Service-Oriented Architecture and its Implications for Software Life Cycle Activities

Grace A. Lewis
Software Engineering Institute
Integration of Software-Intensive Systems (ISIS) Initiative
Agenda

SOA: Basic Concepts

Pillars of Service-Oriented Systems Development

Implications of SOA for Software Life Cycle Activities

Conclusions
What is SOA?

Service-oriented architecture is a way of designing, developing, deploying and managing systems, in which

- Services provide reusable business functionality.
- Service consumers are built using functionality from available services.
- Service interface definitions are first-class artifacts.
- An SOA infrastructure enables discovery, composition, and invocation of services.
- Protocols are predominantly, but not exclusively, message-based document exchanges.
Services

Services are reusable components that represent business tasks.

- Customer lookup
- Credit card validation
- Weather
- Hotel reservation

Services can be

- Globally distributed across organizations
- Reconfigured into new business processes

Service interface definitions are well-defined first-class artifacts (ideally) available in a service repository.
SOA Infrastructure

Set of technologies that bind service consumers to services

- Products, standards and protocols that support communication
  - Typically message-based document exchanges
    - Web Services (HTTP, SOAP, WSDL)
    - Message-oriented middleware (i.e. IBM Websphere MQ)
    - Publish/subscribe (i.e. Java Messaging Service — JMS)
    - CORBA …

- Infrastructure services available to service providers and/or service consumers
  - Security, discovery, data transformation, …

- Development, deployment and management tools and guidelines
Service Consumers

Clients for the functionality provided by the services

- End-user applications
- Internal systems
- External systems
- Composite services

Consumers programmatically bind to services.
A service with equivalent functionality can be implemented, and used by all three applications.
The new application can use available services.

New services can be used by other applications as well.
Services and Adaptability

The SOA Infrastructure provides a standard communication mechanism between consumers and services. Changes in services have potentially no impact on existing service consumers.
Services and Legacy Leverage

Legacy platform diversity and complexity is transparent to consumers.

Consumers access the services in a standard way.

It is the service’s task to invoke the legacy system.

SOA Infrastructure

- Customer Lookup Service
- Credit Check Service
- Item Lookup Service
- Inventory Check Service

Order Processing Application

Customer Management System

Manufacturing System
Components of a Service-Oriented System

End User Application  Portal  Internal System  External Consumer

SOA Infrastructure

Service A  Service B  Service C  Service D

Enterprise Information System  Legacy or New Service Code  External System

Internal Users  Security  Development Tools  Discovery

Infrastructure

Service Consumers

Internet

Service Interfaces

Internal System

External System

Discovery

Development Tools

Security

Internet
Three Basic Operations to Support Service-Oriented Systems

Service Discovery

- Services repositories are queried for services with desired characteristics.

Service Composition

- Applications/service consumers are built by integrating functionality provided by services.

Service Invocation

- Services are invoked and service code is executed.
An Example of SOA Implementation: WS* Web Services

WS* Web Services is one mechanism for implementing a service-oriented system.

- Service interfaces are described using Web Services Description Language (WSDL).
- Data is transmitted using SOAP over HTTP.
- UDDI is optionally used as the directory service.

Because it is the most common mechanism, Web Services is often equated to SOA.

Adapted from “XML and Web Services Unleashed”, SAMS Publishing
So What Is Different?

There is nothing conceptually new, but it has brought together existing technologies and good practices in a way that works.

- More aligned with business
  - Services represent coarse-grained business tasks
- Backed by industry
- Standards-based
- Greater degree of rigor in interface specifications
- Truly loosely-coupled
  - No need to install specific components
  - Platform-independent
    - What is behind the interface is invisible to the consumer
So What is the Challenge? Creating Good Services!

Represent reusable tasks

Have (or may have) multiple consumers

Permit consumers to bind to services programmatically

Correspond to functionality of a stateless nature
  • Service has no knowledge of previous visits

Enable service inputs and outputs that do not require the construction of very complex consumers

Are of a request-response nature
  • Although SOA 2.0 supports events
Agenda

SOA: Basic Concepts

Pillars of Service-Oriented Systems Development

Implications of SOA for Software Life Cycle Activities

Conclusions
Pillars of Service-Oriented Systems Development

SOA-Based Systems Development

**Strategic Alignment**
High-level mission and business goals need to dictate the focus of the strategy of SOA adoption and implementation.

**SOA Governance**
SOA governance provides a set of policies, rules, and enforcement mechanisms for developing, using, and evolving SOA assets and for analysis of their business value.

**Technology Evaluation**
Contextual technology evaluation allows early insight into technologies within the context in which they will be used.

**Change of Mindset**
Development is different from traditional systems development—loose coupling between consumers and providers, multiple ownership, shared semantics, etc.
Different Business Needs and Goals Drive Different SOA Strategies

<table>
<thead>
<tr>
<th>Business Needs and Goals</th>
<th>SOA Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase information available to business customers</td>
<td>• Intuitive portals</td>
</tr>
<tr>
<td></td>
<td>• Creation of services related to customer information</td>
</tr>
<tr>
<td>Integrate business partners</td>
<td>• Heterogeneous interoperability</td>
</tr>
<tr>
<td></td>
<td>• Back office integration</td>
</tr>
<tr>
<td></td>
<td>• Identification of business rules</td>
</tr>
<tr>
<td>Improve business processes</td>
<td>• Identification of key processes</td>
</tr>
<tr>
<td></td>
<td>• Elimination of redundancy</td>
</tr>
<tr>
<td></td>
<td>• Consistency between processes</td>
</tr>
<tr>
<td></td>
<td>• Services that access legacy systems</td>
</tr>
</tbody>
</table>
Linkage of Business Processes to Services

1. Business processes to support business goals are identified.

2. Candidate services are identified.
   - Top-Down
     - Shared steps between business processes are identified as service candidates.
   - Bottom-Up
     - Legacy system inventory is performed.
     - Systems with functionality to support business processes are identified as migration candidates.

3. Services are selected based on relationship to business goals.
Disciplined SOA Adoption

Adapted from “Meeting the Challenges of SOA Adoption,” keynote by Roy Schulte at the SOA In Action Virtual Conference, November 2006.
SOA Governance

SOA governance provides a set of policies, rules, and enforcement mechanisms for developing, using, and evolving SOA assets and for analysis of their business value.

It provides the **who**, that **what** and the **how** decisions business, engineering and operations are made in order to support a SOA strategy.

- Policies and procedures
- Roles and responsibilities
- Design-time governance
- Runtime governance
Examples of Governance Elements

Governance elements adapted from a presentation by Dr Mohamad Afshar from Oracle Corporation and Ben Moreland from The Hartford at the Business Transformation Conference 2007.
Match of Technologies to the Problem Domain

Need a realistic understanding on what technologies can do in the specific problem domain.

How to understand and keep up with the “alphabet soup”?  
- XML, SOAP, WSDL, UDDI, WS-Security?

How to determine which standards and technologies to implement in specific situations?

How to build systems that are resilient to changes in standards and commercial products that implement them?

How to determine if selected technologies will meet QoS requirements?  
- Security
- Availability
- Performance

All the above questions suggest a need for contextual experimentation.
T-Check<sup>SM</sup>

Experiment, situated in a specific context, with the goal of providing a “technology sanity check”

The approach

1. Formulate hypotheses about the technology
2. Examine these hypotheses against very specific criteria through experimentation

Extremely efficient

- Focus on implementing the simplest experiment to validate technology claims
Service-Oriented Systems Require a Different Development Approach

<table>
<thead>
<tr>
<th>Traditional Systems Development</th>
<th>Service-Oriented Systems Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tight coupling between system components</td>
<td>Loose coupling between service consumers and services</td>
</tr>
<tr>
<td>Semantics shared explicitly at design time</td>
<td>Semantics shared without much communication between developers of consumers and services</td>
</tr>
<tr>
<td>— In the future, even at runtime</td>
<td></td>
</tr>
<tr>
<td>Known set of users and usage patterns</td>
<td>Potentially unknown set of users and usage patterns</td>
</tr>
<tr>
<td>System components owned by the same organization</td>
<td>Systems components potentially owned by multiple organizations</td>
</tr>
</tbody>
</table>
Agenda

SOA: Basic Concepts
Pillars of Service-Oriented Systems Development

Implications of SOA for Software Life Cycle Activities

Conclusions
General Process Implications

Processes must reflect the required strategic approach

Processes must be iterative
  • Short iterations to respond to business needs

SOA governance must be embedded across the full life cycle
  • Emphasis on problematic areas
  • Automation where possible

Processes must be targeted
  • Processes targeted at service provider types: internal services vs. external services
  • Processes targeted at system component provider: service, consumer or infrastructure

Technology evaluation must become an integral process component
Some Implications for Requirements Activities

Require an business process management (BPM) focus

Must deal with a larger number of stakeholders

First step is to look at the inventory of business processes and services

- Negotiation and adaptation to increase reuse
- May cause refactoring of services
- A high quality registry makes the process easier

In the case of service providers, these need to work with potential requirements

- In the same way COTS products vendors work
Some Implications for Architecture and Design Activities

The responsibilities of each system component need to be clearly defined—consumers, services and infrastructure

- Security, transaction management, data transformations, etc.

Constant technology evaluation

Evaluation of expected quality of service (QoS)

- Tradeoff analysis
- Contextual experimentation
- Implications of external consumers and services

Decisions must promote reuse
Some Implications for Development Activities

Development environments need to be similar/same as production environments—as in any distributed system environment

- In some cases, the simulation of the production environment might be necessary

The emergent characteristics of many SOA technologies cause instability in development activities

Require the establishment of processes for the implementation of service interfaces and infrastructure components

- Traditional processes apply to service implementation
Some Implications for Testing Activities

System testing of a service consumer requires all services (or test instances of them) to be available

- From a service consumer perspective, the service is a black box

Requires greater and more diverse exception handling

- For example, what happens if the service is not available?

Regression tests have to evaluate against all consumer requirements and service-level agreements (SLAs)
Some Implications for Maintenance Activities

Impact analysis affects a larger number of users

- Internal and external users
- Service users and users of the systems that implement the services

Configuration management becomes more complicated

- What artifacts are placed under configuration management? Consumers? Services? Infrastructure?

Greater coordination of release cycles (if possible)

- Between services and consumers
- Between infrastructure and services/consumers
Less Control

Requires giving up full control—not easy

  • Tradeoff is agility

Anticipate objections and understand validity

  • Security
  • Performance
  • Control

Greatest challenges come from

  • Single organization may not own the complete system
  • Services used in unknown ways by (potentially) unknown users

Education and training on new mindset is needed.
Agenda

SOA: Basic Concepts

Pillars of Service-Oriented Systems Development

Implications of SOA for Software Life Cycle Activities

Conclusions
SOA is an approach to software development where

- Services provide reusable functionality with well-defined interfaces.
- An SOA infrastructure enables discovery, composition, and invocation of services.
- Service consumers are built using functionality from available services.

SOA is real

- Many successful case studies, mainly in commercial enterprises
- Main goal for adoption of SOA is internal integration and business process improvement
- Main adoption barriers are lack of governance and finding people with the right skills
Conclusions

SOA is not just a buzzword

- Currently the best option available for loosely coupled systems integration and leverage of legacy systems
- The technologies to implement SOA will change over time, but the concepts are here to stay
  - SOA is much broader than its most popular instantiation (Web Services)

Development of service-oriented systems is about more than just technologies and standards

- It is a way of developing systems
- It requires a change of mindset
- It requires alignment with business goals and processes to be of value
### 2008-2009 Course Dates for *Migrating Legacy Components to SOA Environments*

<table>
<thead>
<tr>
<th>Year</th>
<th>Date</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>October 15-16</td>
<td>SEI DC</td>
</tr>
<tr>
<td>2009</td>
<td>January 28-29</td>
<td>SEI Pittsburgh</td>
</tr>
<tr>
<td></td>
<td>April 16-17</td>
<td>SEI Pittsburgh</td>
</tr>
<tr>
<td></td>
<td>July 29-30</td>
<td>SEI DC</td>
</tr>
<tr>
<td></td>
<td>September 10-11</td>
<td>SEI Europe</td>
</tr>
<tr>
<td></td>
<td>October 28-29</td>
<td>SEI Pittsburgh</td>
</tr>
</tbody>
</table>
References

SMART: The Service Migration and Reuse Technique

- SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment:
  http://www.sei.cmu.edu/publications/documents/08.reports/08tn008.html

T-Checks

- Process: http://www.sei.cmu.edu/publications/documents/05.reports/05tn025.html
- Applications:
  - Web Services and Security: Single Sign-On:
    http://www.sei.cmu.edu/publications/documents/08.reports/08tn026.html
  - Open Grid Services Architecture:
  - Web Services:
    http://www.sei.cmu.edu/publications/documents/06.reports/06tn021.html
  - OWL-S (OWL Web Ontology Language for Services):
    http://www.sei.cmu.edu/publications/documents/06.reports/06tn018.html
  - MDA (Model-Driven Architecture):
    http://www.sei.cmu.edu/publications/documents/05.reports/05tn022.html