☹ Overhead of WSDL processing
Modifiability	☹ Service user and provider more tightly coupled	☺ Dynamic binding enables service provider location to change without affecting service user
Availability	☹ Failover logic has to be in the service user or other intermediary	☺ Directory can route service calls (for failover or load-balancing) ☹ Directory can be a SPOF

Agenda

Introduction

SOA Architectural Approaches

SOA Design Questions

Closing

Important Takeaways

ATAM with no changes can be used to evaluate SOAs

ESB versus point-to-point, pros and cons

SOAP is not the only option for SOA communication – REST and Messaging Systems also work

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

SOAs involve a lot of technical design considerations

Questions – Now or Later

Paulo Merson – pfm@sei.cmu.edu
Phil Bianco – pbianco@sei.cmu.edu
Rick Kotermanski – rek@summa-tech.com

- *What's SaaS?*
- *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?*

Software Engineering Institute | Carnegie Mellon

SUMMA

Evaluating SOA
Merson, Bianco, Kotermanski 37