A Research Agenda for Service-Oriented Architecture
Challenges of Maintenance and Evolution of Service-Oriented Systems

Grace A. Lewis
Software Engineering Institute, USA
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 permission@sei.cmu.edu.

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.
Approach

Assembled an international research group to analyze the current state of the practice and current research initiatives in SOA

Proposed a long-term consensus research agenda

Performed an extensive literature review and looked at case studies of successful SOA adoption

Created a service-oriented systems development life cycle that supports the strategic approach to SOA adoption shown in case studies

Identified areas of SOA research necessary to fill in the gaps

Validated and evolved finding through multiple workshops and interviews
Successful SOA adoption has a strong link between business strategy and SOA strategy.
Expanded View of the SOA Problem and Solution Space

SOA strategy is the way in which SOA is going to address the organization's business drivers for SOA adoption.

Feedback loops reflect the dynamic and iterative nature of service-oriented environments.

SOA plans are executed to produce or evolve a service-oriented system.

Measurements are gathered to test the effectiveness of the strategy and the system itself.
# Mapping Between Phases, Activities and Indicators

<table>
<thead>
<tr>
<th>PHASES</th>
<th>P1: Strategic Analysis</th>
<th>P2: Planning</th>
<th>P3: Construction</th>
<th>P4: Transition</th>
<th>P5: Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVITIES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1: Business Context Understanding</td>
<td>+++</td>
<td></td>
<td></td>
<td></td>
<td>++</td>
</tr>
<tr>
<td>A2: Business Objectives Specification</td>
<td>+++</td>
<td>++</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3: Risk Analysis and Initial Requirements Gathering</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>A4: Prototyping and Requirements Tuning</td>
<td>++</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5: Implementation</td>
<td>+</td>
<td>+++</td>
<td>++</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A6: Integration and Testing</td>
<td>++</td>
<td>+++</td>
<td>++</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A7: Deployment</td>
<td>+++</td>
<td>++</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A8: Maintenance</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A9: Management</td>
<td>++</td>
<td>+++</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDICATORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I1: Financial Indicator Measurements</td>
<td>++</td>
<td>+++</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2: Technology Indicator Measurements</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>I3: User Rating Measurements</td>
<td>++</td>
<td>+++</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I4: Compliance Indicators</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>+++</td>
</tr>
</tbody>
</table>
Relationship between Solution Space and Research Topics

The development of a service-oriented system requires business, engineering and operations to be made, as well as other cross-cutting decisions.

Our proposed taxonomy of research topics is divided into these decision areas.

The research topics correspond to areas where new/more/different research is needed to support a strategic approach to service-oriented systems development.
Taxonomy of Research Issues

SOA Research Taxonomy

Business
- SOA Strategy Selection
- Business Case for Service Orientation
- Mapping between Business Processes and Services
- Organizational Structures to Support Service-Oriented Environments
- Business Indicators

Engineering
- Process and Life Cycle Requirements
- Service Selection
- Service Definition and Categorization
- Technology Assessments
- Architecture and Design
- Code
- Tools and Products
- Quality Assurance and Testing
- Deployment
- Maintenance and Evolution
- Engineering Indicators

Operations
- Adoption
- Monitoring
- Support
- Service-Level Agreements in SOA Environments
- Operations Indicators

Cross-Cutting
- Governance
- Security
- Training and Education
- Risk Management in SOA Projects
- Social and Legal Issues
Sample of Engineering Research Topics

- Service-Oriented System Life Cycle Models
- Development Processes and Methodologies for Service-Oriented Systems
- System or End-to-End Testing
- Infrastructure Testing
- Simulation and "What-If" Analysis Testing
- Practices for Service Providers to Support System Testing of their Consumers
- Establishing Test Beds and Benchmarks
- Service Certification

- Process and Life Cycle
  - Requirements
  - Service Selection
  - Service Definition and Categorization
  - Technology Assessments
  - Architecture and Design
  - Code
  - Tools and Products
  - Quality Assurance and Testing
  - Deployment
  - Maintenance and Evolution
  - Engineering Indicators

- Service Modeling Languages
- Modeling Dynamic Runtime Architecture of Service-Oriented Systems
- Features and Properties for Next-Generation Frameworks for Development of Service-Oriented Systems
- Architectural Styles for Service-Oriented Systems
- Communication and Connectors
- Architectures for Service Types
- Design for Personalization, Context Awareness and Adaptation
- Design for Service Composition
- Design for Runtime Semantic-Based Discovery and Composition
- Design for Recovery-Oriented Computing
- SOA and Product Lines
- Design for Mobility
- Real-Time SOA
System or End-to-End Testing — Rationale

In an SOA environment, system testing means end-to-end testing.

Problem is that in SOA environments, systems components are distributed, deployed on heterogeneous platforms, and often not even available.
System or End-to-End Testing — Current Efforts

Market for testing tools for SOA environments (mainly for Web Services) is growing.

- Testing at multiple levels—from business processes to messages
- Testing for qualities such as availability, performance, and security

Most testing tools are incapable of building composite interdependent tests across technology platforms, languages and systems.

Most testing tools assume control over all elements of the service-oriented system.

- Sometimes client developers typically only have access to interfaces

Some research into use of gray-box testing, which is appropriate when there is limited knowledge.
System or End-to-End Testing — Challenges and Gaps

Challenges driven by the distributed, heterogeneous nature of service-oriented systems components and a growing market of third-party services which means that there is not a single owner of the complete system.

- Dynamic testing in distributed, heterogeneous environments
- Service certification
  - What does a certification process look like? What can be certified?
- Enhanced service repositories that provide test cases for services
  - How are test cases specified?
- Test-aware interfaces for service consumers to test services
  - How are these test services specified? How do service consumers become aware of their existence?
System or End-to-End Testing — Challenges and Gaps

May be a need to recognize that it is not always possible to do end-to-end testing

- Simulation of service-oriented system environments
- Best practices for exception handling
Sample of Business Research Topics

- Techniques for Service Identification
- Techniques and Processes to Support the Strategic Reuse of Legacy Components and Services
- Analytic Methods for Service Evaluation against Business Needs
- Techniques and Processes for Establishing Relations between Business and Service Models
- Processes to Support the Adaptation of Services to Meet Changes in Business Processes
- Process Refactoring and Reengineering and its Impact on Services

Business

- SOA Strategy Selection
- Business Case for Service Orientation
- Mapping between Business Processes and Services
- Organizational Structures to Support Service-Oriented Environments
- Business Indicators

- Techniques to Establish and Document the Business Case for SOA Adoption
- Cost Models for SOA
- Funding Models for SOA
- SOA Business Value Framework

- Models for Organizational Structures that Enable Service-Oriented Systems Development
- Skills Required to Develop, Use and Maintain Service-Oriented Systems
- Models for Workforce Allocation in Service-Oriented Systems Projects
- Organizational and Funding Models for Shared Services
Sample of Operations Research Topics

- Monitoring of Business Processes in an SOA Environment
- Operations Monitoring for Auditing Purposes
- Self-Healing Service-Oriented Systems
- Resource Allocation and Configuration Management in SOA Environments
- Verification and Validation of Policies at Runtime

Operations

- Adoption
- Monitoring
- Support
- Service-Level Agreements in SOA Environments
- Operations Indicators

- Service Usability
- End-User Service Composition Tools
- Models of Service Consumer Adoption
- Pricing Models for Service Providers

- Processes for Support of Service-Oriented Systems
- Front-end and Back-end Problem Management in Service-Oriented Environments
Sample of Cross-Cutting Research Topics

- Identity Management in Multi-Organizational SOA Environments
- Secure Dynamic Service Composition
- Security Management in Distributed SOA Environments
- Trust Establishment and Trust Brokering
- Identification of Top Risks in SOA Environments
- Incorporation of Mitigation Strategies into SOA Processes and Methodologies

Cross-Cutting

- Governance
- Security
- Training and Education
- Risk Management in SOA Environments
- Social and Legal Issues

- Techniques and Guidelines to Develop SOA Governance
- Enterprise-Wide vs. Local SOA Governance
- Modeling of Policy, Risk and Trust
- Compliance Monitoring

- Establishing the Area of Services Science
- Investigating and Developing Appropriate University Curricula
Research Topics in Maintenance and Evolution of Service-Oriented Systems

What does maintenance and evolution look like in this dynamic, heterogeneous and potentially distributed development and maintenance environment?

- Process and Life Cycle
- Requirements
- Service Selection
- Service Definition and Categorization
- Technology Assessments
- Architecture and Design
- Code
- Tools and Products
- Quality Assurance and Testing
- Deployment
- Maintenance and Evolution
- Engineering Indicators

- Tools, Techniques and Environments to Support Maintenance Activities
- Multilanguage System Analysis and Maintenance
- Reengineering Processes for Migration to SOA Environments
- Transition Patterns for Service-Oriented Systems
Tools, Techniques and Environments to Support Maintenance Activities — Rationale

Complexity of the maintenance process in an SOA environment increases, especially if there are external consumers and providers involved

• Impact analysis activities for service providers have to consider a potentially unknown set of users
• Impact analysis for service implementation code has to consider direct users of the service implementation code, as well as users of the service interfaces
• Configuration management also becomes more complex, starting from the decision of what to put under configuration management
• Release cycles between services and consumers, services and infrastructure, and consumers and infrastructure ideally should be coordinated, but may not be possible when these are external
Another aspect that makes maintenance challenging is services that are shared among multiple business processes or consumers

- Who is responsible for the maintenance of a shared service?
- What happens when multiple business units have different requirements for the same service?
- How is a service evolved in the context of the multiple business processes that use it?
Not much work that specifically addresses or provides guidelines for maintenance activities in SOA environments

Maintenance Processes

• SOA Life Cycles, such as the one proposed by IBM and others, include maintenance in the post-deployment management phase of a very iterative life cycle
• Mittal recommends the use of a robust development methodology the first time the service-oriented is rolled out and the use of lighter methodologies to support ongoing maintenance
• However, there is no concrete methodology for maintenance of service-oriented systems
Change Impact Analysis

- Active area of work at different levels
  - Top-down approach to analyze the impact of changes to business processes all the way down to the source code to identify affected system components
  - Bottom-up approach is to analyze the impact of changes to a service—or its implementation—on the business processes and other consumers of the service
- Integrated development environments are starting to integrate impact analysis, but the usual assumption is that there is control and full access to all system elements
Tools, Techniques and Environments to Support Maintenance Activities — Current Efforts

Change Management and Version Control
- Area that has received a lot of attention from the research and vendor community
- Reason is that the stability of service interfaces is part of the agreement (formal or informal) between service providers and consumers
- Usually refers to versioning of the service—mainly Web Services—and not to other components of a service-oriented system

Organizational Structures and Roles
- Some preliminary research that is looking at roles and responsibilities for development, maintenance and evolution of service-oriented systems
Tools, Techniques and Environments to Support Maintenance Activities — Challenges and Gaps

Development of specialized methods and tools to support the maintenance and evolution of large service-oriented systems is in the early stages

- Current efforts seem to indicate that maintenance activities for service-oriented systems are not that different than in traditional systems
- However, we are still in the stage where most service-oriented systems are deployed for internal integration, where there is still some control over deployed services

Emergence of market for third-party services and the deployment of more service-oriented systems that cross organizational boundaries will have to change current maintenance practices
Multilanguage System Analysis and Maintenance — Rationale

One of the benefits associated to SOA, and especially Web Services, is true platform independence

- Standard interfaces are exposed but the underlying service implementation could be written in any language

While this is a huge benefit, it makes looking at the system as a whole difficult
Reengineering community has been working on this issue for a number of years to assist in analysis and migration of multi-language systems. Most work in this area is based on parsing of source code to create common higher-level representations that can then be analyzed using tools.

Some of the problems with multi-language analysis are related to the mapping between data types between different languages:

- In a Web Services environment this problem might be alleviated because XML Schema data types are used at the service interface level.
Multilanguage System Analysis and Maintenance — Challenges and Gaps

Most research in this area is limited to small size projects and a small number of languages, which is a problem for an environment that promotes platform independence.

In the case of third-party service providers, access to source code is most probably not possible.

- If that is the case, an important area of research is the identification of the type of information that service providers would need to expose to service consumers that wish to do code analysis, as well as tools to support the process.
Reengineering Processes for Migration to SOA Environments — Rationale

Migration of legacy systems to SOA environments has been achieved within a number of domains, including banking, electronic payment, and development tools, showing that the promise is beginning to be fulfilled.

While migration can have significant value, any specific migration requires a concrete analysis of the feasibility, risk and cost involved.

The strategic identification and extraction of services from legacy code is crucial as well.
There are not many reengineering techniques that focus on a “full-circle” model, such as the "SOA-Migration Horseshoe" proposed by Winter and Ziemann

This approach integrates software reengineering techniques with business process modeling.
Reengineering Processes for Migration to SOA Environments — Current Efforts

The larger amount of work is on techniques in the “bottom portion” of the horseshoe for exposing legacy functionality as services, mainly Web Services.

Tools to support this type of migration are available as language libraries and/or integrated into common IDEs such as the Eclipse WTP and the .NET development environment, or as part of infrastructure products such as Apache Axis.
Reengineering Processes for Migration to SOA Environments — Current Efforts

Some work on techniques and research proposals that take into consideration business goals and drivers—these techniques work in the “top portion” of the horseshoe

- Service Migration and Reuse Technique (SMART)—Output is a migration strategy that includes preliminary estimates of cost and risk and a list of migration issues
- Ziemann et. al. propose a business-driven legacy-to-SOA approach based on enterprise modeling that considers both the business and legacy system aspects
- IBM has a method called Service Oriented Modeling and Analysis (SOMA) that focuses on full system development but has some portions that address legacy reuse
- Cetin et. al. propose a mashup-based approach for migration of legacy software to pervasive service-oriented computing platforms
Reengineering Processes for Migration to SOA Environments — Current Efforts

There is work related to the identification of services in legacy code, addressing the “left portion” of the horseshoe:

- In the context of Web Services, Aversano et. al. propose an approach that combines information retrieval tracing with structural matching of the target WSDL with existing methods.
- Also in the context of Web Services, Sneed proposes an approach that consists of salvaging the legacy code, wrapping the salvaged code and making the code available as a web service.
  - In the salvaging step he proposes a technique for extracting services based on identifying business rules that produce a desired result.
Reengineering Processes for Migration to SOA Environments — Challenges and Gaps

The ideal reengineering process would be one that implements the SOA-Migration Horseshoe

- Currently techniques and tools that implement portions of the horseshoe but not the full horseshoe
- An important area of research would be the development of concrete processes that implement the horseshoe and tools (or suites of tools) to support the process

Real challenge is mining legacy code for services that have business value

- Tools and techniques for analyzing large source code bases to discover code that is of business value
- Metrics for "wrapability" and business value to determine reusability
- Application of feature extraction techniques to service identification
Transition Patterns for Service-Oriented Systems — Rationale

There is general agreement that the migration of legacy systems to service-oriented systems should be done incrementally.

However, this means that throughout the life cycle of the project there will be a mix of

- Migrated legacy components
- Legacy components waiting to be migrated
- Legacy components that will not be migrated

Legacy components include

- Application front ends
- Business logic
- Data logic
- Data

Major challenge: Minimization of “throw-away” cost and effort to support intermediate system states
Transition Patterns for Service-Oriented Systems — Current Efforts

Active work in academia and industry related to incremental modernization and enterprise transformation by moving legacy systems to SOA environments.

Although not directly related, Thomas Erl has developed several design patterns that can be used when legacy systems are part of service-oriented systems, e.g.

- Service Façade
- Service Data Replication
- Legacy Wrapper
- File Gateway

Architectural reconstruction and program analysis techniques are able to isolate “chunks” of code and discover dependencies between components.
Transition Patterns for Service-Oriented Systems — Challenges and Gaps

If a new system is built to replace the old system in $n$ steps then **Total Modernization Cost (TMC)** is

$$ TMC = \sum_{i=1}^{n} (DC_i + IC_i + DMC_i + TC_i) $$

where

- $DC = $ Development Cost
- $IC = $ Implementation Cost (Deployment + Infrastructure)
- $DMC = $ Data Migration Cost
- $TC = $ Transition Cost

And, if we define **Transition Cost** as $TC = ECC + NTC + TI$, where

- $ECC = $ Existing Code Changes + Maintenance Cost
- $NTC = $ New Throwaway Code + Maintenance Cost
- $TI = $ Temporary Infrastructure and Operation Cost

Source: Ongoing work between Grace A. Lewis from Software Engineering Institute and Parviz Dousti from Carnegie Mellon University
What would be the process and supporting techniques and tools that could determine

- The **right number of increments**
- that would **minimize the throwaway costs** due to
  - Temporary infrastructure such as gateways and ETL tools
  - Temporary code to deal with mismatches
  - Changes to legacy code waiting to be modernized, e.g. adding code to invoke a service knowing that it will be modernized in a future increment
- **in a repeatable fashion** such that it could be recalculated when business changes its mind about priorities

Source: Ongoing work between Grace A. Lewis from Software Engineering Institute and Parviz Dousti from Carnegie Mellon University
Conclusions on Key Challenges

Engineering challenges are significant if SOA is to be used in “advanced ways”, e.g.
  - Multi-organizational
  - Dynamic discovery and composition
  - Real-time applications

Main challenges for enterprise applications are related to business and operations, and not engineering. As third-party services become the new business model, there needs to be support for
  - Service-level agreements
  - Runtime monitoring
  - End-to-end testing involving third parties
  - Pricing models for third-party services
  - Service usability—from a design and an adoption perspective
Conclusions on Key Challenges

There are some areas where what is needed is not more basic research, but rather non-vendor surveys, studies and experiments to produce more concrete guidance:

- SOA governance
- Business case for SOA adoption
- ROI for SOA adoption
- Development processes and practices for service-oriented systems development

There needs to be more collaborative research between industry and academia to create real practices.
Conclusions on Maintenance and Evolution of Service-Oriented Systems

In the short term, maintenance and evolution practices will have to evolve and adapt to support this dynamic and changing environment, taking into consideration the emergence of third-party services over which there is less control and visibility.

Good starting points

- Tools and techniques to support maintenance and evolution activities in these environments
- Reengineering processes that combine business as well as technical aspects
- Capabilities for multi-language analysis