Migration to Service Oriented Architecture (SOA) with Selected Research Challenges

Dennis Smith

ICSOC 2008

December 5, 2008

Agenda

Introduction

- SOA Challenges
- Common Misconceptions
- Consequences of Decisions

Introduction to SOA Research Agenda

Pillars of Service-Oriented Systems Development

Challenges of Migration to SOA Environments

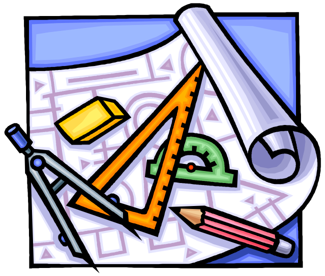
SMART (Service Migration and Reuse Technique)

Conclusions

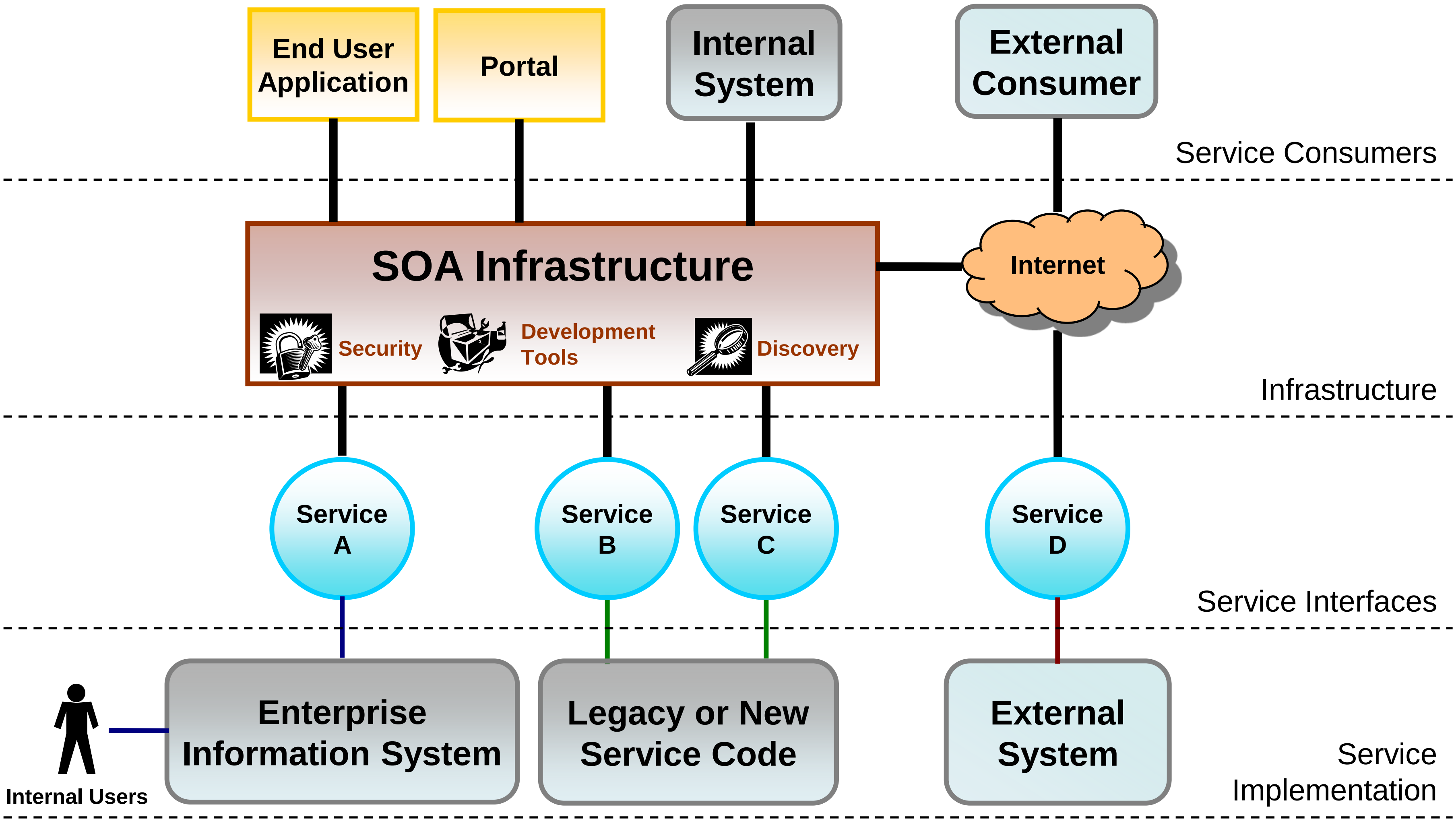
What is SOA?

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.



Components of a Service-Oriented System



Challenges for Service Consumers

Available services might not meet functional and non-functional requirements.

Services may change or disappear without notification.

Tools and programs provided by the infrastructure may conflict with development environment.

Services may not be semantically correct from the consumer’s point of view.

Services coming from different organizations can have inconsistencies between them.

End-to-end testing would require test instances of all services to be available.

Challenges for Service Developers

If consumer requirements are not understood, services may never be used.

The effort to translate legacy data types into data types that can be transmitted in messages can be greater than expected.

If dealing with proprietary SOA environments, there may be

- Constraints imposed on developed services
- Dependencies on tools and programs provided by the infrastructure that are in conflict with development tools

Guidance for using Service-Level Agreements (SLAs) is often not clear.

- Benefits of SLAs are not well quantified.

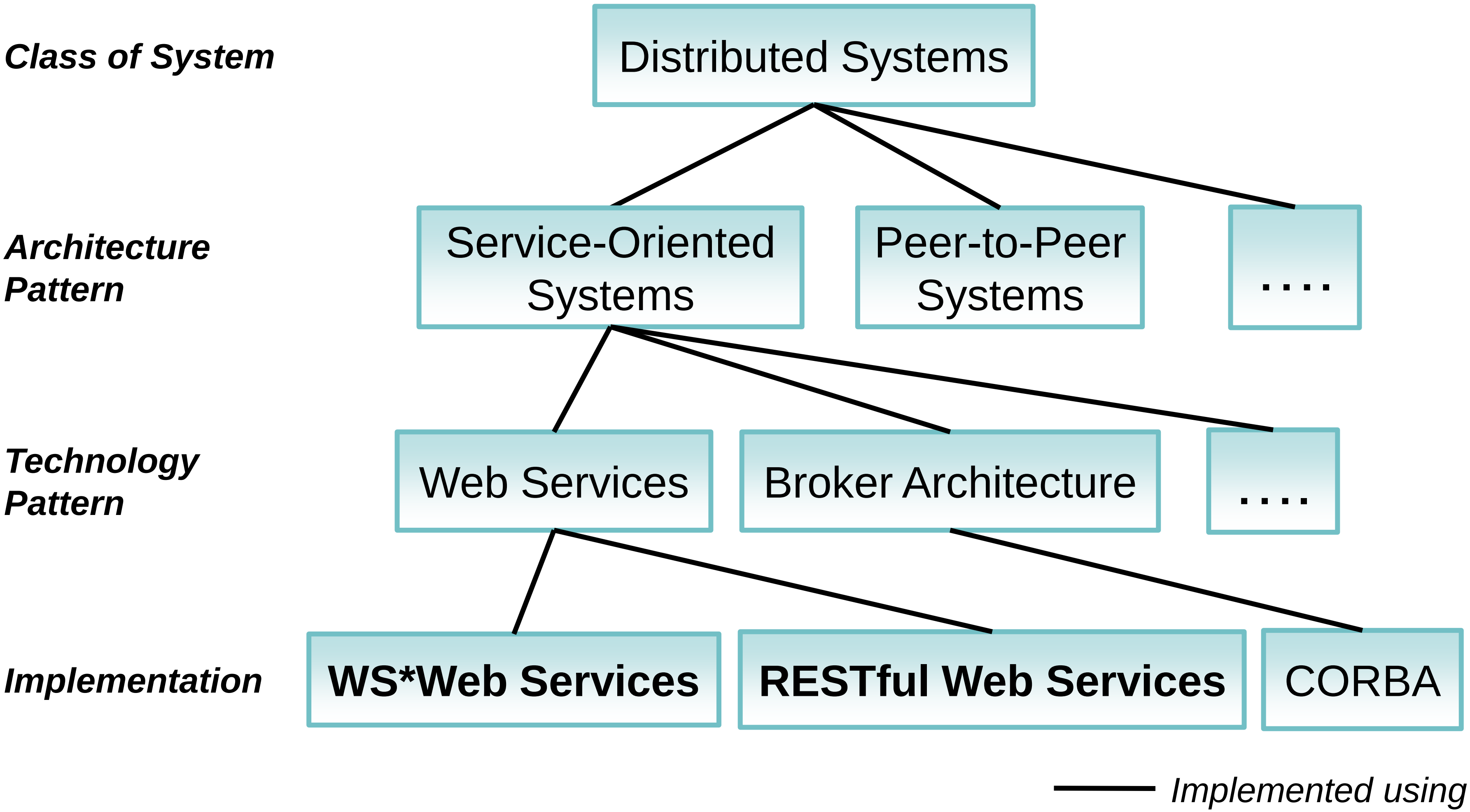
Challenges for Infrastructure Developers

Changes in standards and products used in the infrastructure may have a large impact on its users.

- Especially emerging standards

Effort for development, support, and training for the use of tools and infrastructure may be underestimated.

Web Services in the Context of Distributed Systems



Class of System

Distributed Systems

Architecture Pattern

Service-Oriented Systems

Peer-to-Peer Systems

....

Technology Pattern

Web Services

Broker Architecture

....

Implementation

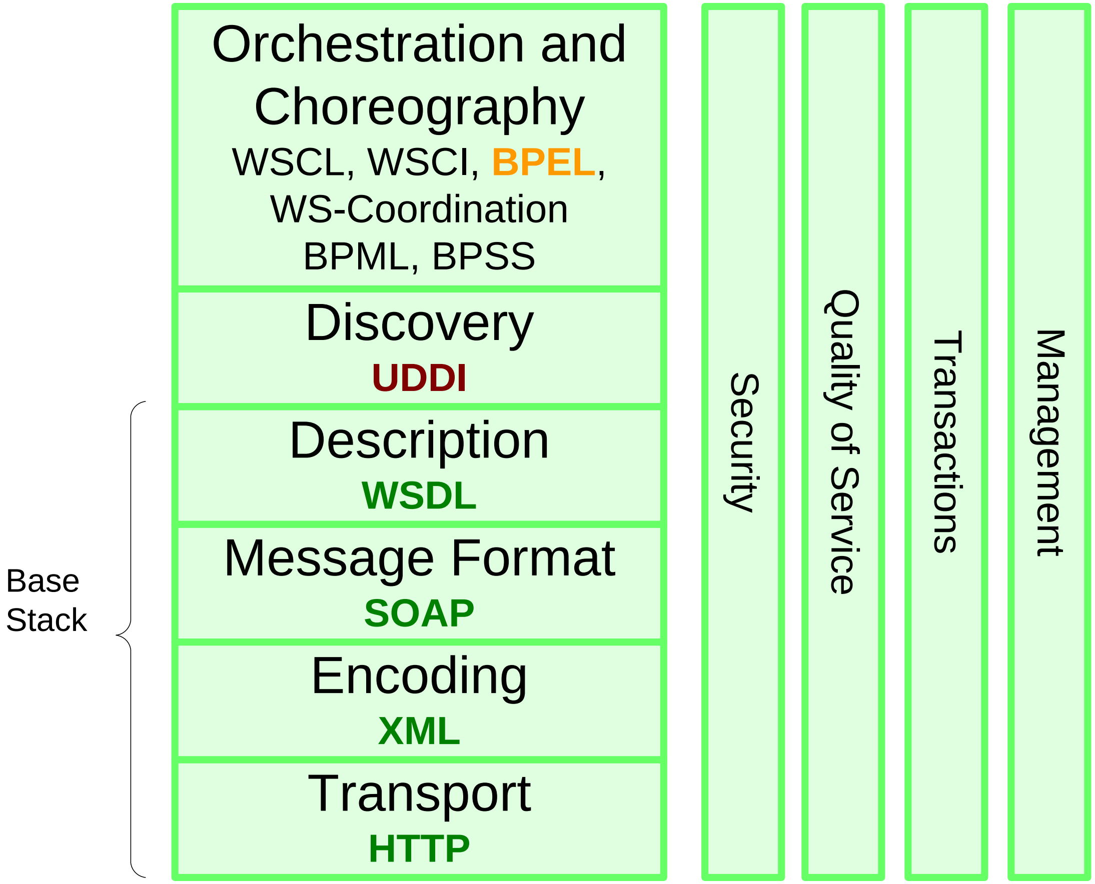
WS*Web Services

RESTful Web Services

CORBA

Implemented using

WS* Protocol Stack



Orchestration and Choreography
WSCL, WSCI, BPEL, WS-Coordination BPML, BPSS

Discovery
UDDI

Description
WSDL

Message Format
SOAP

Encoding
XML

Transport
HTTP

Base Stack

Security

Quality of Service

Transactions

Management

Adapted from “XML and Web Services Unleashed”, SAMS Publishing

The highlighted standards are the most commonly used.

Most WS* standards are emerging and even competing.

Security, QoS, Transactions, and Management have to be addressed in all layers.

Agenda

Introduction

- SOA Challenges
- Common Misconceptions
- Consequences of Decisions

Introduction to SOA Research Agenda

Pillars of Service-Oriented Systems Development

Challenges of Migration to SOA Environments

SMART (Service Migration and Reuse Technique)

Conclusions

SOA Provides the Complete Architecture for a System

SOA is an architectural pattern/style/paradigm and not the architecture of the system itself.

An architectural pattern provides guidance that embodies best practices.

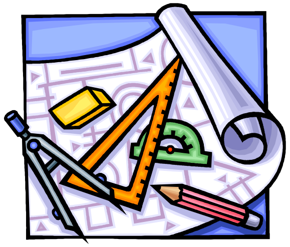
- The concrete elements and their interactions are the architecture of the system.

Any number of systems can be developed based on an architectural pattern.

- An architecture based on SOA inherits both the good and the bad.

Corollary: SOA cannot be bought off-the shelf.

- System qualities have to be built into the architecture of the system.
- Decisions have to be made—service design and implementation, technologies, tradeoffs.



The Use of Standards Guarantees Interoperability in an SOA environment

Interoperability needs agreement on both syntax and semantics.

Web Services enable syntactic interoperability.

- XML Schema defines structure and data types.
- WSDL defines the interfaces: operations, parameters and return values.
- Available information, technologies, and tool support.

Web Services do not guarantee semantic interoperability.

- XML and WSDL do not define the meaning of data.
- WSDL does not define what a service does.
- It is an active research area—unresolved issues.

It Is Very Easy To Develop Applications Based on Services

It is relatively easy to build applications and services that work with a particular infrastructure . . . but designing a "good" service might not be that easy.

From a service provider perspective

- Not many best practices for designing services
  - What is the right granularity?
  - What is the right Quality of Service (QoS)? Can you guarantee it?
- Have to know and anticipate potential consumers and usage patterns
  - "If you build it they will come" – Can you afford this?

From a service consumer perspective

- Ease depends on tool availability for SOA infrastructure.
- Larger granularity may lead to larger incompatibilities.
- Most difficult part is composition—data and process mismatches.

A Service Registry Allows Service Binding Dynamically at Runtime

Current technologies have not advanced to the point that this is possible in production environments.

Requires the use of a common formal ontology by service providers and consumers within a domain.

- Data model that represents a set of concepts within a domain and the relationships between those concepts (from Wikipedia)

Requires the construction of intelligent service consumers that

- Construct the right queries for the discovery of services
- Compose services when there is not a single service that can process the request
- Provide the right data to invoke a service that was discovered at runtime

Agenda

Introduction

- SOA Challenges
- Common Misconceptions
- Consequences of Decisions

Introduction to SOA Research Agenda

Pillars of Service-Oriented Systems Development

Challenges of Migration to SOA Environments

SMART (Service Migration and Reuse Technique)

Conclusions

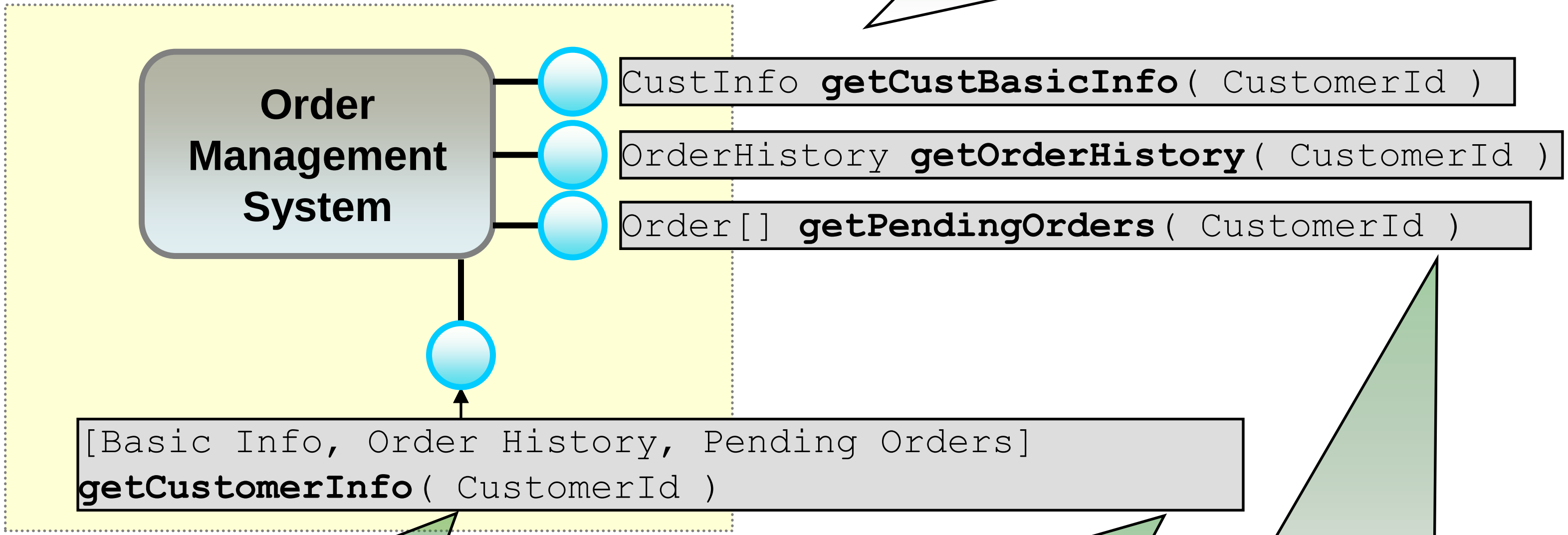
Sample Consequences of Decisions: Service Granularity 1

The granularity of service interfaces can affect the end-to-end performance of systems because services are executed across a network as an exchange of a service request and a service response.

- If service interfaces are too coarse-grained, consumers will receive more data than they need in their response message.
- If service interfaces are too fine-grained, consumers will have to make multiple trips to the service to get all the data they need.

Sample Consequences of Decisions: Service Granularity 2

… or the service can be more granular and provide three different operations for each type of information



```
CustInfo getCustBasicInfo( CustomerId )
OrderHistory getOrderHistory( CustomerId )
Order[] getPendingOrders( CustomerId )
[Basic Info, Order History, Pending Orders]
getCustomerInfo( CustomerId )
```

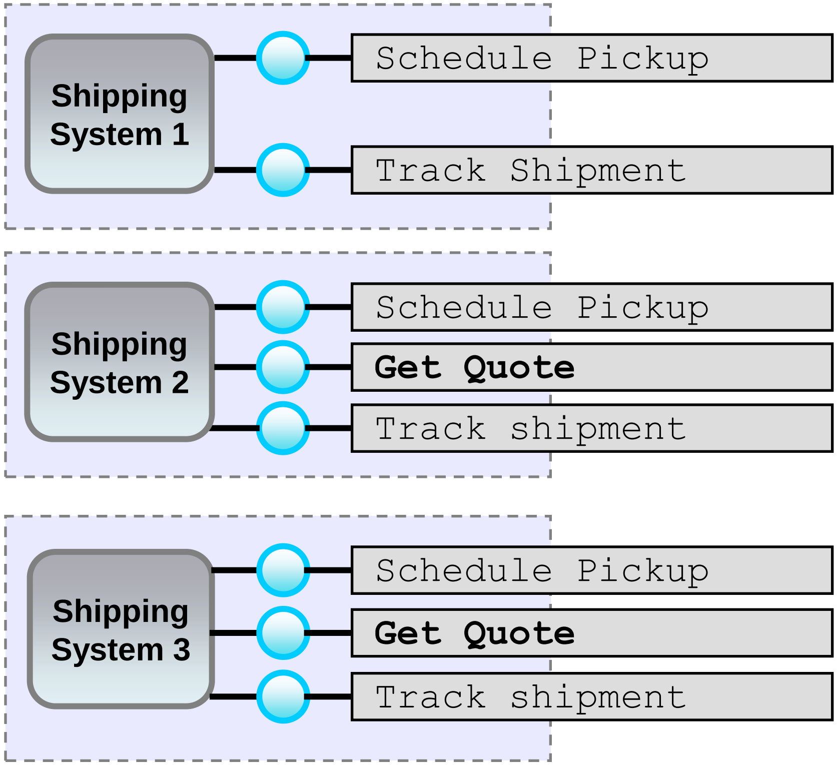
The Order Management System can expose the business functionality of getting all the customer information in one call …

… or all four operations can be implemented where getCustomerInfo is a composite service, but the individual services are available as well.

Sample Consequences of Decisions: Requirements 1

If service developers do not understand functionality and QoS needs of potential users of services, they might end up developing and deploying services that are never used.

Sample Consequences of Decisions: Requirements $_2$



If Shipping System 1 does not implement the Get Quote functionality, consumers cannot "automatically" decide on the cheapest option.

This can result in potential revenue loss for the shipping system.

Sample Consequences of Decisions: Transaction Management 1

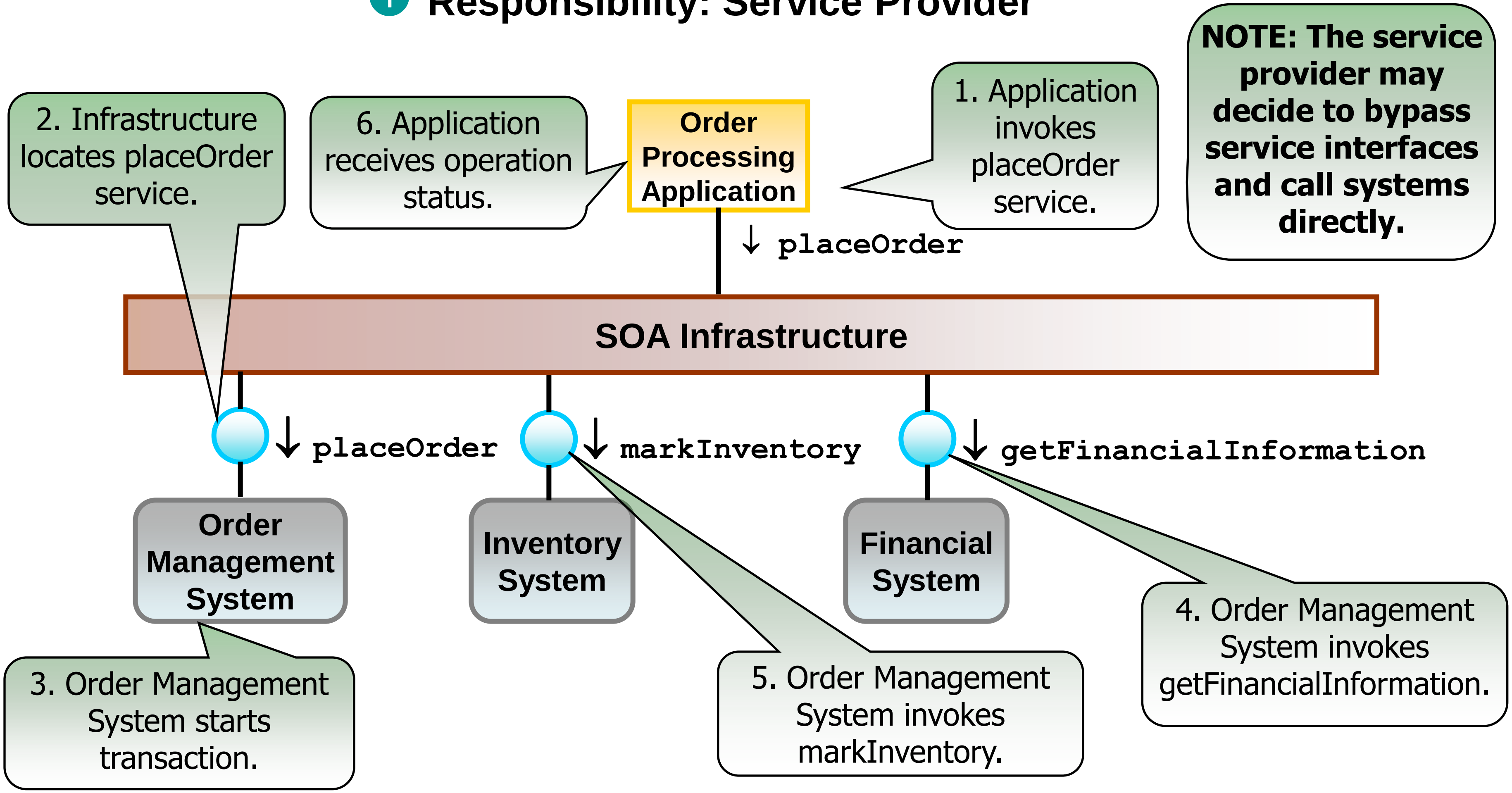
The decision of where to assign responsibility for transaction management has an effect on development.

Scenario

- Order Processing application needs to place an order.
- Three systems are involved
  - The Order Management System controls order creation
  - The Financial System contains customer financial information
  - The Inventory System contains part information and stock
- An order is considered complete after the customer financial status is verified and the parts in inventory are marked for shipment.

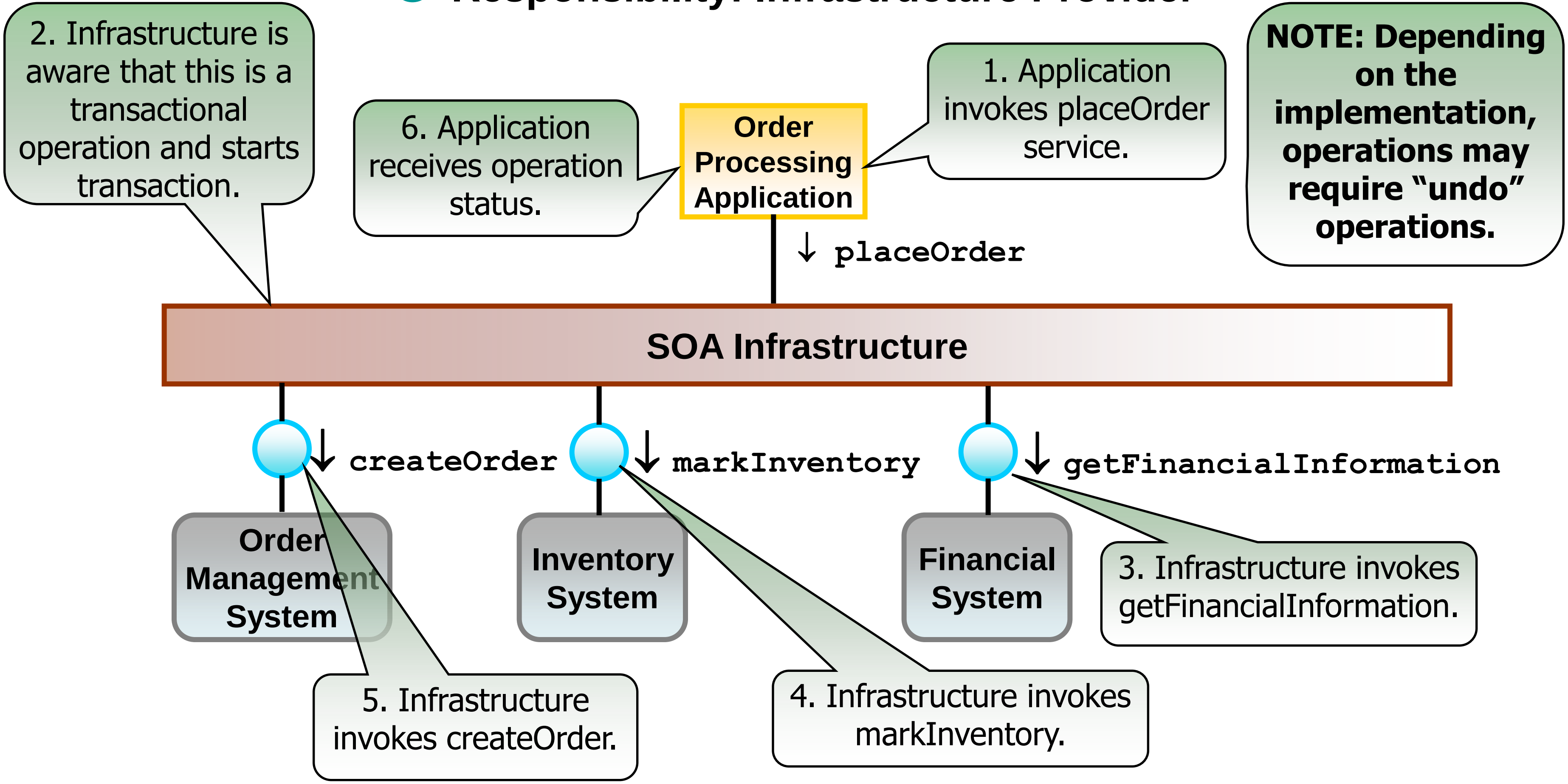
Sample Consequences of Decisions: Transaction Management 2

➊ Responsibility: Service Provider


NOTE: The service provider may decide to bypass service interfaces and call systems directly.
1. Application invokes placeOrder service.
2. Infrastructure locates placeOrder service.
3. Order Management System starts transaction.
4. Order Management System invokes getFinancialInformation.
5. Order Management System invokes markInventory.
6. Application receives operation status.
Order Processing Application
↓ placeOrder
SOA Infrastructure
↓ placeOrder
↓ markInventory
↓ getFinancialInformation
Order Management System
Inventory System
Financial System

Sample Consequences of Decisions: Transaction Management 3

❷ Responsibility: Infrastructure Provider



1. Application invokes placeOrder service.

2. Infrastructure is aware that this is a transactional operation and starts transaction.

3. Infrastructure invokes getFinancialInformation.

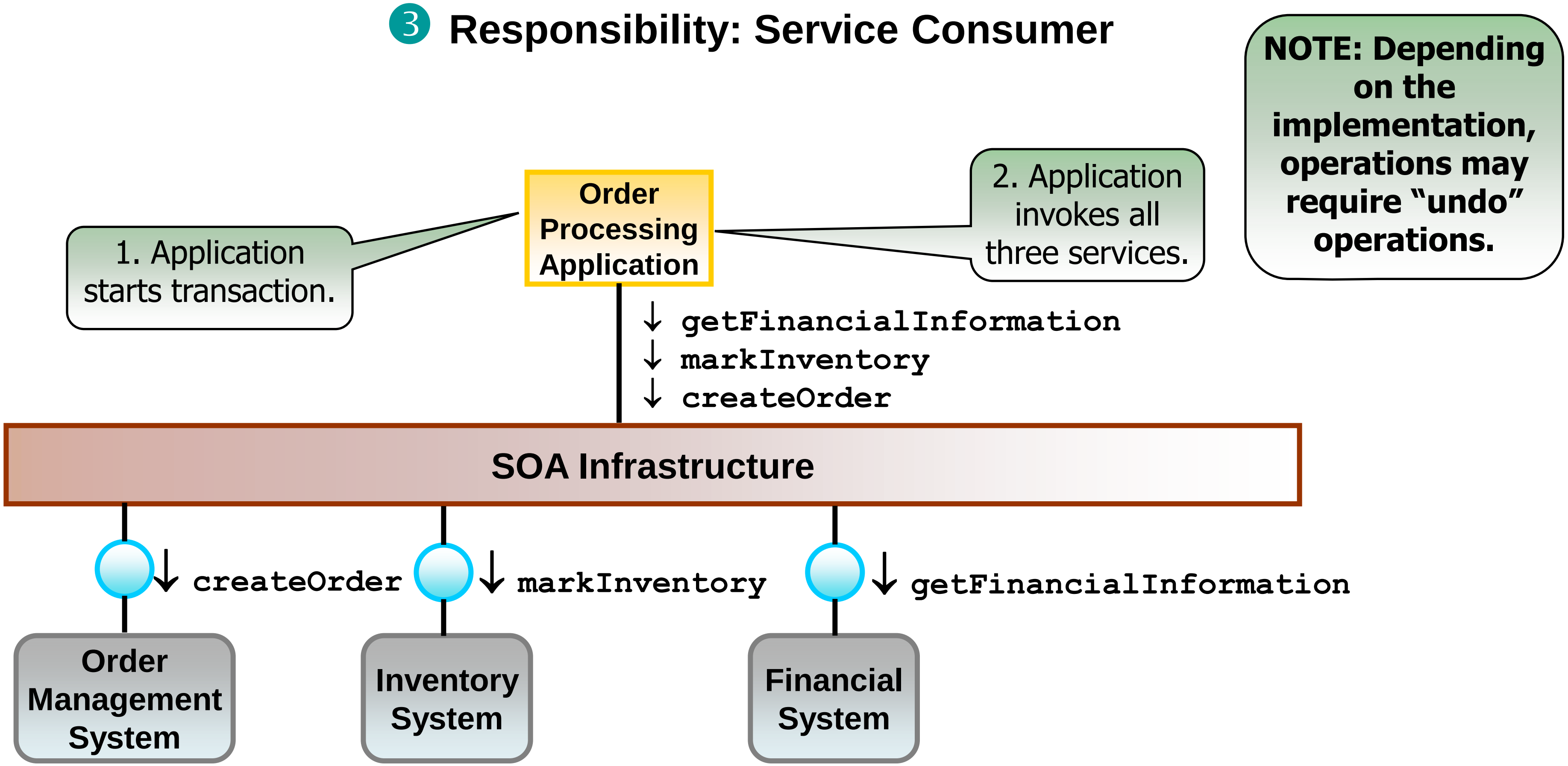
4. Infrastructure invokes markInventory.

5. Infrastructure invokes createOrder.

6. Application receives operation status.

NOTE: Depending on the implementation, operations may require "undo" operations.

Sample Consequences of Decisions: Transaction Management 4


❸ Responsibility: Service Consumer
NOTE: Depending on the implementation, operations may require “undo” operations.
Order Processing Application
1. Application starts transaction.
2. Application invokes all three services.
↓ getFinancialInformation
↓ markInventory
↓ createOrder
SOA Infrastructure
↓ createOrder
↓ markInventory
↓ getFinancialInformation
Order Management System
Inventory System
Financial System

Agenda

Introduction

- SOA Challenges
- Common Misconceptions
- Consequences of Decisions

Introduction to SOA Research Agenda ⇦

Pillars of Service-Oriented Systems Development

Challenges of Migration to SOA Environments

SMART (Service Migration and Reuse Technique)

Conclusions

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

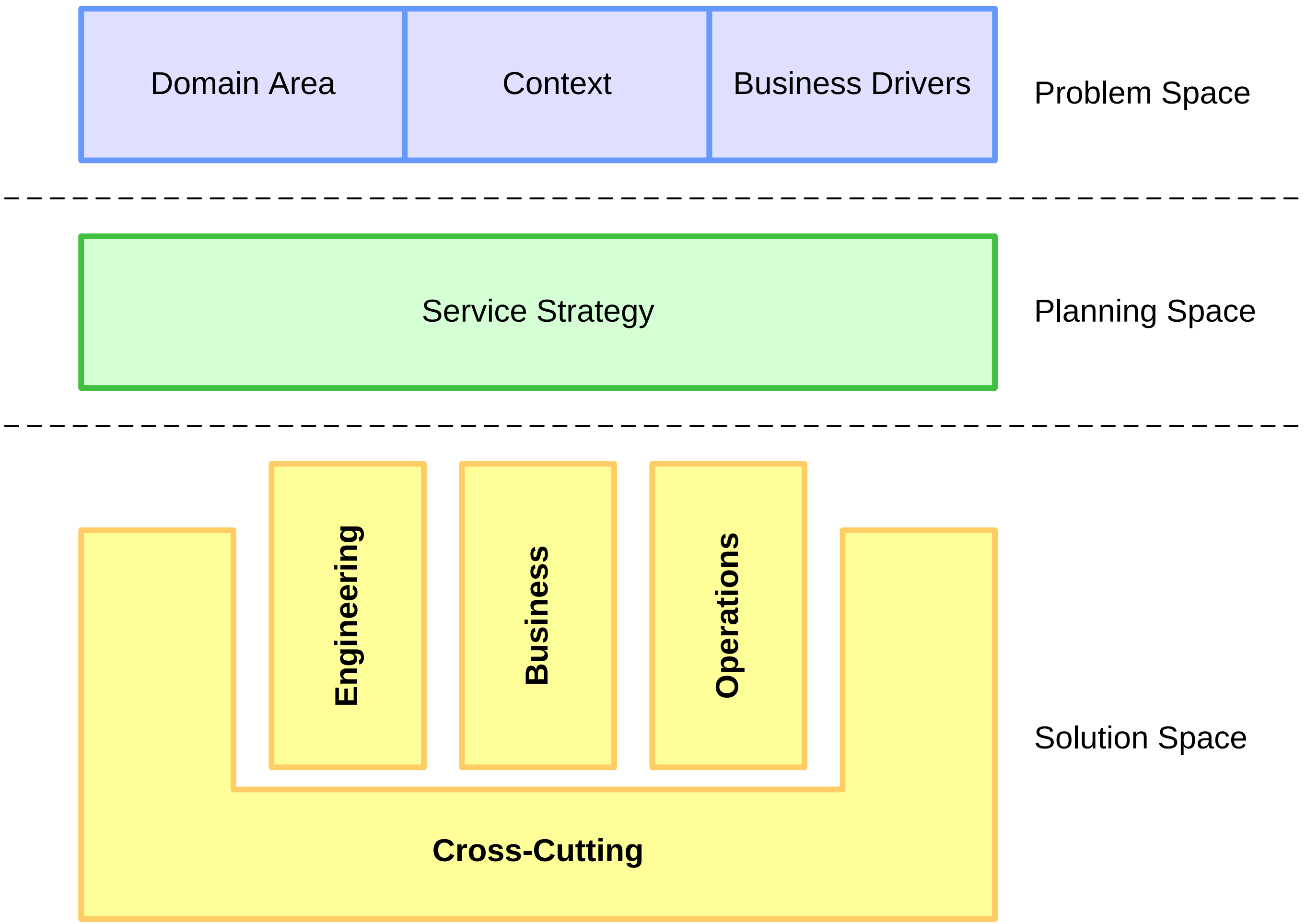
Performed an extensive literature review and looked at case studies of successful SOA adoption

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

Identified areas of SOA research necessary to fill in the gaps

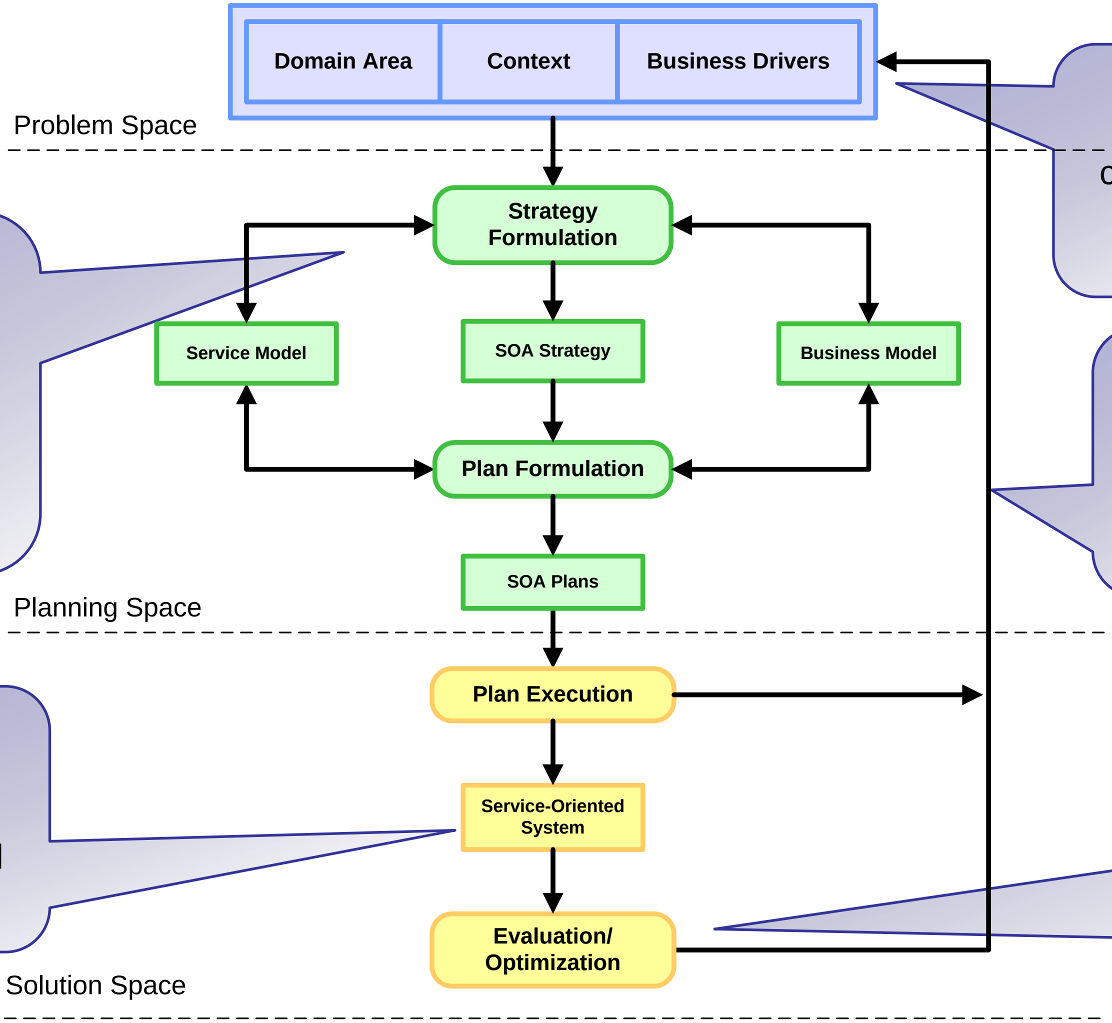
Evolved findings through multiple workshops

SOA Problem and Solution Space



Successful SOA adoption has a strong link between business strategy and SOA strategy

Expanded View of the SOA Problem and Solution Space



The organization’s domain area and context enable and/or constrain the SOA strategy

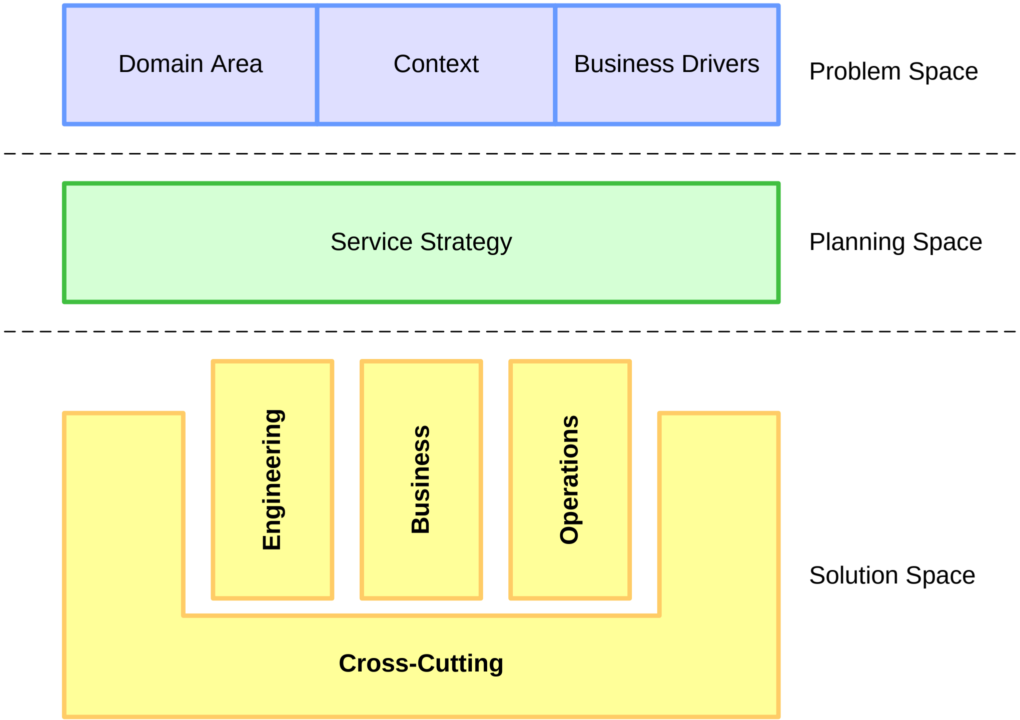
SOA strategy is the way in which SOA is going to address the organization's business drivers for SOA adoption

Feedback loops reflect the dynamic nature of service-oriented environments

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

Measurements are gathered to test the effectiveness of the strategy and the system itself

Relationship between Solution Space and Research Topics

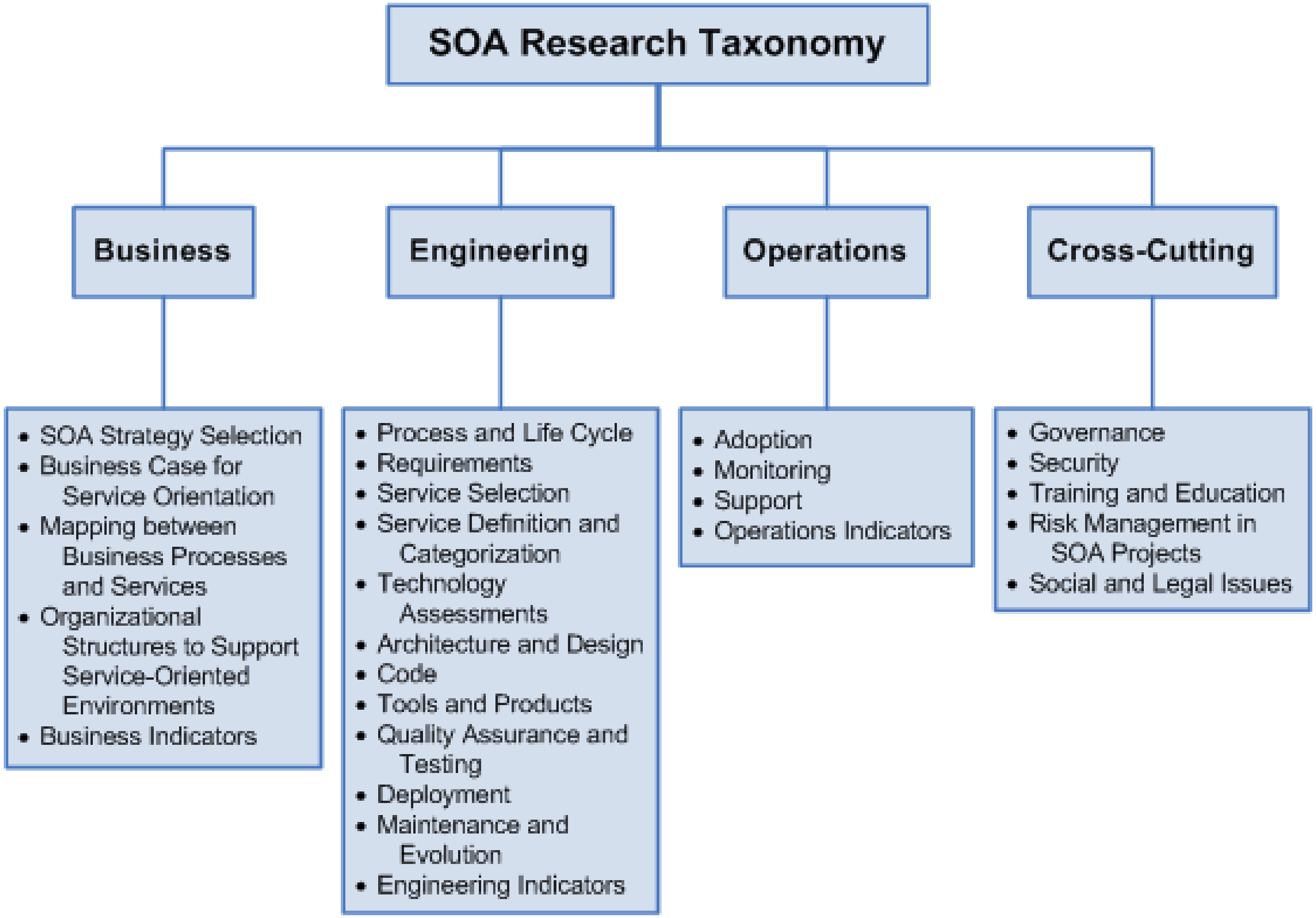


The development of a service-oriented system requires business, engineering and operations to be made, as well as other cross-cutting decisions.

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

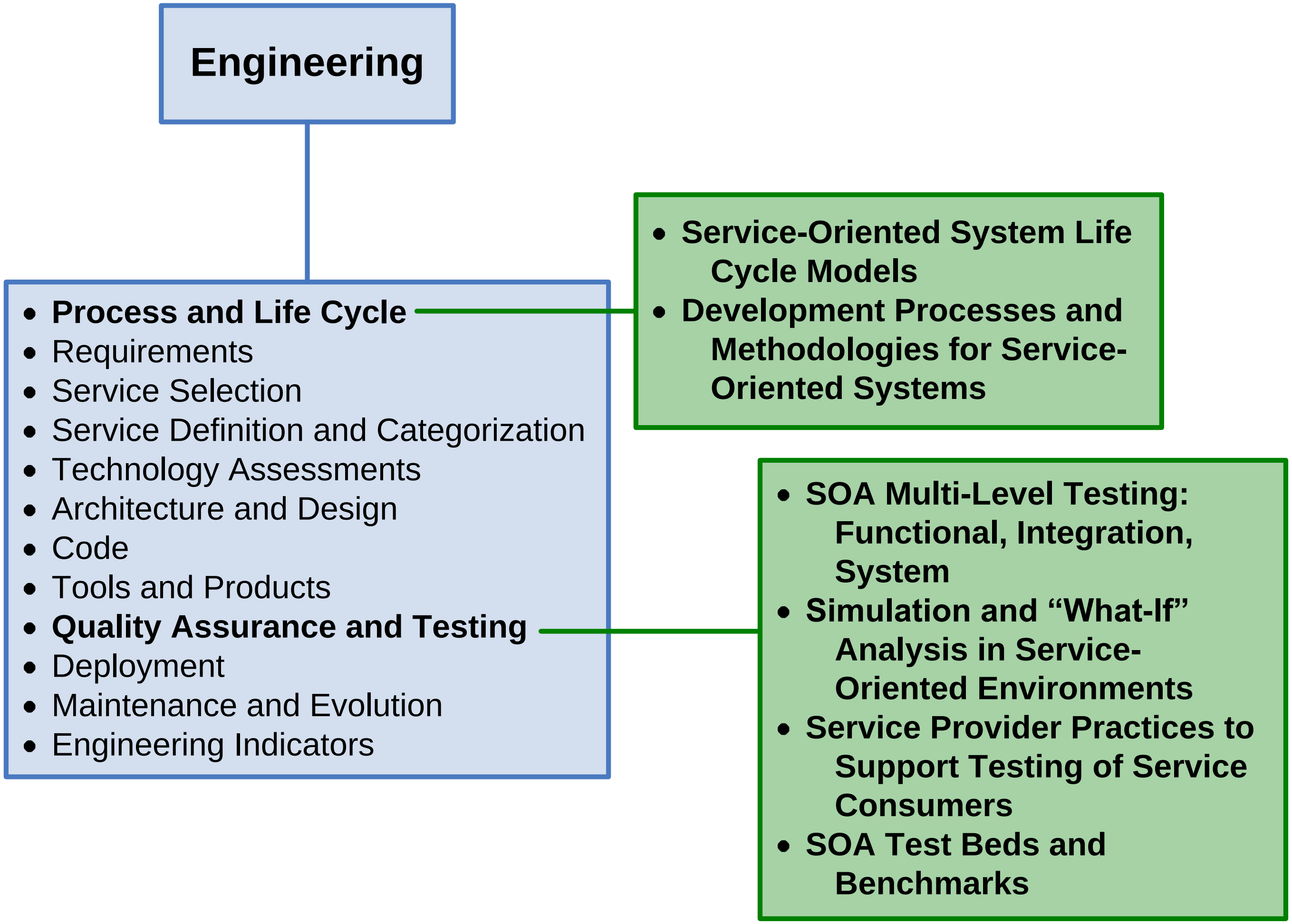
Operations

- Adoption
- Monitoring
- Support
- Operations Indicators

Cross-Cutting

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

Sample of Engineering Research Topics



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

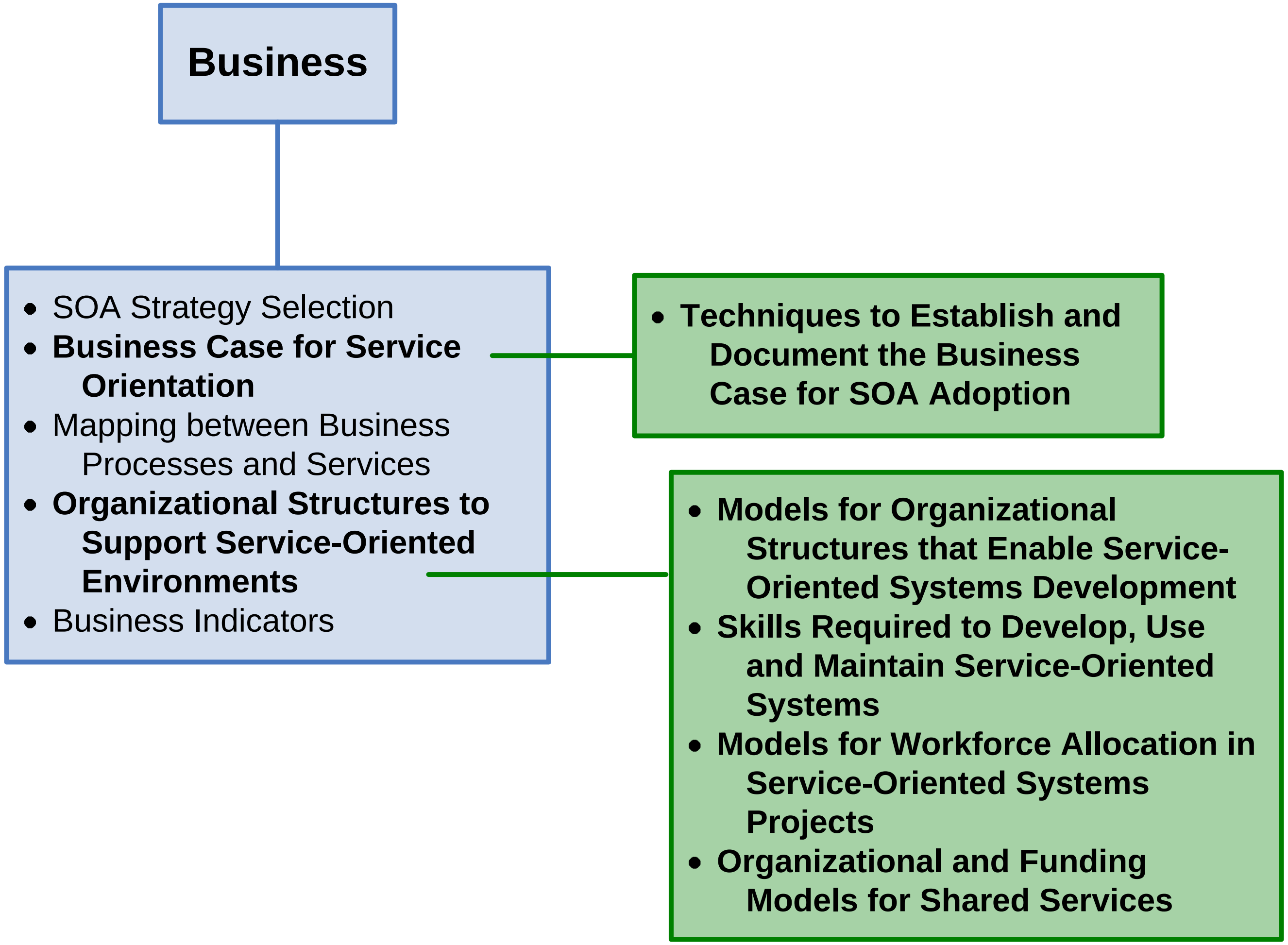
Process and Life Cycle:

- **Service-Oriented System Life Cycle Models**
- **Development Processes and Methodologies for Service-Oriented Systems**

Quality Assurance and Testing:

- **SOA Multi-Level Testing: Functional, Integration, System**
- **Simulation and “What-If” Analysis in Service-Oriented Environments**
- **Service Provider Practices to Support Testing of Service Consumers**
- **SOA Test Beds and Benchmarks**

Sample of Business Research Topics



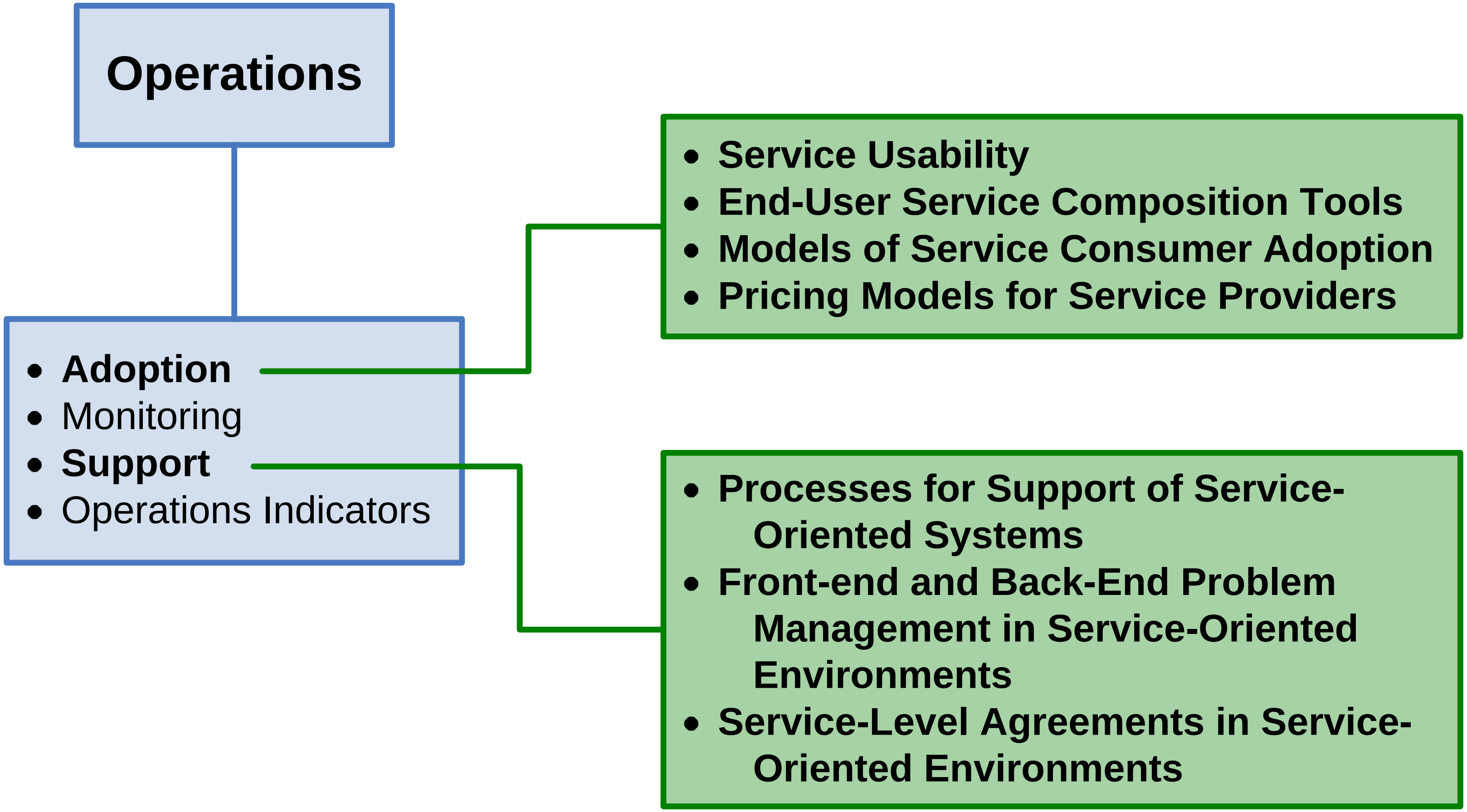
Business

- 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**

- **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



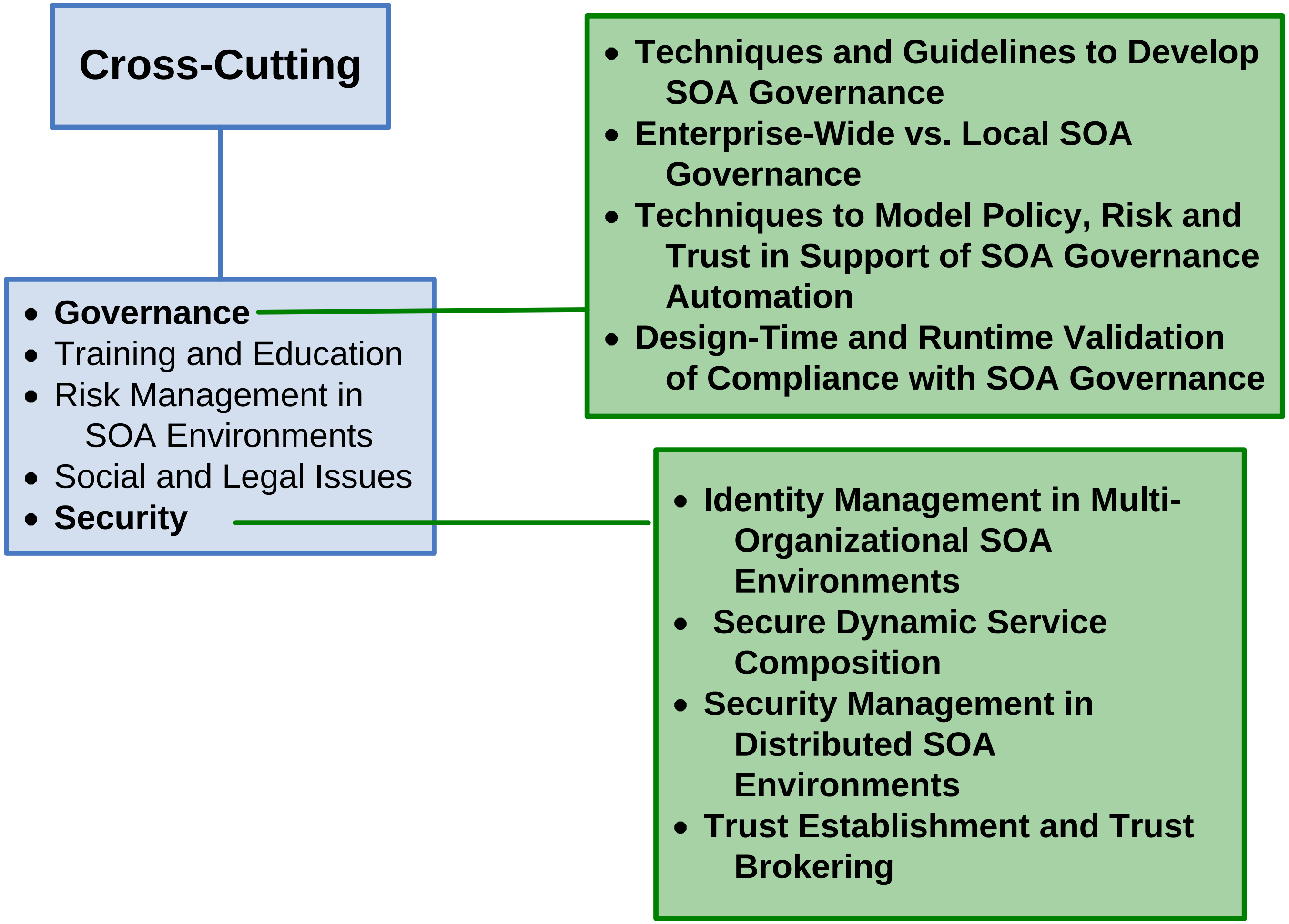
Operations

- **Adoption**
- Monitoring
- **Support**
- 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**
- **Service-Level Agreements in Service-Oriented Environments**

Sample of Cross-Cutting Research Topics



Cross-Cutting

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

Governance:

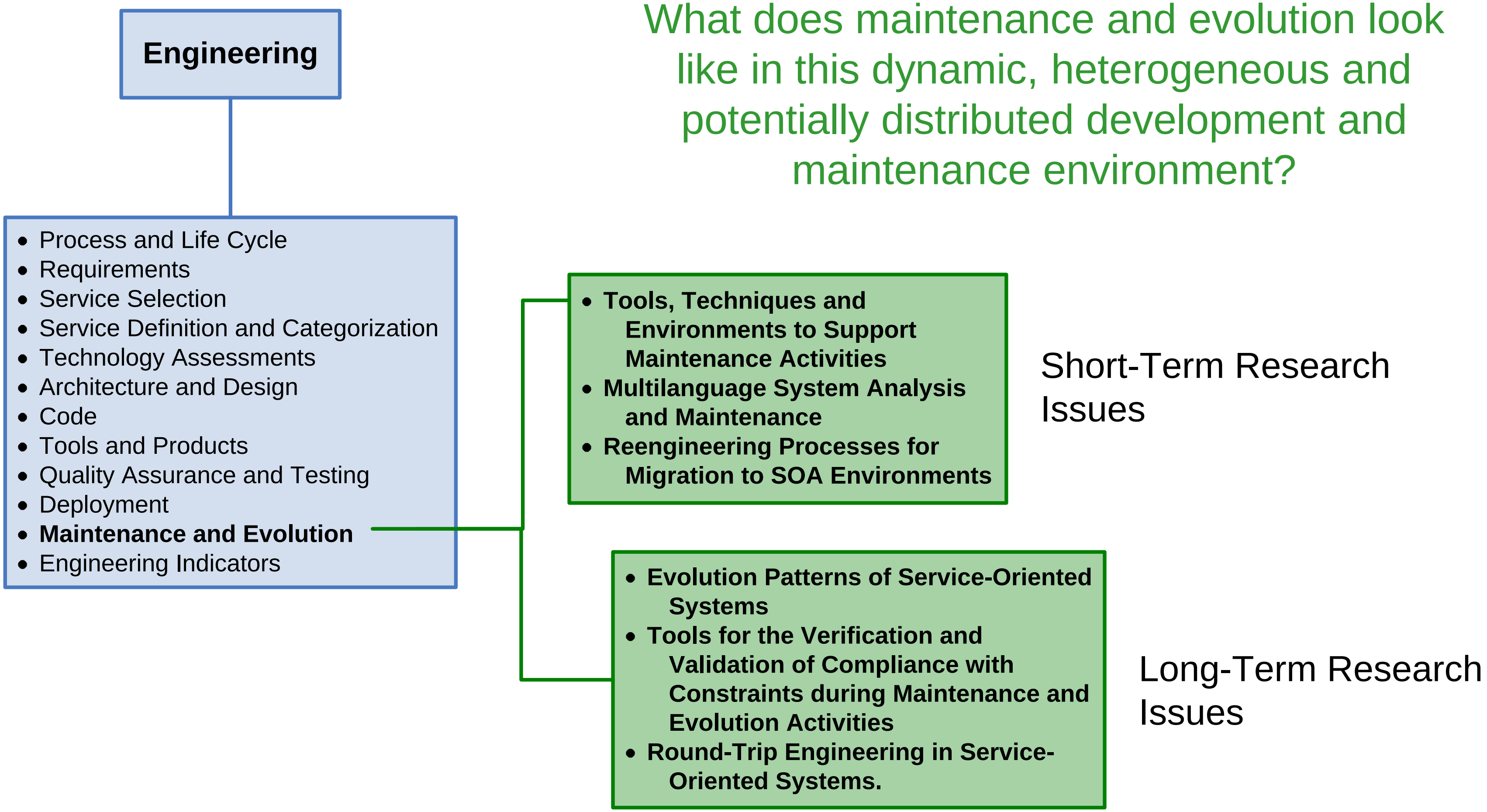
- **Techniques and Guidelines to Develop SOA Governance**
- **Enterprise-Wide vs. Local SOA Governance**
- **Techniques to Model Policy, Risk and Trust in Support of SOA Governance Automation**
- **Design-Time and Runtime Validation of Compliance with SOA Governance**

Security:

- **Identity Management in Multi-Organizational SOA Environments**
- **Secure Dynamic Service Composition**
- **Security Management in Distributed SOA Environments**
- **Trust Establishment and Trust Brokering**

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?



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

- **Tools, Techniques and Environments to Support Maintenance Activities**
- **Multilanguage System Analysis and Maintenance**
- **Reengineering Processes for Migration to SOA Environments**

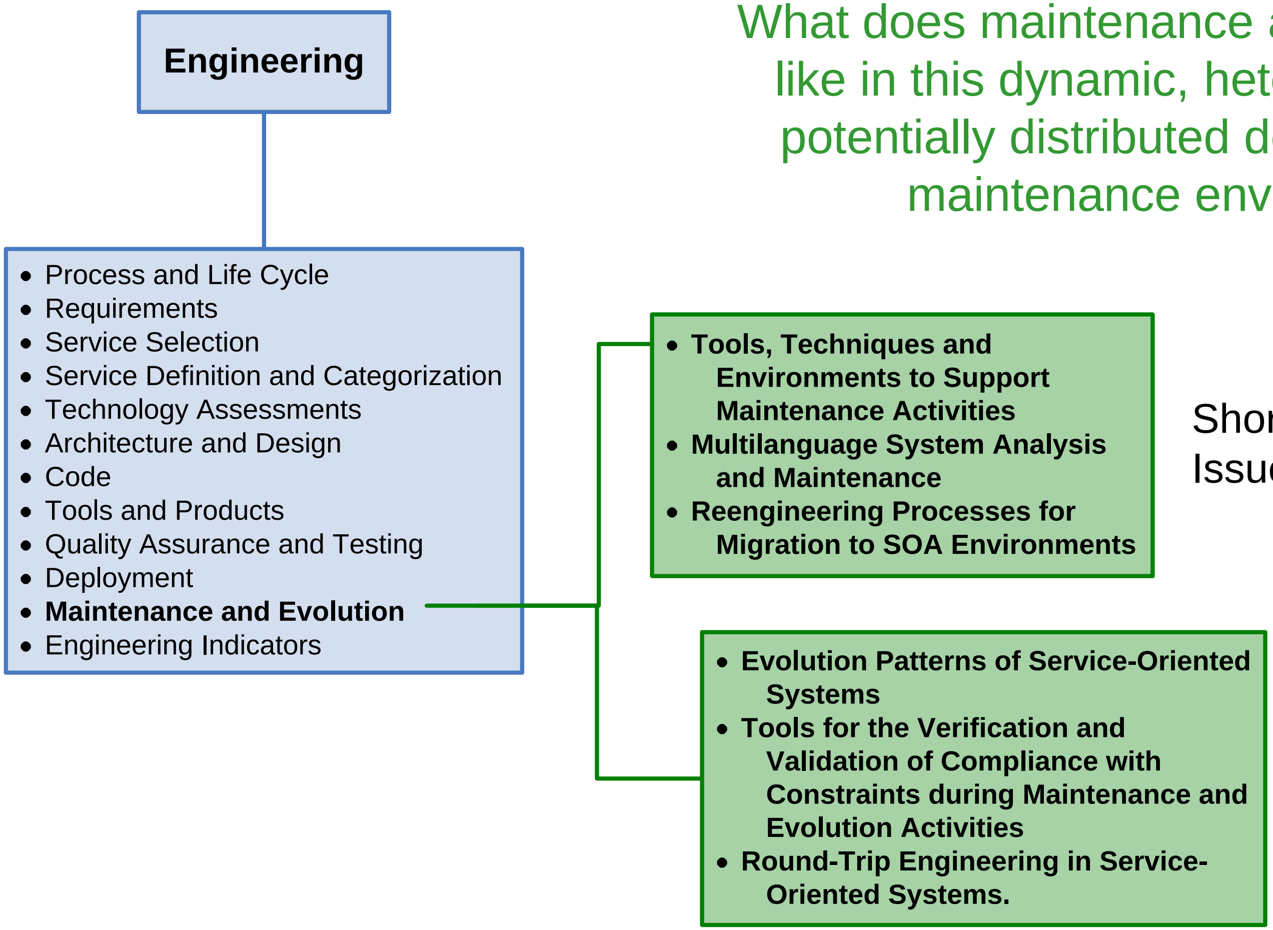
Short-Term Research Issues

- **Evolution Patterns of Service-Oriented Systems**
- **Tools for the Verification and Validation of Compliance with Constraints during Maintenance and Evolution Activities**
- **Round-Trip Engineering in Service-Oriented Systems.**

Long-Term Research Issues

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?



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

- **Tools, Techniques and Environments to Support Maintenance Activities**
- **Multilanguage System Analysis and Maintenance**
- **Reengineering Processes for Migration to SOA Environments**

Short-Term Research Issues

- **Evolution Patterns of Service-Oriented Systems**
- **Tools for the Verification and Validation of Compliance with Constraints during Maintenance and Evolution Activities**
- **Round-Trip Engineering in Service-Oriented Systems.**

Long-Term Research Issues

Tools, Techniques and Environments to Support Maintenance Activities—Rationale $_1$

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

Tools, Techniques and Environments to Support Maintenance Activities—Rationale 2

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?

Tools, Techniques and Environments to Support Maintenance Activities—Current Efforts 1

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

Tools, Techniques and Environments to Support Maintenance Activities—Current Efforts 3

Change Management and Version Control

- Area that has received a lot of attention from the research and vendor community [Brown, Evdemon, Lhotaka, Lublisnky, Peltz, Robinson]
- 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 [Kajko-Mattsson]

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

Tools, Techniques and Environments to Support Maintenance Activities—Current Efforts $_2$

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 [Xiao]
  - 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 [Zhang]
- 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

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

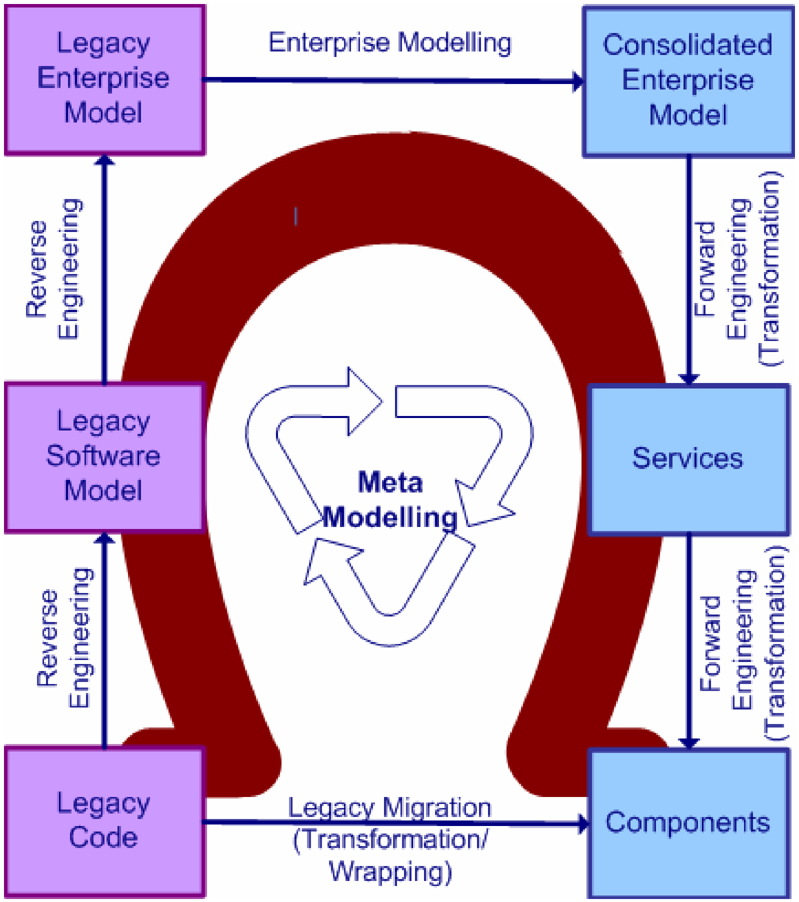
While migration can have significant value, any specific migration requires a concrete analysis of the feasibility, risk and cost involved

The strategic identification and extraction of services from legacy code is crucial as well

Reengineering Processes for Migration to SOA Environments—Current Efforts 1

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 $_2$

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

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 $_3$

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 [Lewis]
- 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 4

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

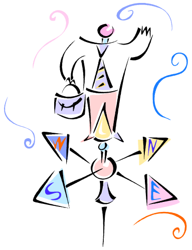
Real challenge is mining legacy code for services that have business value

- 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 [Sneed]
- Application of feature extraction techniques to service identification, given that services usually correspond to features [Sneed]

Conclusions on Key Challenges 1

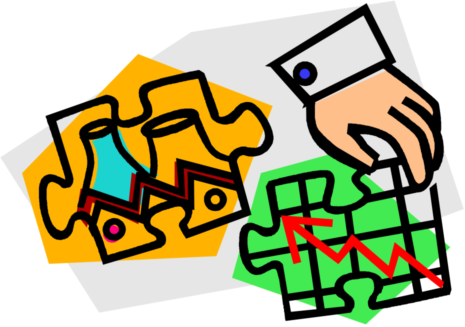
Engineering challenges are significant if SOA is to be used in "advanced ways"

- Semantics
- 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

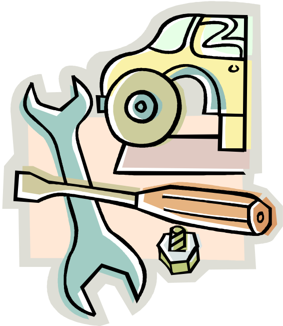


Conclusions on Maintenance and Evolution of Service-Oriented Systems

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

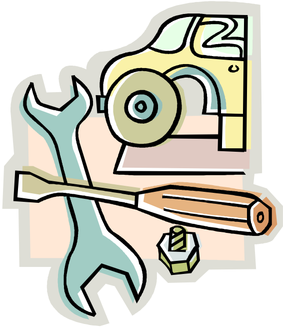


Conclusions on Maintenance and Evolution of Service-Oriented Systems

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



Agenda

Introduction

- SOA Challenges
- Common Misconceptions
- Consequences of Decisions

Introduction to SOA Research Agenda

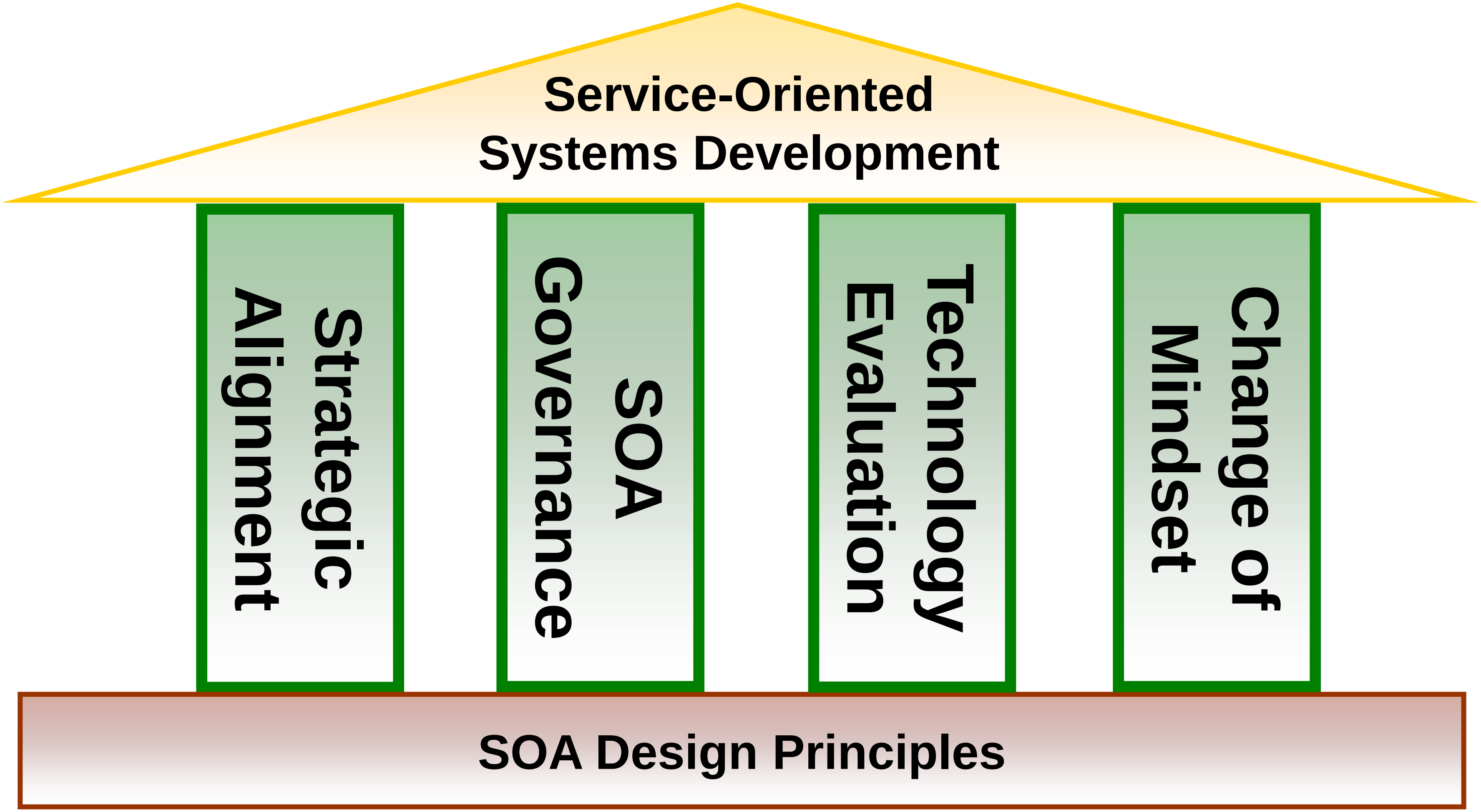
Pillars of Service-Oriented Systems Development ⇦

Challenges of Migration to SOA Environments

SMART (Service Migration and Reuse Technique)

Conclusions

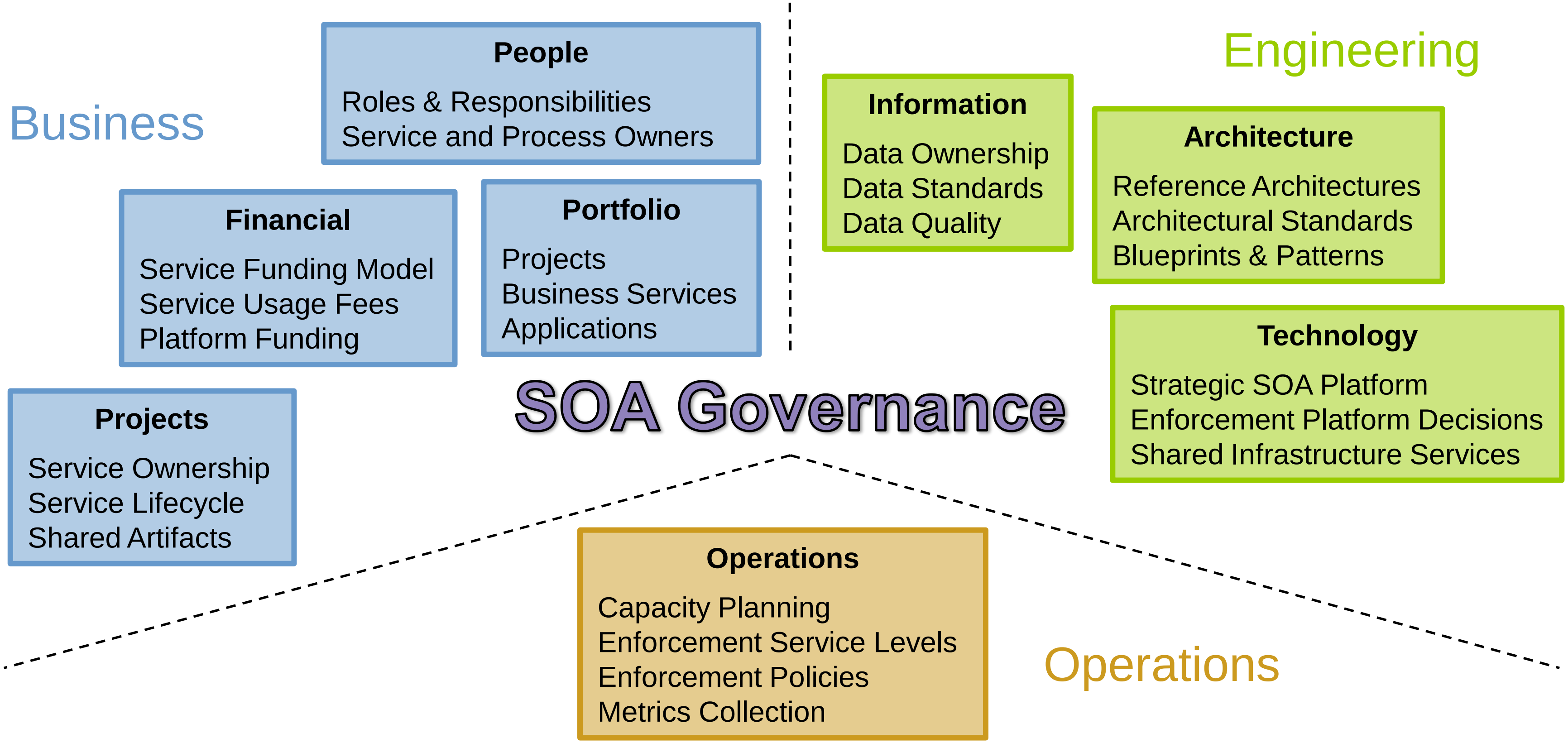
Pillars of Service-Oriented Systems Development



Different Business Needs and Goals Drive Different SOA Strategies

Business Needs and Goals	SOA Strategy
Increase information available to business customers	• Intuitive portals • Creation of services related to customer information
Integrate business partners	• Heterogeneous interoperability • Back office integration • Identification of business rules
Improve business processes	• Identification of key processes • Elimination of redundancy • Consistency between processes • Services that access legacy systems

Examples of Governance Elements



Business

People

Roles & Responsibilities
Service and Process Owners

Financial

Service Funding Model
Service Usage Fees
Platform Funding

Portfolio

Projects
Business Services
Applications

Projects

Service Ownership
Service Lifecycle
Shared Artifacts

Engineering

Information

Data Ownership
Data Standards
Data Quality

Architecture

Reference Architectures
Architectural Standards
Blueprints & Patterns

Technology

Strategic SOA Platform
Enforcement Platform Decisions
Shared Infrastructure Services

SOA Governance

Operations

Capacity Planning
Enforcement Service Levels
Enforcement Policies
Metrics Collection

Operations

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

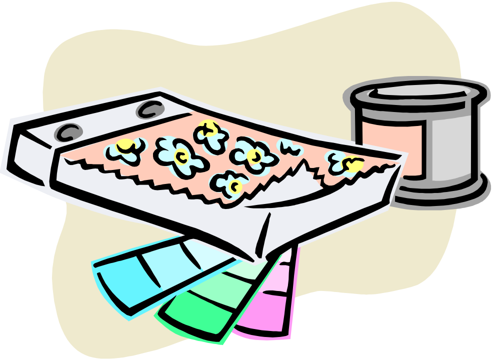
Design-Time Governance

Because of the wide number of potential services, develop decision rules for guiding development of services that

- Are closely aligned with business goals
- Have greatest impact with least risk

Enforce consistency in

- Use of standards
- Access to the infrastructure
- Processes



Manage reuse by enforcing

- Systematic evaluation of migration feasibility
- Consistent approach to legacy component migration

Runtime Governance

Policy enforcement rules relative to

- Execution of services only in ways that are legal
- Security, especially to account for new access points to systems and data
- Replacement of services
- Consistency in interaction with SOA infrastructure

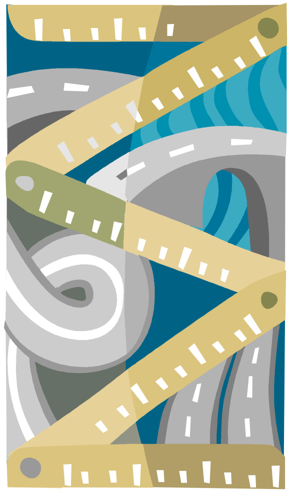
Service level agreements (SLAs)

- Runtime validation of promises made in SLAs
  - Performance, throughput, availability
- Automated metrics, tracking, and reporting
  - Frequency of use of services
  - Identification of exceptions to policies
  - Identification of problem areas
- Problem management

Examples of SOA-Related Metrics

Measurements are used to adjust the SOA strategy

- Effort to develop services
- Effort to reuse services from legacy assets
- Service usage
- Change history
- Policy waiver requests
- Policy violations
- Service performance



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?

T-Check^{SM}

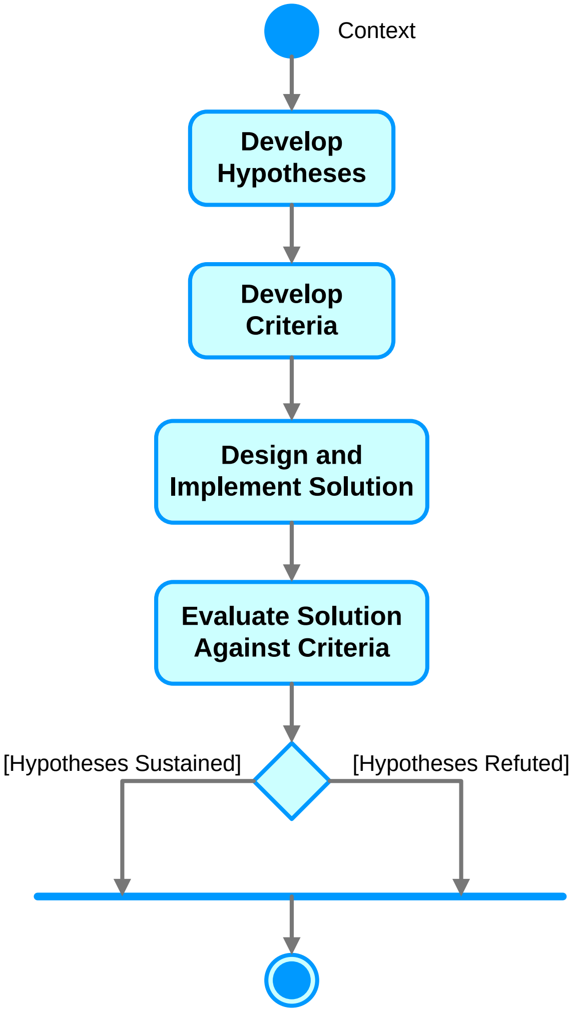
Experiment, situated in a specific context, with the goal of providing a “technology sanity check”

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



Benefits of Contextual Experimentation

Context framing provides for more realistic evaluations

Clear hypothesis and criteria avoid time wasted “playing” with technologies

Simplicity of experiments allows early insight into technologies without a huge investment

Other benefits

- Early competence development of people conducting the experiments
- “Side knowledge”—available support, communities, common problems, adoption risks, etc.

Service-Oriented Systems Require a Different Development Approach

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

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 product vendors work

Some Implications for Architecture and Design Activities

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

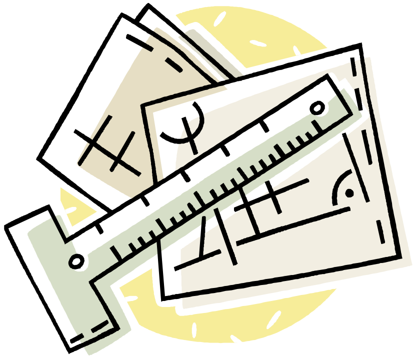
- Security, transaction management, data transformations, etc.

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

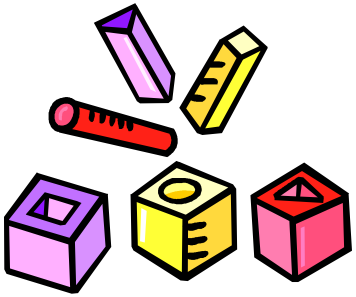
Development environments need to be similar/same as production environments—as in any distributed system environment

- 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

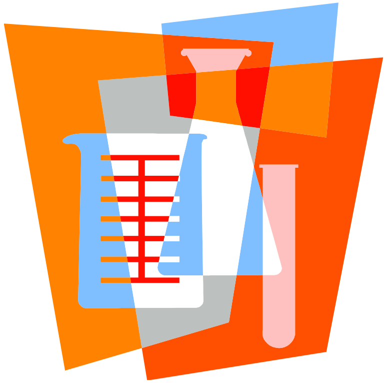
System testing of a service consumer requires all services (or test instances of them) to be available

- 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*)



Agenda

Introduction

- SOA Challenges
- Common Misconceptions
- Consequences of Decisions

Introduction to SOA Research Agenda

Pillars of Service-Oriented Systems Development

Challenges of Migration to SOA Environments

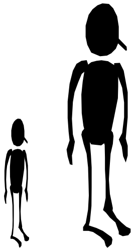
SMART (Service Migration and Reuse Technique)

Conclusions

Reuse Challenges

Reuse at the service level is more complex than reuse at the module or component level.

- From the service provider perspective
  - Designing reusable services requires a different approach, skill set, and mindset
  - Bigger stakeholder community because services are typically reused at organization and sub-organization level
  - Services need to be as generic as possible so that they are of interest to multiple service consumers and at the same time need to add value to potential consumers
- From the service consumer perspective
  - Larger granularity may lead to larger incompatibilities



Legacy System Challenges

It may not always be possible to reuse functionality of legacy systems by exposing them as services.

- Technical constraints due to the nature of the legacy system
  - A batch system needs to be exposed as a service for an interactive online Web application.
- Immature technology or lack of technology for a particular legacy environment

Cost of exposing a legacy system as services may be higher than replacing it with a new service-oriented system.

Examples of Challenging Legacy System Characteristics

Poor separation of concerns

- User interface code tightly coupled with business function code

Tool availability

- Target is Web Services; XML and SOAP libraries are not available for all legacy platforms.

Architectural mismatch

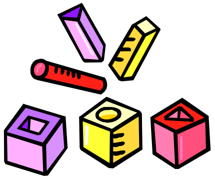
- The asynchronous call to the service might be in conflict with legacy system synchronous behavior.

Operational mismatch

- The legacy system is batch-oriented, the service user expects an immediate response.

Dependencies on commercial products

- Licensing issues?



Addressing Legacy System Challenges

Identify relevant and non-relevant legacy components.

- Not all legacy components can be meaningfully reused as services—from a strategic and a technical perspective.

Make decisions based on “hands-on,” contextual analysis.

- System-specific analysis is important because every system is unique.
- Previous analysis and results can be used a guidelines.

Estimate cost, risk, and confidence of estimates of changes required to each legacy component.

Migration to SOA Environments: A Potentially Complex Engineering Task

The characteristics of SOA enable the exposure of legacy system functionality as services.

- Presumably without making significant changes to the legacy systems

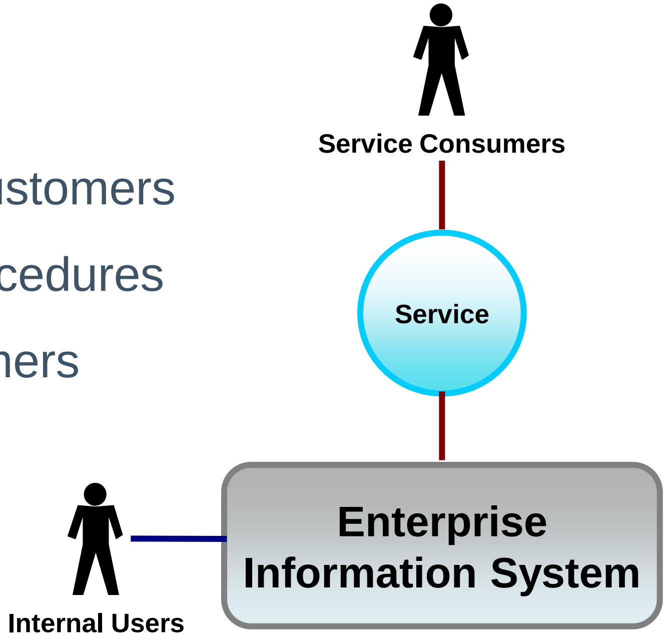
The complexity of the migration will largely depend on the characteristics of the SOA environment—some examples:

- User community
- SOA infrastructure technology
- SOA strategy
- Operations

Operations

A stand-alone system can become a component of a system of systems by exposing services.

- Startup procedures
- Policies for communication of changes and updates to internal and service consumers
- Potential for
  - Conflicting requirements—two sets of customers
  - More complex change management procedures
  - Performance degradation—more customers



Agenda

Introduction

- SOA Challenges
- Common Misconceptions
- Consequences of Decisions

Introduction to SOA Research Agenda

Pillars of Service-Oriented Systems Development

Challenges of Migration to SOA Environments

SMART (Service Migration and Reuse Technique) ⇦

Conclusions

SMART Goals

SMART analyzes the viability of reusing legacy components as the basis for services by answering these questions:

- Does it make sense to migrate the legacy system to services?
- What services make sense to develop?
- What components can be used to implement these services?
- What changes are needed to accomplish the migration?
- What migration strategies are most appropriate?
- What are the preliminary estimates of cost and risk?

The SMART Family

Migration Pilot

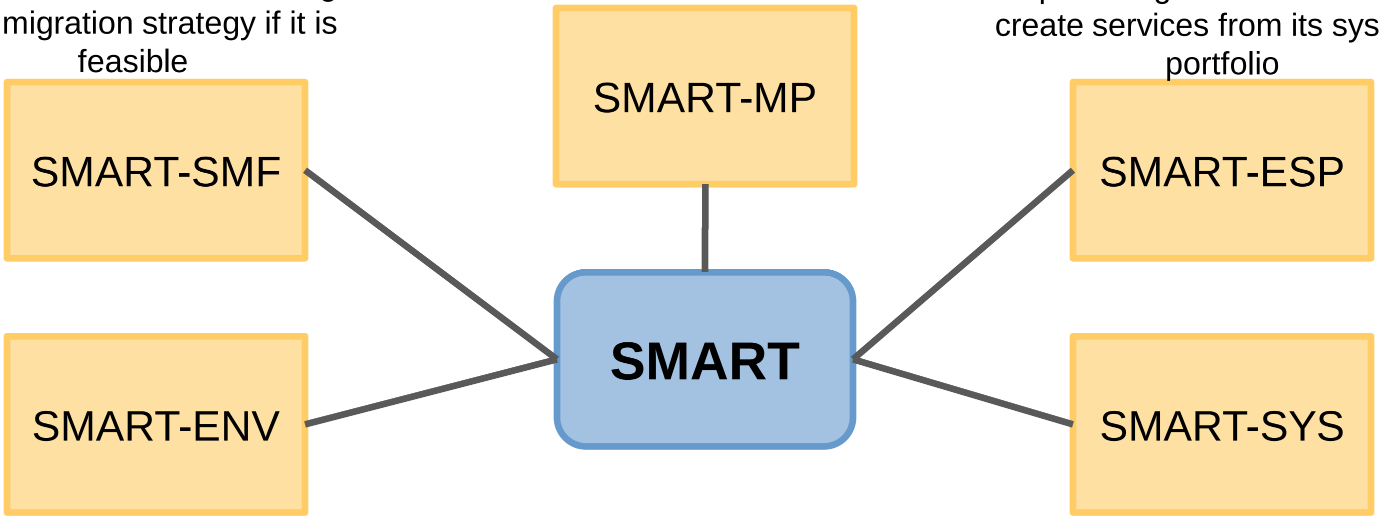
Helps an organization select a pilot project that includes a migration strategy with understanding of costs and risks involved

Service Migration Feasibility

Helps an organization establish the feasibility of migration to an SOA environment and creates a high-level migration strategy if it is feasible

Enterprise Service Portfolio

Helps an organization select and create services from its systems portfolio



SMART-MP

SMART-SMF

SMART-ESP

SMART

SMART-ENV

SMART-SYS

SOA Environment

Helps an organization understand a target SOA environment in detail, including associated costs and risks of migrating to that environment

SOA-Based Systems Development

Helps an organization understand a complete SOA-based system—services, consumers, environment—including risk and cost data

Why a SMART Family? 1

The pre-requisite of the current SMART is the identification of a target SOA environment

Reality is that

- Many organizations are at earlier stages in the SOA adoption process
- There are multiple entry points to SOA adoption

We have begun to identify variations on the SMART process to deal with these differences

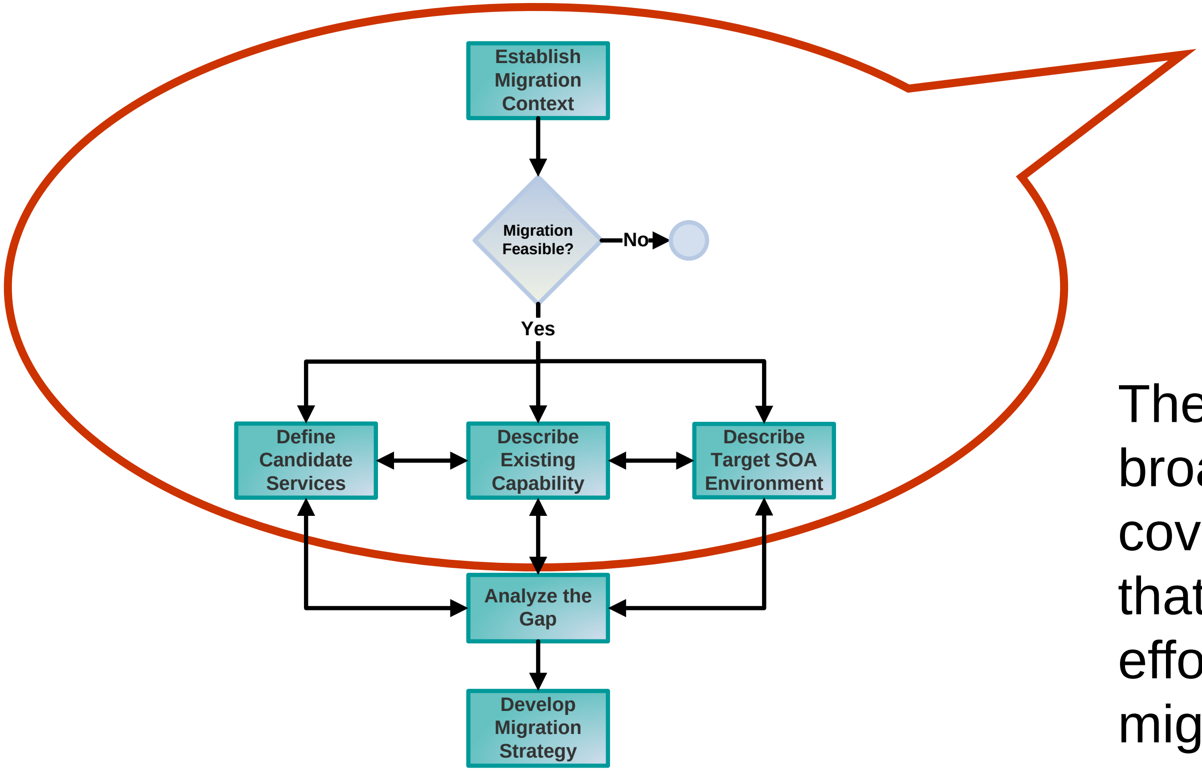
The members of the SMART Family follow the same process described earlier, but the emphasis is on certain activities in the process where the SMIG has been enhanced to go into more detail in specific areas

Four Elements of SMART

Process	Service Migration Interview Guide (SMIG)	SMART Tool	Artifacts
Gathers information about • Goals and expectations of migration effort • Candidate services • Legacy components • Target SOA environment Analyzes gap between legacy and target state	Guides discussions in initial SMART activities	Automates data collection Identifies potential risks from data base	• Stakeholder List • Characteristics List • Migration Issues List • Business Process-Service Mapping • Service Table • Component Table • Notional SOA-Based System Architecture • Service-Component Alternatives • Migration Strategy

Service Migration Interview Guide (SMIG)

62 categories of questions that gather information about the migration context, the legacy components, the candidate services, and the target SOA environment


Establish Migration Context
Migration Feasible?
No
Yes
Define Candidate Services
Describe Existing Capability
Describe Target SOA Environment
Analyze the Gap
Develop Migration Strategy

Guides information gathering for the first set of activities

The goal is to assure broad and consistent coverage of the factors that influence the cost, effort, and risk in migration to services.

SMART Tool

Supports information gathering and analysis activities of SMART

- SMIG is implemented as a data model that maps questions to answers to risks to mitigation strategies

Produces draft migration strategy and migration issues list

Consolidates data from a single engagement for information sharing and analysis

Consolidates data from multiple engagements for trend analysis

SMART Tool Components

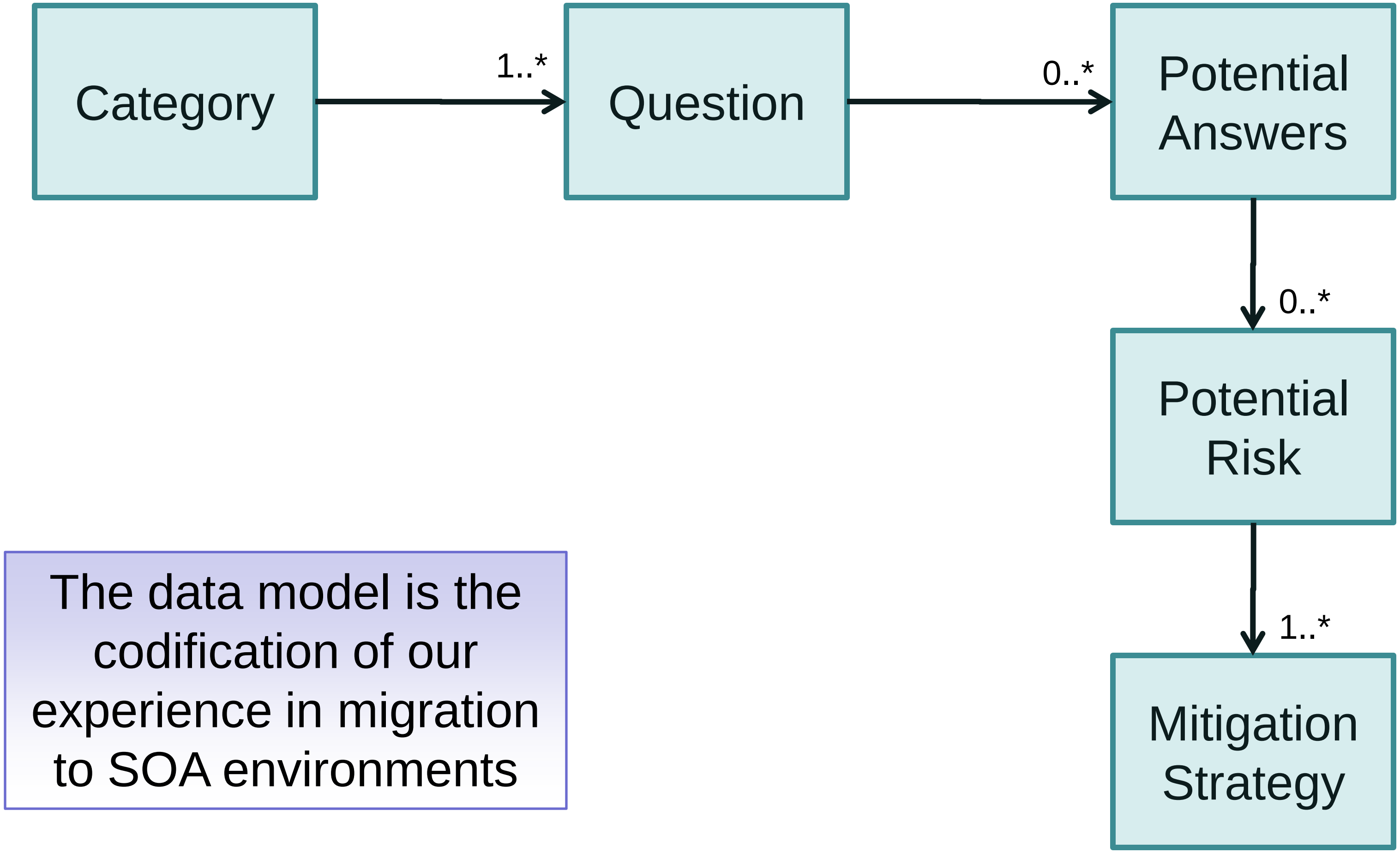
SMART Client

- Java application built using Eclipse RCP
- Runs in offline mode during an engagement
- Uploads data to the SMART Server for consolidation
- Reporting capability

SMART Server

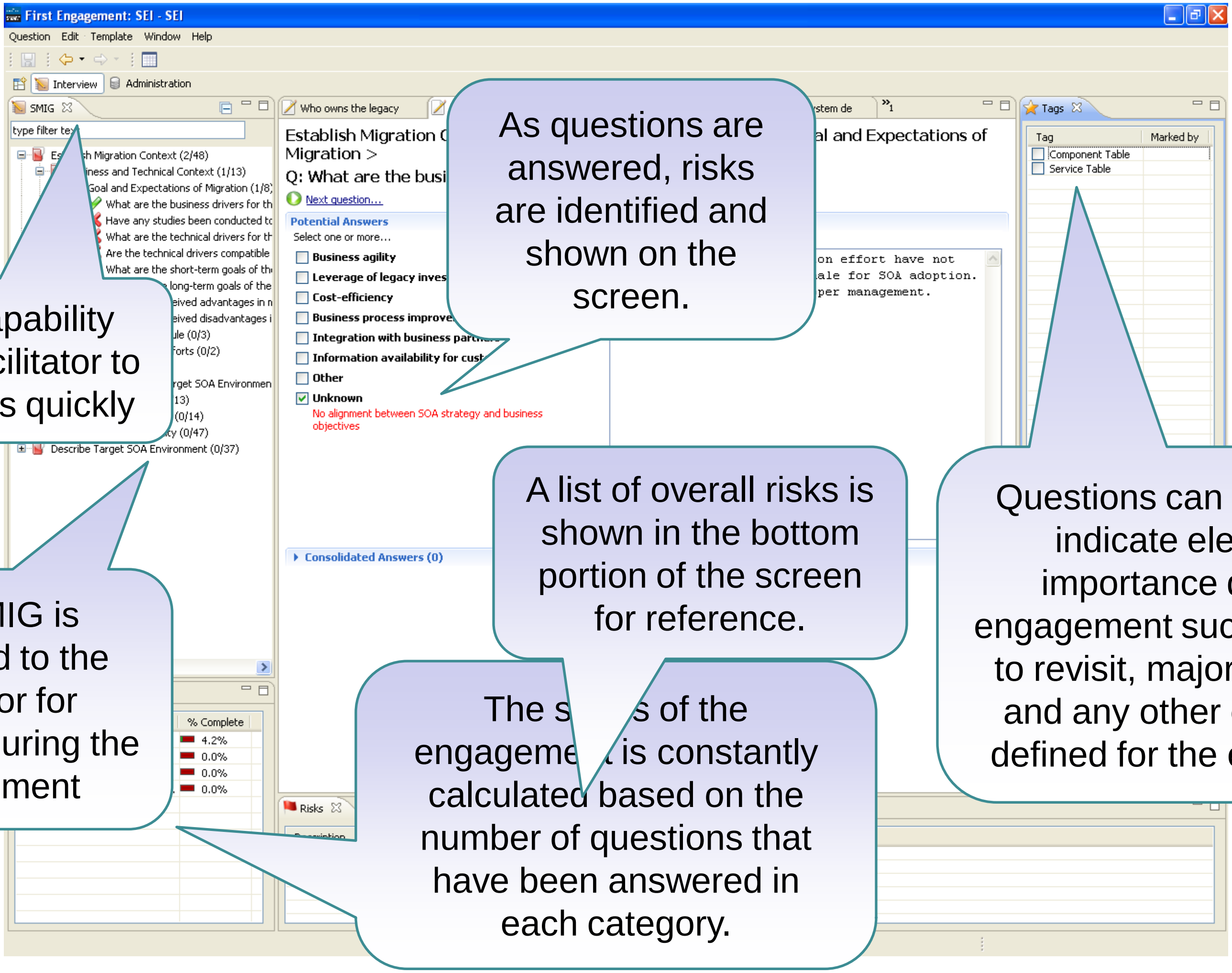
- Web application with an underlying MySQL database
- Runs on an organization’s server
- Enables SMIG maintenance, engagement setup, user maintenance, export/import SMIG, reports

SMIG Data Model



The data model is the codification of our experience in migration to SOA environments

SMART Client – Interview Perspective



First Engagement: SEI - SEI

Question Edit Template Window Help

Interview Administration

SMIG

type filter te

Es   sh Migration Context (2/48)
   iness and Technical Context (1/13)
   Goal and Expectations of Migration (1/8)
   What are the business drivers for th
   Have any studies been conducted to
   What are the technical drivers for th
   Are the technical drivers compatible
   What are the short-term goals of th
   long-term goals of the
   eived advantages in n
   eived disadvantages i
   ule (0/3)
   forts (0/2)
   rget SOA Environmen
   13)
   (0/14)
   ty (0/47)
Describe Target SOA Environment (0/37)

% Complete
4.2%
0.0%
0.0%
0.0%

Who owns the legacy

Establish Migration C   al and Expectations of
Migration >

Q: What are the busi

Next question...

Potential Answers

Select one or more...

- □ Business agility
- □ Leverage of legacy inves
- □ Cost-efficiency
- □ Business process improve
- □ Integration with business part
- □ Information availability for cust
- □ Other
- ☑ Unknown

No alignment between SOA strategy and business objectives

```
on effort have not
ale for SOA adoption.
per management.
```

Consolidated Answers (0)

Risks

Tags

Tag	Marked by
□ Component Table	
□ Service Table	

A search capability allows the facilitator to find questions quickly

As questions are answered, risks are identified and shown on the screen.

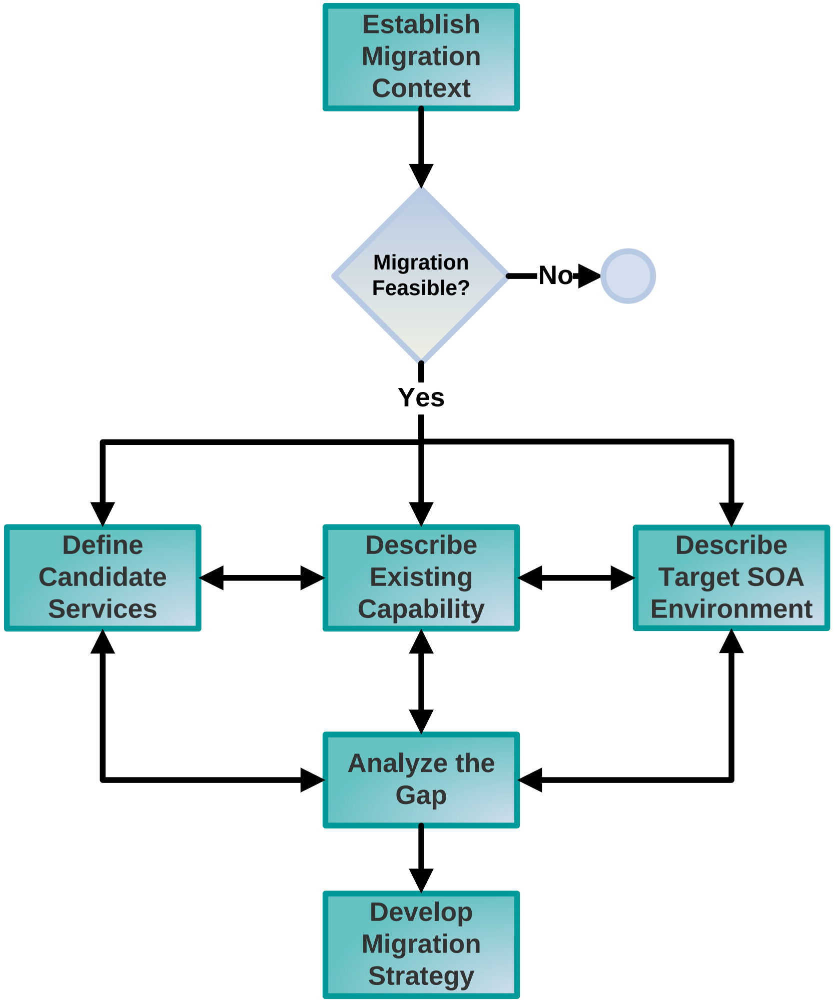
The SMIG is presented to the facilitator for reference during the engagement

A list of overall risks is shown in the bottom portion of the screen for reference.

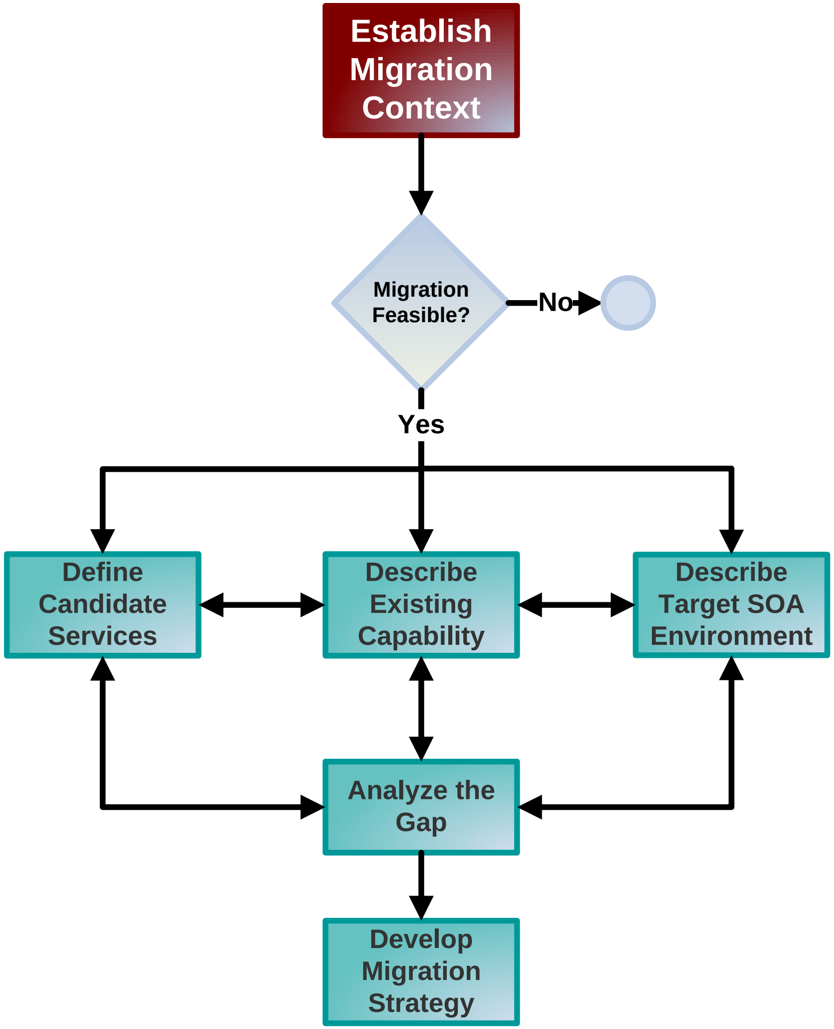
Questions can be tagged to indicate elements of importance during the engagement such as the need to revisit, major area of risk, and any other custom tags defined for the engagement.

The s   s of the engageme t is constantly calculated based on the number of questions that have been answered in each category.

SMART Process Activities


Establish Migration Context
Migration Feasible?
No
Yes
Define Candidate Services
Describe Existing Capability
Describe Target SOA Environment
Analyze the Gap
Develop Migration Strategy

Establish Migration Context


Establish Migration Context
Migration Feasible?
No
Yes
Define Candidate Services
Describe Existing Capability
Describe Target SOA Environment
Analyze the Gap
Develop Migration Strategy

Understand the business and technical context for migration

Identify stakeholders

Understand legacy system and target SOA environment at a high level

Identify a set of candidate services for migration

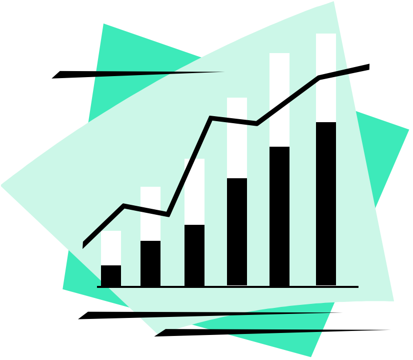
Understand Business and Technical Context

Understand rationale, goals, and expectations for migration to an SOA environment

Understand technical and business drivers

Understand project constraints (e.g. schedule, budget)

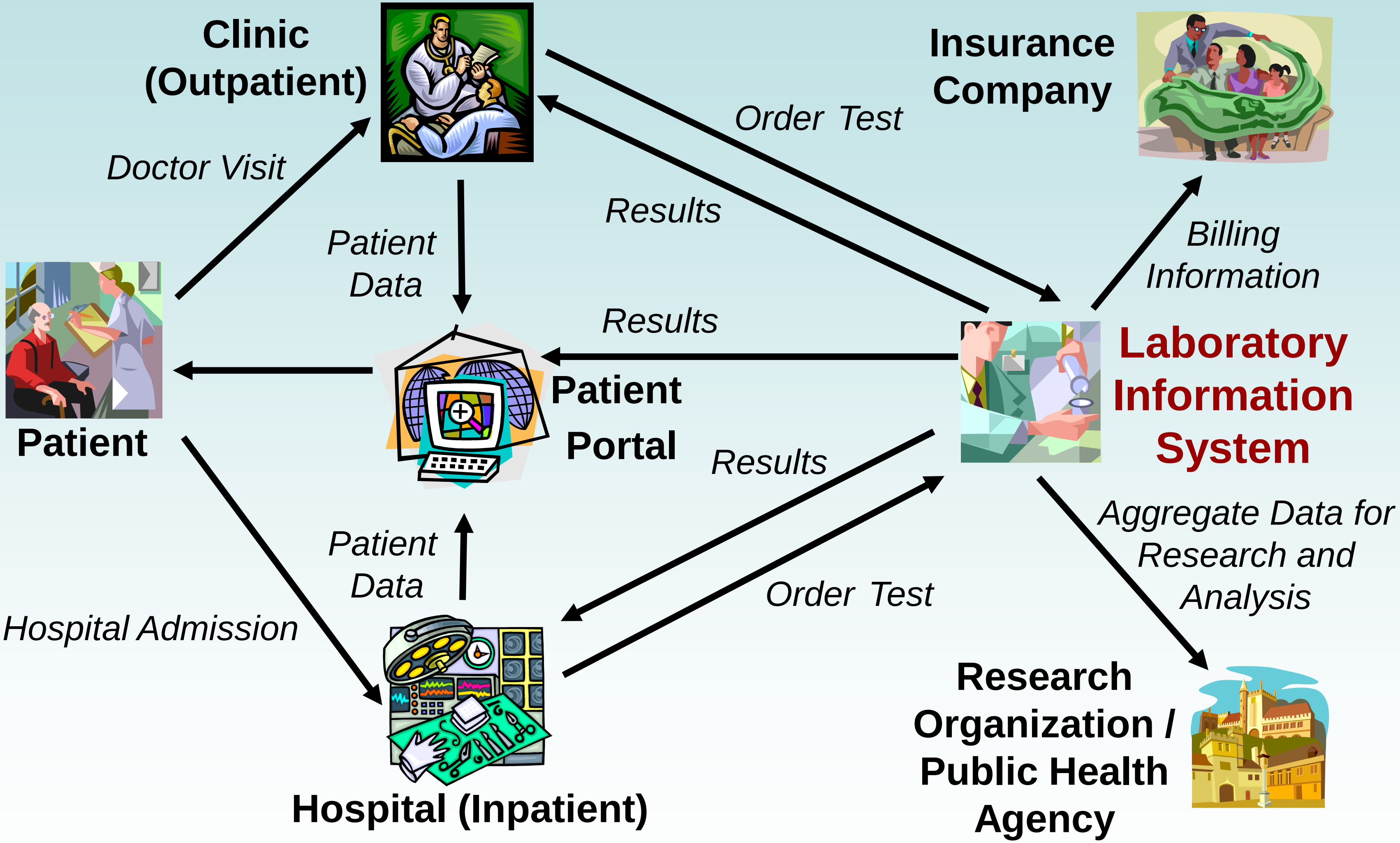
Gain knowledge about previous related efforts or analyses



Establish Migration Context: SMIG Examples

Discussion Topic	Related Questions	Potential Migration Issues
Goal and Expectations of Migration Effort	• What are the business and technical drivers for the migration effort? • What are the short-term and long-term goals?	• No SOA strategy • Goals for migration are not clear.
High-Level Understanding of Legacy System	• What is the main functionality provided by the legacy system? • What is the high-level architecture of the system? • What is the current user interface to the system?	• Legacy system knowledge is not available. • Architectural mismatch • User interface complexity hard to replicate in service consumers
High-Level Understanding of Target SOA Environment	• What are the main components in the target SOA environment? • Is this the organization’s first attempt to deploy services in this environment?	• Target SOA environment has not been identified. • No in-house knowledge of target SOA environment
Potential Service Consumers	• Who are the potential service consumers?	• Consumers for services have not been identified.

Example Context: Laboratory Information System (LIS)



Clinic (Outpatient)

Insurance Company

Order Test

Doctor Visit

Results

Patient Data

Billing Information

Results

Patient

Patient Portal

Laboratory Information System

Results

Aggregate Data for Research and Analysis

Patient Data

Order Test

Hospital Admission

Research Organization / Public Health Agency

Hospital (Inpatient)

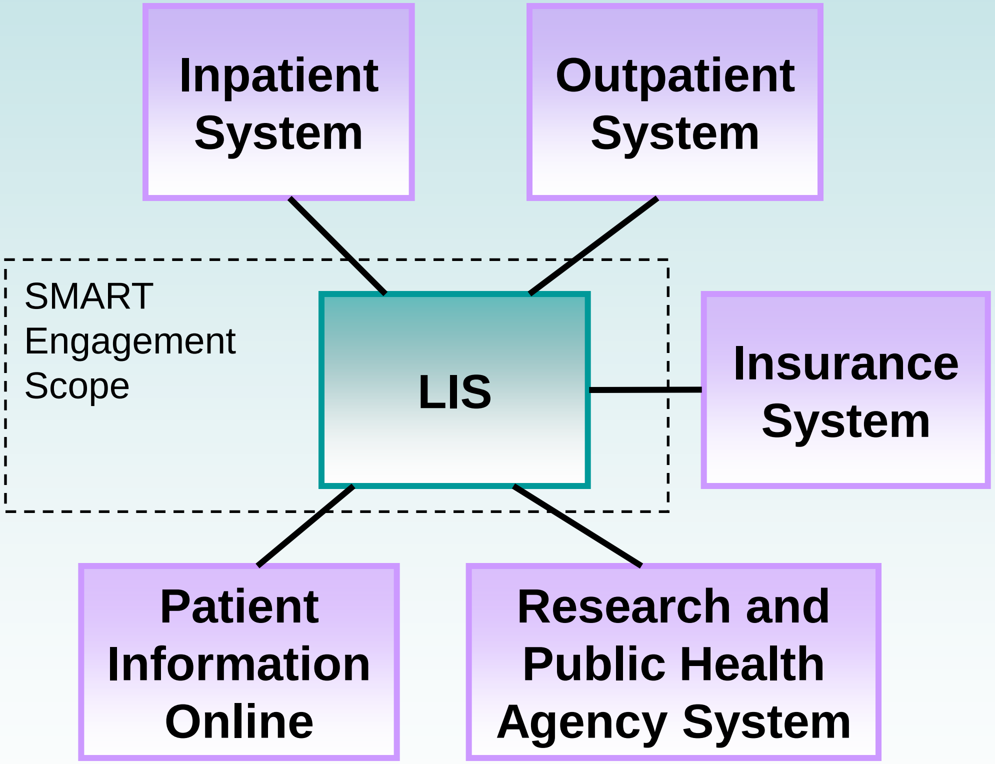
Example Context: LIS Context Diagram

Lab information shared between many systems

Need to move to a SOA environment to increase reusability of common lab tasks

Key questions:

1. Which services should be created?
2. In what order?
3. Should some legacy components be replaced with new components?



LIS: Drivers for Legacy Migration

Improve patient care by

- Providing access to lab information from any clinical system in real time (current access is mostly batch-oriented)
- Making lab information accessible to patients via the Internet using a patient portal

Reduce IT costs by

- Creating common and reusable services
- Reducing the number of different interaction points (interfaces)
- Lowering maintenance and upgrade costs

LIS: Legacy System at a High Level

Laboratory Information System (LIS)

- 800,000 lines of code
- Six major modules—~2500 C++ classes and ~1500 Java classes
  - Lab Test Catalog module is written in Java but is actually a wrapper to a legacy COBOL system
- Some components run on Windows operating system and some on Linux OS

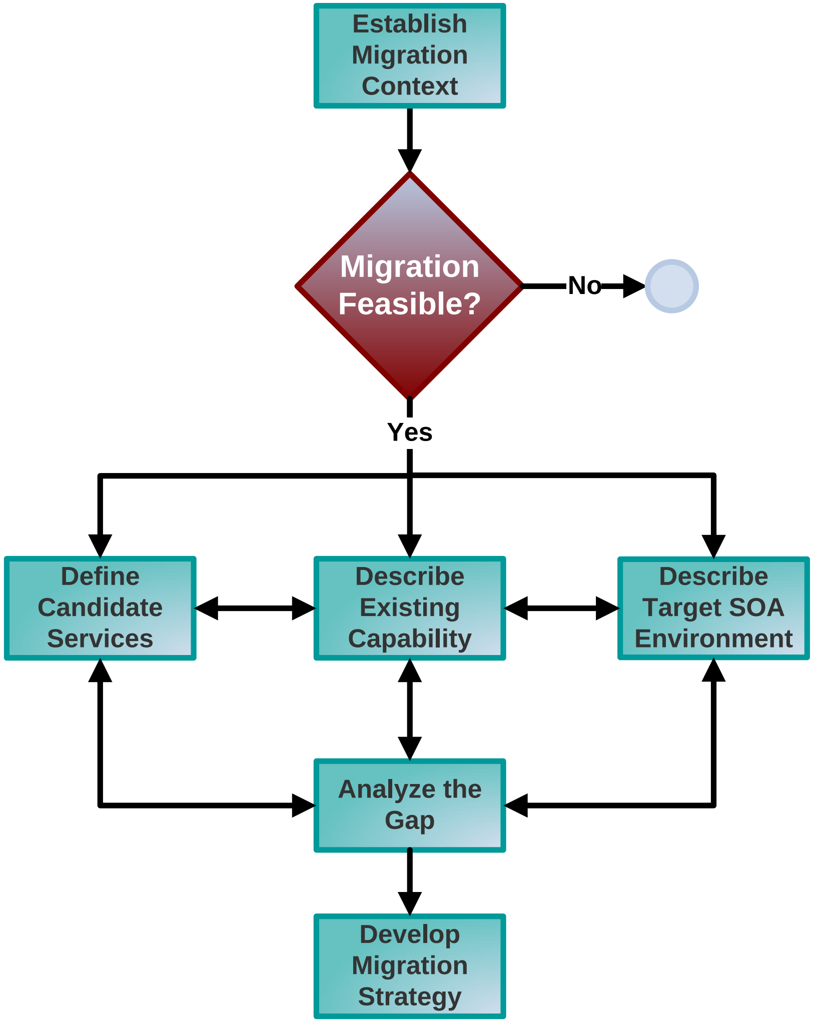
Interaction with external systems is point-to-point through dedicated sockets

- Some data transfers are done in batch mode overnight (i.e., lab results)
- Not all exchanged information uses the same version of HL7 (V3 vs. V2.X)

Dependencies on several commercial products

- Oracle Database
- Weblogic Application Server

Checkpoint for Migration Feasibility

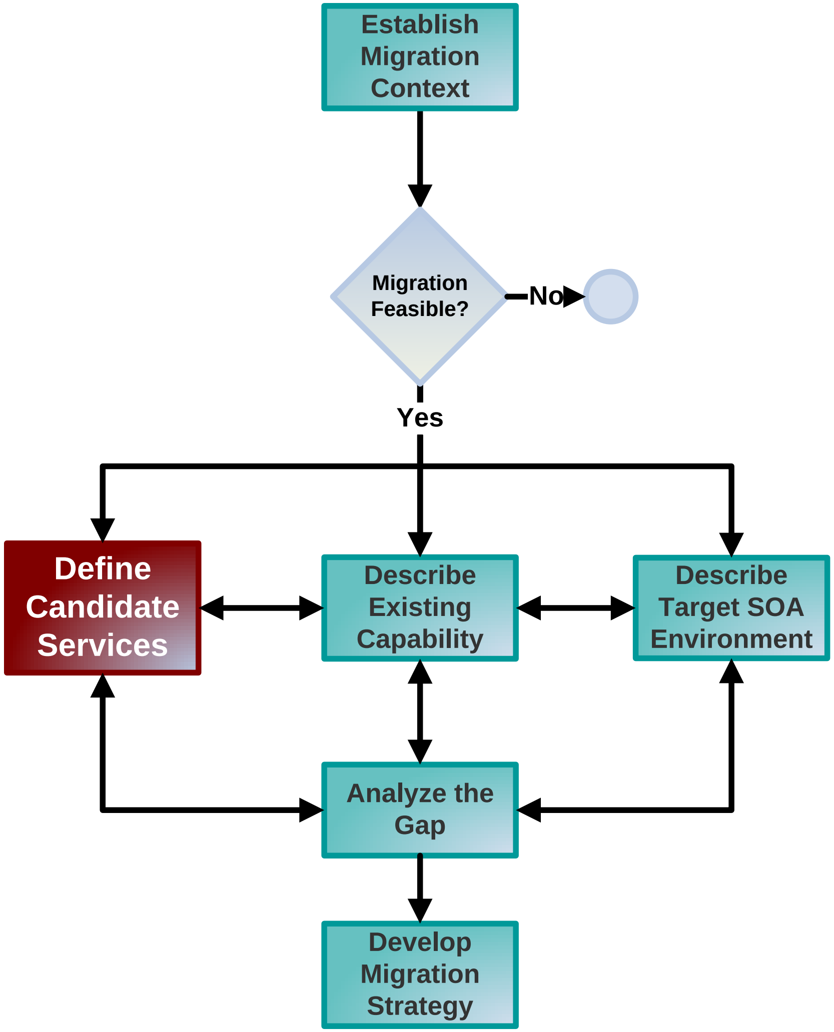

Establish Migration Context
Migration Feasible?
No
Yes
Define Candidate Services
Describe Existing Capability
Describe Target SOA Environment
Analyze the Gap
Develop Migration Strategy

Decision to continue with the process has to be made.

Potential outcomes at this point are

- The migration is initially feasible.
- The migration has potential but requires additional information to make an informed decision.
- The migration is not feasible.

Define Candidate Services


Establish Migration Context
Migration Feasible?
No
Yes
Define Candidate Services
Describe Existing Capability
Describe Target SOA Environment
Analyze the Gap
Develop Migration Strategy

Select a small number of services, usually 3-4, from the initial list of candidate services

For these candidate services, the end goal is to fully specify inputs and outputs

Initial Business Process-Service Mapping

Business Process	Candidate Services
Search Lab Test Catalog	Get Test Catalog, Get Test Details
Order/Re-order Test	Get Test Catalog, Get Patient Information, Get Test Details, Create Lab Test Order
Track Status of Tests	Get Patient Information, Get Test Details
Provide Billing Information	Get Patient Information, Get Test Details
Review and Report Test Results	Get Patient Information, Get Test Details, Get Test Results
Analysis and Mining for Trends	Get Test Details, Get Aggregate Test Results

NOTE: This table was created during *Establish Migration Context*

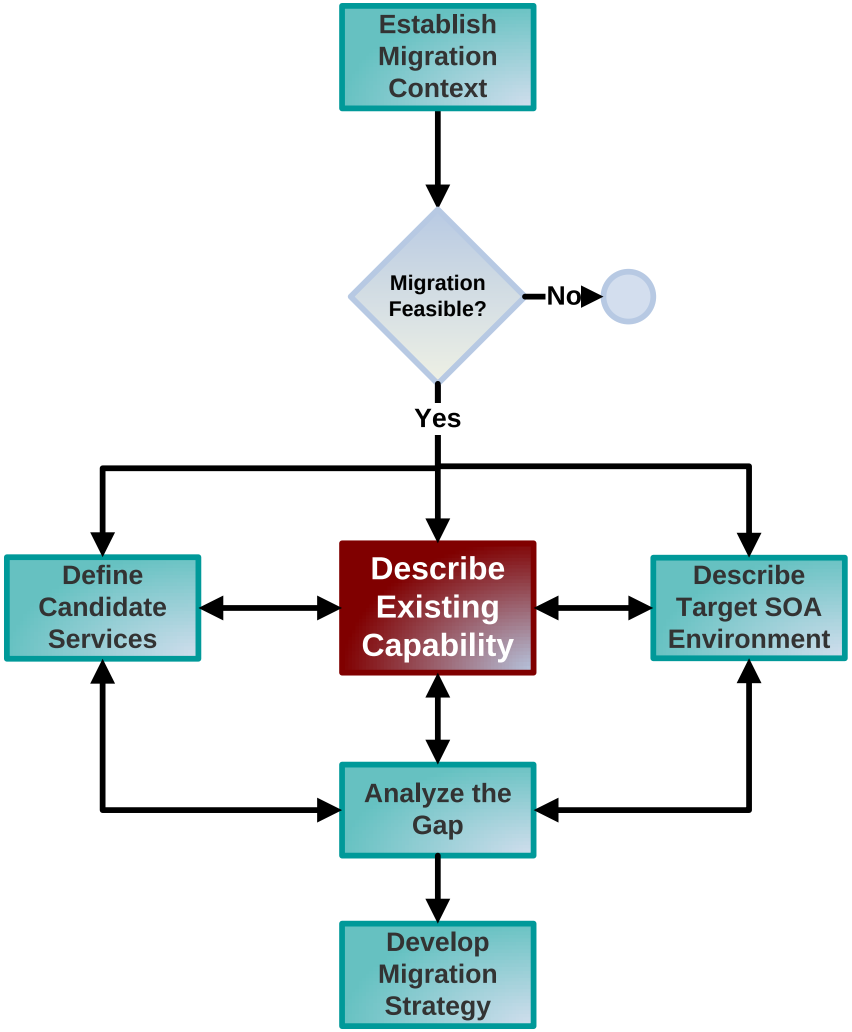
Initial Service Table

Service	Description	Type	Potential Service Consumers
Get Test Results	Obtains detailed test results either for one patient or for all the patients for which tests were completed on a day for a particular location	Business	EMR Systems
Get Test Catalog	Obtains the catalog of tests provided by the clinical lab	Business	EMR systems
Data Format Service	Formats message according to a given version of HL7	Infrastructure	Internal services and applications
…	…	…	…

Service	Inputs	Outputs	Key Quality Attribute Requirements	…
Get Test Results	Patient ID (s) Test ID Location ID Date	Test Result Details	Security	
Get Test Catalog	Test type(s)	Test catalog	Configurability	
Data Format Service	Data, HL7 version	HL7-Formatted Data	Interoperability	
…	…	…	…	…

NOTE: By the end of this iterative process, inputs and outputs should include data types.

Describe Existing Capability



Establish Migration Context
Migration Feasible?
No
Yes
Define Candidate Services
Describe Existing Capability
Describe Target SOA Environment
Analyze the Gap
Develop Migration Strategy

Obtain descriptive data about legacy components

- Name, function, size, language, operating platform, age of legacy components, etc.

Question technical personnel about

- Architecture and design paradigms
- Complexity, coupling, interfaces
- Quality of documentation
- Component/product dependencies

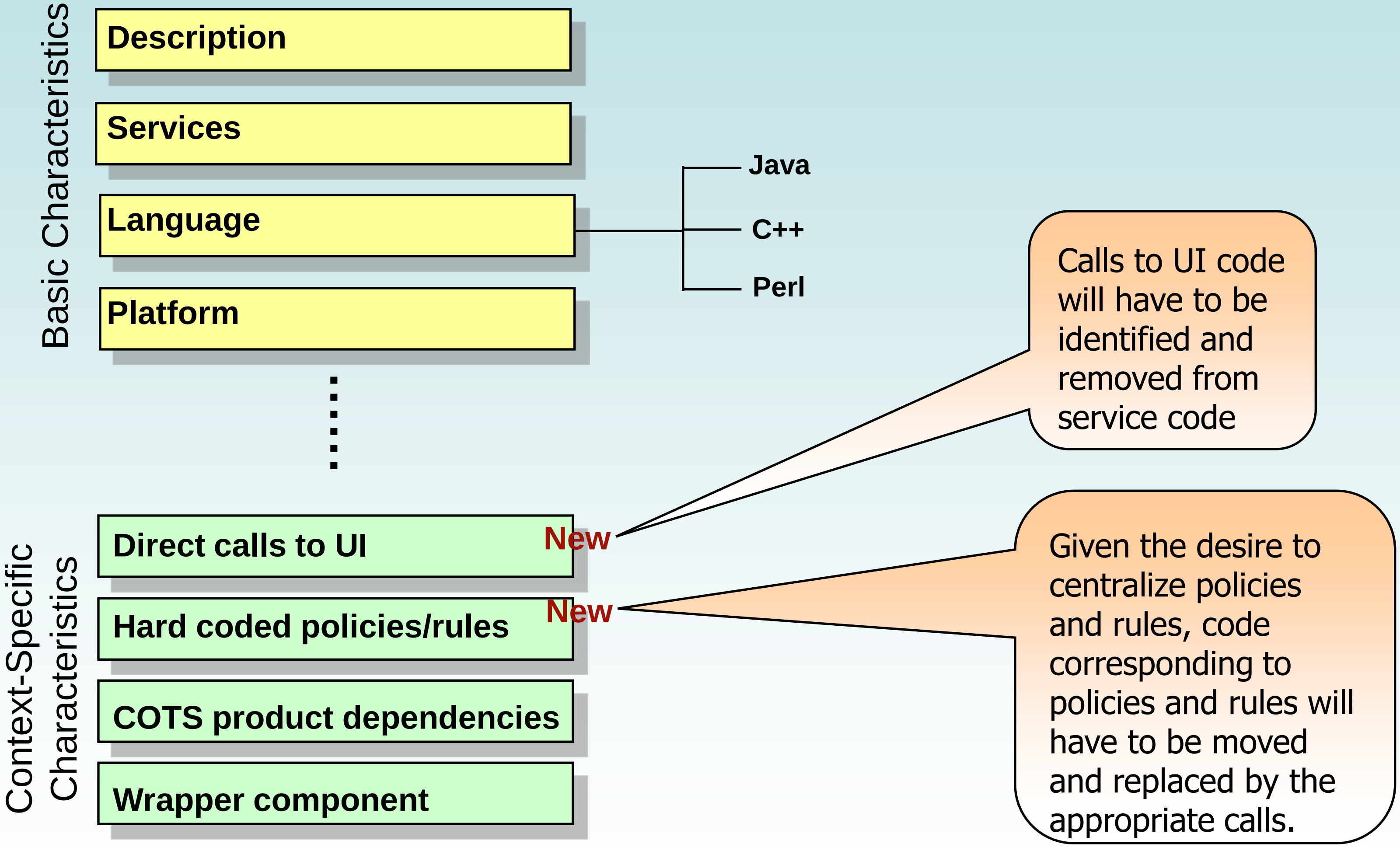
Gather data about

- Quality, maturity, existing problems
- Change history
- User satisfaction

Describe Existing Capability: SMIG Examples

Discussion Topic	Related Questions	Potential Migration Issues
Legacy System Characteristics	• What is the history of the system? • Is the system a proof of concept, prototype, under development, in testing, or a fielded system? • What system documentation is available? • Does the system have interfaces to other systems? • What are potential locking, persistence, or transaction problems if accessed by multiple users when migrated to services?	• Planned development concurrent with service migration • Limited system documentation • Interfaces to other systems will open doors to service consumers. • Single-user system may have problems in a multi-user environment.
Legacy System Architecture	• What architecture views are available? • What are the major modules of the system and dependencies between modules? • Is user interface code separate from the business logic code? • Are there any design paradigms or patterns implemented in the system? • What are the key quality attributes built into the current architecture of the system?	• Lack of architecture documentation may lead to underestimation of complexity. • Tight coupling between user interface code and business logic code increases effort. • Undocumented violations of design patterns may cause problems. • Key quality attributes may not hold true in a services environment.
Code Characteristics	• What code documentation is available? • What coding standards are followed?	• Poor coding practices will increase migration effort.

LIS: Updated Characteristics List



Basic Characteristics

Description

Services

Language

- Java
- C++
- Perl

Platform

Context-Specific Characteristics

Direct calls to UI New

Hard coded policies/rules New

COTS product dependencies

Wrapper component

Calls to UI code will have to identified and removed from service code

Given the desire to centralize policies and rules, code corresponding to policies and rules will have to be moved and replaced by the appropriate calls.

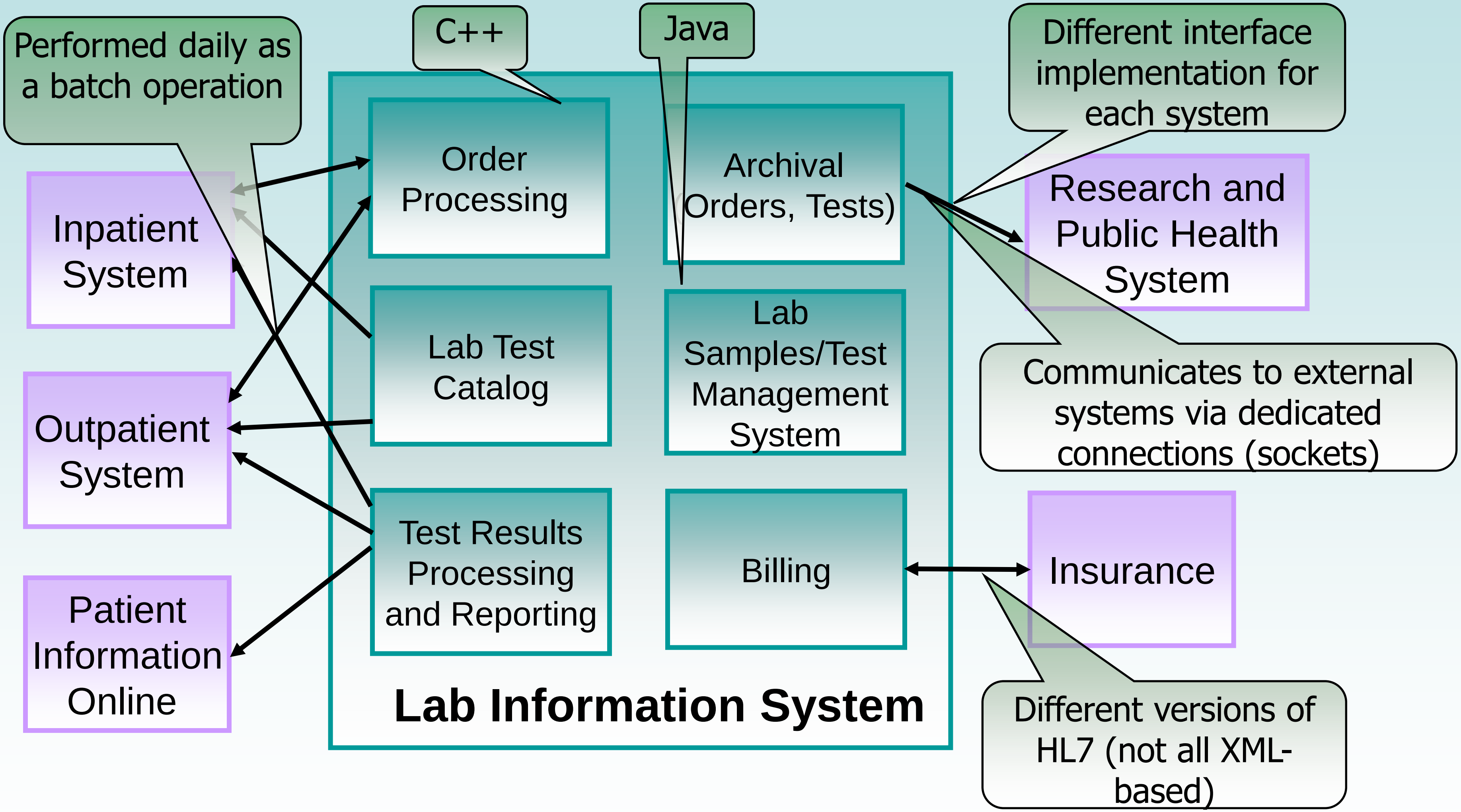
LIS: Component Table

Component	Description	Services	Language	Platform	Size (SLOC)
LabTestCatalog	Manages the catalog of all available lab tests	Get Test Catalog, Get Test Details, Create Lab Test Order	Java	Linux	1,000
ResultsProcessor	Contains business rules for processing test results and providing processed results to external systems	Get Test Results, Get Aggregate Test Results	C++, Java, Perl	Unix, Windows XP	8,000

Component	Complexity	Version	Level of Documentation	Last Release Date
LabTestCatalog	Medium	5.6	High	02/10/2005
ResultsProcessor	Very High	8.2	Medium	06/01/2005

Component	Wrapper Component	COTS Product Dependencies	Direct Calls to UI	Hard-coded Policies and Rules
LabTestCatalog	Yes	3rd party libraries, HL7 v2.3	No	No
ResultsProcessor	No	Oracle database, Weblogic Application Server	Yes	Yes

LIS: Module View



Performed daily as a batch operation

C++

Java

Different interface implementation for each system

Inpatient System

Order Processing

Archival (Orders, Tests)

Research and Public Health System

Lab Test Catalog

Lab Samples/Test Management System

Communicates to external systems via dedicated connections (sockets)

Outpatient System

Test Results Processing and Reporting

Billing

Insurance

Patient Information Online

Lab Information System

Different versions of HL7 (not all XML-based)

LIS: Additional Migration Issues

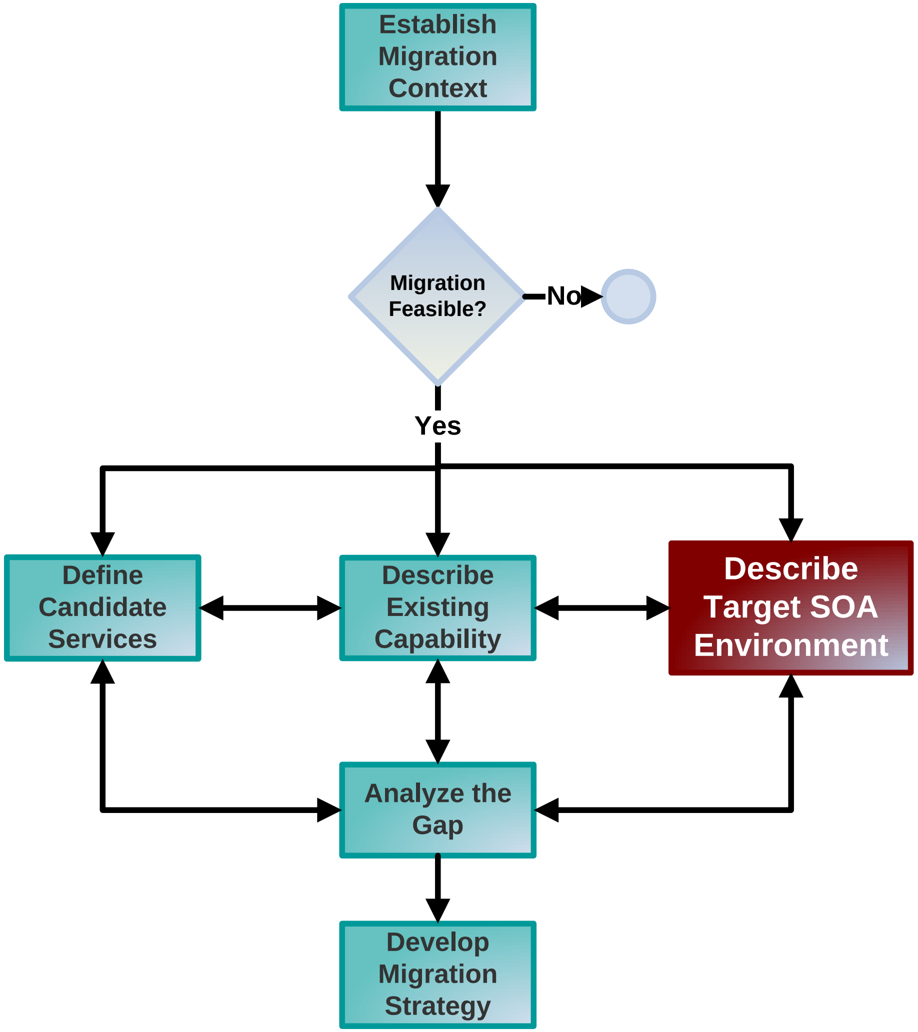
Description: All service consumers do not plan to move to the XML-compliant version of HL7.

Description: Some legacy components are designed only for batch operations. . . .

Description: Some legacy components have direct calls to UI embedded in the core business logic of the code.
Type: Technical **Impact:** Medium New

Description: Different data filtering policies are applied to the same data depending on the interacting external system.
Type: Business, Policy **Impact:** High New

Describe Target SOA Environment

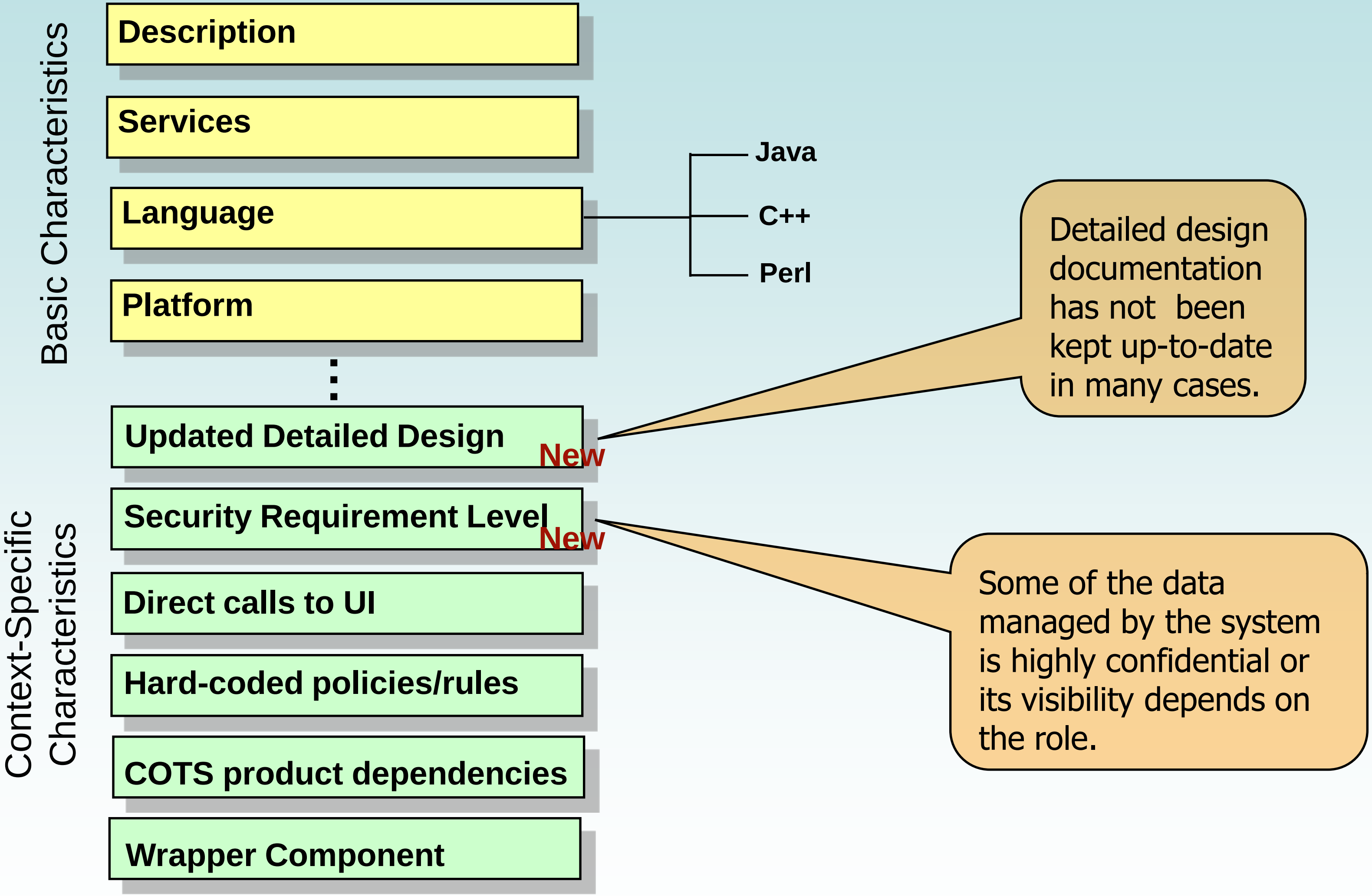

Establish Migration Context
Migration Feasible?
No
Yes
Define Candidate Services
Describe Existing Capability
Describe Target SOA Environment
Analyze the Gap
Develop Migration Strategy

- Identify the impact of specific technologies, standards, and guidelines for service implementation
- Determine state of target SOA environment
- Identify how services would interact with the SOA environment
- Determine QoS expectations and execution environment for services

Describe Target SOA Environment: SMIG Examples

Discussion Topic	Related Questions	Potential Migration Issues
SOA Environment Characteristics	• What is the status of the target SOA environment? • What are the major components of the SOA infrastructure? • Does the target SOA environment provide infrastructure services (i.e., communication, discovery, security, data storage)? • What is the communication model? • What constraints does the target SOA environment impose on services? • Does the legacy system have any behavior that would be incompatible with the target SOA environment? • Once developed, where will services execute?	• Target SOA environment undefined • Redundancy/conflicts between infrastructure services and legacy code • Lack of tools to support legacy code migration to target infrastructure • Compliance with constraints requires major effort. • Architectural mismatch • No thought given to service deployment and execution
Support	• Do you have to provide automated test scripts for the services and make them publicly available? • How will service consumers report problems and provide feedback? • How will service consumers be informed of potential changes in service interfaces and downtime due to upgrades or problems?	• Underestimation of effort to provide service consumer support • Lack of awareness of support requirements

LIS: Updated Characteristics List



Basic Characteristics

Description

Services

Language

Java

C++

Perl

Platform

⋮

Context-Specific Characteristics

Updated Detailed Design New

Security Requirement Level New

Direct calls to UI

Hard-coded policies/rules

COTS product dependencies

Wrapper Component

Detailed design documentation has not been kept up-to-date in many cases.

Some of the data managed by the system is highly confidential or its visibility depends on the role.

LIS: Updated Component Table

Component	Description	Services	Language	Platform	Size (SLOC)	Complexity
LabTestCatalog	Manages the catalog of all available lab tests	Get Test Catalog, Create Order	Java	Linux	1,000	Medium
ResultsProcessor	Contains business rules for processing test results and providing processed results to external systems	Get Test Results, Get Aggregate Test Results	Java, C++, Perl	Unix, Windows XP	8,000	Very High

Component	Version	Level of Documentation	Last Release Date	Wrapper Component	COTS Product Dependencies
LabTestCatalog	5.6	High	02/10/2005	Yes	3rd party libraries, HL7 v2.3
ResultsProcessor	8.2	Medium	06/01/2005	No	Oracle database, Weblogic Application Server

Component	Direct Calls to UI	Hard-coded Policies and Rules	Security Level Requirement	Updated Detailed Design
LabTestCatalog	No	No	Low	No
ResultsProcessor	Yes	Yes	High	Yes

LIS: Target SOA Environment Constraints

Services need to support different versions of the HL7 standard.

- Patient Portal will use the XML-complaint v3 version of HL7.
- EMR systems (Outpatient, Inpatient) plan to move to HL7 v3 in near term while others do not have any plans.

Services need to take into account the different policy requirements for the same data.

- Research data should be completely anonymous (without any Personally Identifiable Information – PII).
- Inpatient/outpatient data should be completely identifiable for each patient.

LIS: Important Infrastructure Services

Policy Manager

- Centralizes the configuration, deployment, change management and storage of policies

Infrastructure Data Transfer Service

- Used by all the business services to transfer and receive data from external systems

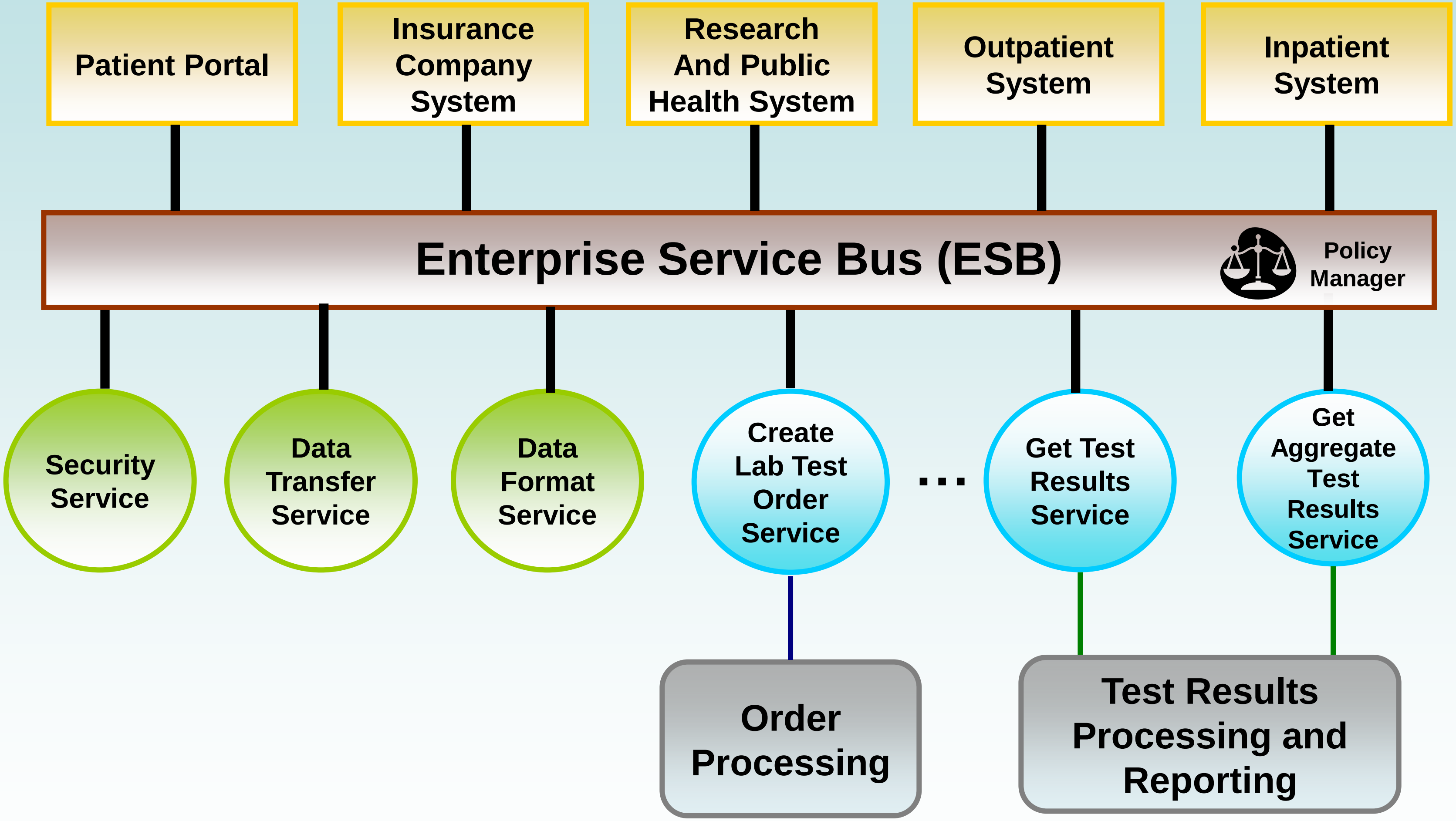
Infrastructure Security Service

- Provides secure transmission of confidential data
- Provides authorization and authentication services

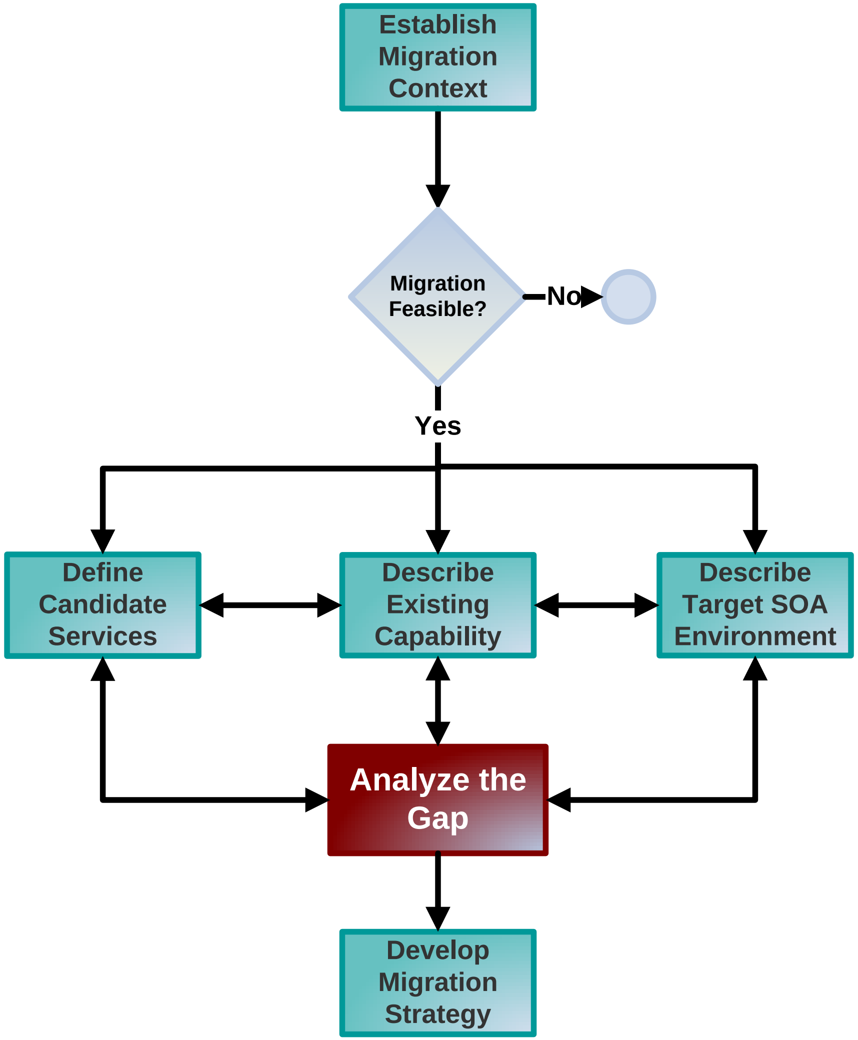
Infrastructure Data Format Service

- Formats messages according to HL7 v2.x or HL7 v3 as needed by business services and applications

LIS: Notional Service-Oriented System Architecture



Analyze the Gap


Establish Migration Context
Migration Feasible?
No
Yes
Define Candidate Services
Describe Existing Capability
Describe Target SOA Environment
Analyze the Gap
Develop Migration Strategy

- Define effort, risk, and cost to convert legacy components into services, given candidate service requirements and target SOA characteristics
- Determine need for additional analyses

LIS: Updated Component Table

Component	…	…	Migration Method	Summary of Changes Required	Effort (Person-Weeks)	Cost	Level of Difficulty	Level of Risk
LabTestCatalog			Wrapping	1. Create an interface that provides the business methods for searching the lab test catalog based on various criteria. 2. Wrap and reuse the existing logic present in the LabTestCatalog component by calling the appropriate method.	3		Low	Low
ResultsProcessor			Extraction + New	1. Create an interface that provides the necessary business methods for getting the test results based on input criteria such as patient id, order number etc. 2. Reuse the business rules inside the *ResultsProcessor* by wrapping and modifying subcomponent code to comply with the new service interface. 3. Create code for the interface methods that are not provided by the *ResultsProcessor* subcomponent. 4. Add input validation code. 5. Add missing input elements to the *TestResults* data structure. 6. …	15		Medium	Medium

LIS: Analyses Performed

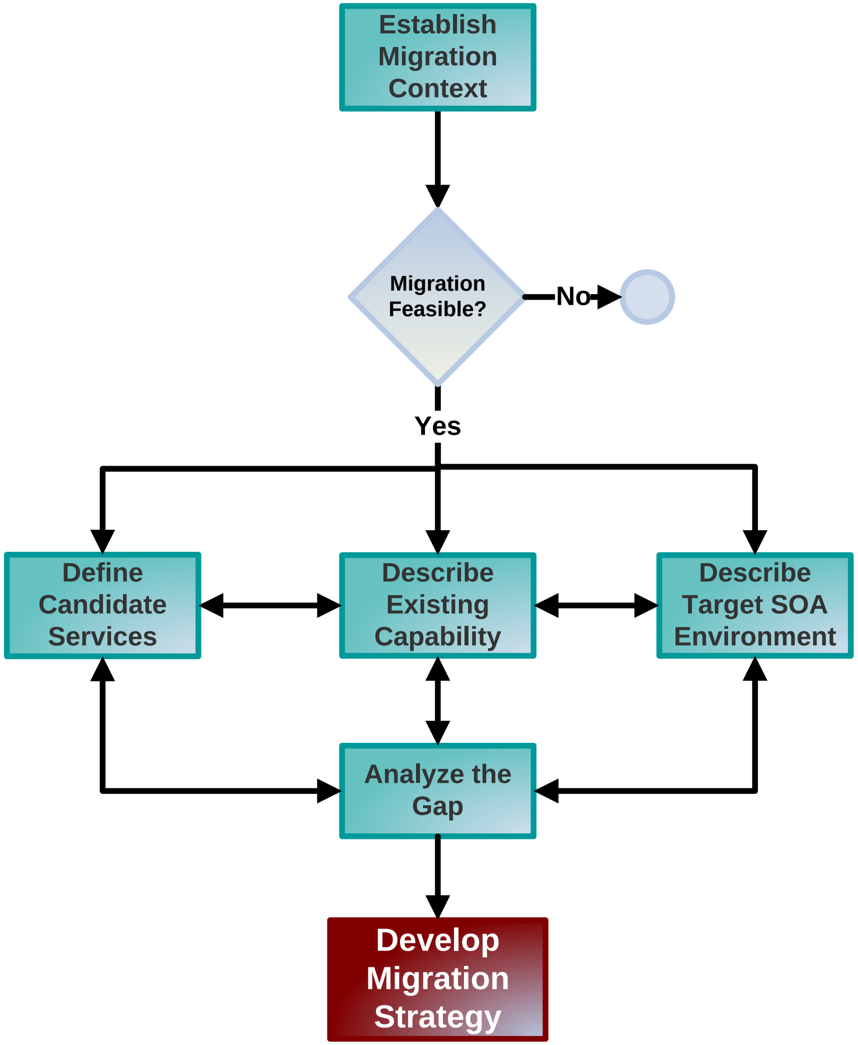
Given the lack of architectural documentation and the lack of confidence in the estimates, two analyses were performed:

- Informal evaluation of code quality
  - No consistent coding standards in force
  - Parts of the code had little cohesion
  - Awkward and non-standard class/modules organization
- Architectural reconstruction to gain a better understanding of code dependencies when the SMART team found discrepancies

LIS: Service-Component Alternatives

Service	Options	Components	Effort (Person-Weeks)	Cost	Level of Difficulty	Level of Risk
Get Test Catalog	Create interface to LabTestCatalog component	LabTestCatalog	3	$ 9,375	Low	Low
	Rewrite code wrapped by LabTestCatalog component in Java		15	$ 46,875	High	Medium
Get Test Results	Create interface to ResultsProcessor components	…	…	…	…	…
…						

Develop Migration Strategy


Establish Migration Context
Migration Feasible?
No
Yes
Define Candidate Services
Describe Existing Capability
Describe Target SOA Environment
Analyze the Gap
Develop Migration Strategy

Develop one or more migration strategies that may include

- Order in which to create services
- Guidelines for creation of services
  - Service reference architectures
  - Source of service code (legacy, COTS, external services, etc.)
  - Mechanism—wrapping, rewriting, extraction, new
- Specific migration paths to follow (e.g., wrap first and rewrite later)
- Needs for training, technology evaluation, market research, etc.

Stakeholder Workshop

Rationale. There are a large number of stakeholders that will be affected by migration of LIS to services. The workshop will help to obtain buy-in for migration.

Goal of the workshop is to

- Share LIS migration plans
- Reach agreement on
  - Timetable for service release schedule
  - Phase-out plan for LIS legacy components supporting current interactions to be replaced by services
- Gather service consumer needs
- Discuss any support to be provided by LIS for use of LIS services
- Start the governance discussion

Initial ESB Selection

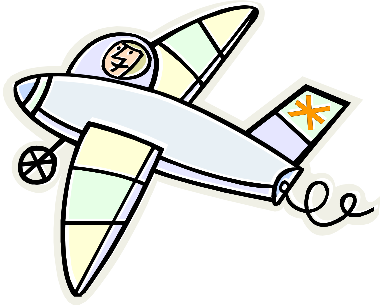
Rationale. There are strong security, privacy and policy requirements that need to be met by the ESB product. There is no context-specific evidence to support that these requirements are met by any of the ESB products being evaluated.

- Perform a preliminary selection based on available evaluation results.
- Work with vendor to obtain a short-term evaluation license.
- Implement the initial SOA Infrastructure
  - Install products
  - Define standards
  - Set up registry …

Implement *Get Test Catalog* Service as a Pilot

Rationale. *Get Test Catalog* is a simple service that is used by multiple internal and external business processes.

- Because the data in the catalog is not patient-related, the service can be more easily exposed to external systems to start testing
- Will provide data to fine-tune migration estimates
- Will also determine if the “double-wrapper” (existing code is a Java wrapper to a COBOL component) has any performance problems



Validate Security and Privacy Requirements

Rationale. LIS is relying on the infrastructure to protect any personally-identifiable information in accordance to HIPAA requirements. The security and privacy provided by the infrastructure may not be enough.

- T-Checks can easily determine if privacy and security requirements are met by the selected ESB product.
- If requirements are not met, the T-Checks can provide information to determine additional elements that would need to be added to the infrastructure to meet requirements.

Understand Policy Management Component

Rationale. LIS is relying on the policy manager to manage all policy currently embedded in LIS components. It is not clear if what is meant by policy in the ESB is the same as what is meant by policy in LIS.

- T-Checks can easily use LIS policy information as the context for experimentation.
- If requirements are not met, the T-Checks can provide information to determine additional elements that would need to be added to the infrastructure to meet requirements.

Evaluate Initial SOA Infrastructure

Rationale. Lessons learned from the pilot and experiment results need to be evaluated against the initial SOA infrastructure.

Potential findings

- Requirements not met by the infrastructure
- Constraints on services
- Quality of service issues
- Incompatibilities with legacy code
- Initial ESB selection is not appropriate

Implement Final SOA Infrastructure

Rationale. The details of the migration will vary depending on the SOA infrastructure. It is important to have a stable infrastructure before adjusting estimates and continuing with the migration.

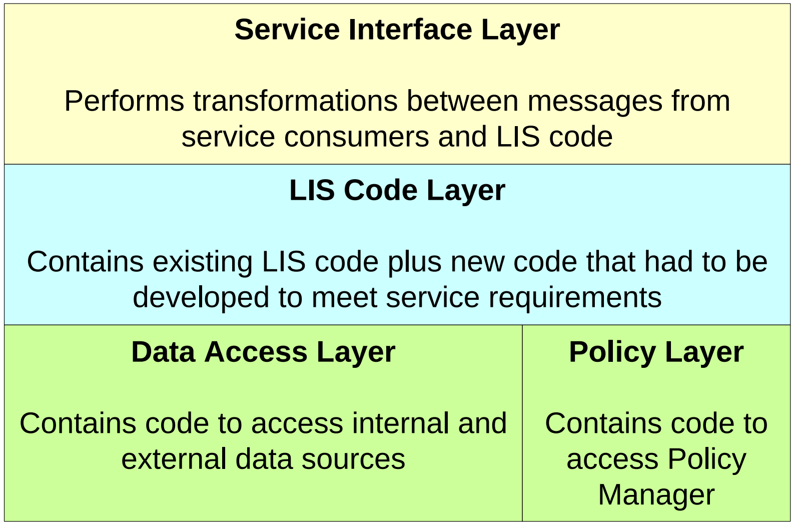
- Define responsibilities of the infrastructure components.
  - Security: Can the service assume that authentication has been done by the infrastructure? Or does the service need to invoke the security service to validate authentication?
  - Data formatting: Will the service call the data format service? Or will the data format service be invoked by the infrastructure before calling the service?
- Define and implement service level agreements and runtime policy enforcement mechanisms
- Identify areas where ESB vendor support is needed.

Document Implementation Guidelines

Rationale. Implementation guidelines will guarantee that all services follow the same development processes, use the same checklists, interact with the infrastructure in the same way, etc.

Beginnings of design-time governance

- Service interface design
- Development checklists
- Service reference architecture
- Testing and deployment procedures
- …



Service Interface Layer

Performs transformations between messages from service consumers and LIS code

LIS Code Layer

Contains existing LIS code plus new code that had to be developed to meet service requirements

Data Access Layer

Contains code to access internal and external data sources

Policy Layer

Contains code to access Policy Manager

Adjust Estimates and Create Migration Plan

Rationale. Lessons learned from the pilot and experiment results will provide additional information on the amount of effort required for migration.

- Finalize service inputs/outputs based on service consumer requirements.
- Adjust migration effort estimates to include SOA infrastructure requirements and any changes in service inputs/outputs.
- Prioritize candidate services.
- Define training needs and provide the training.

Implement Migration Plan

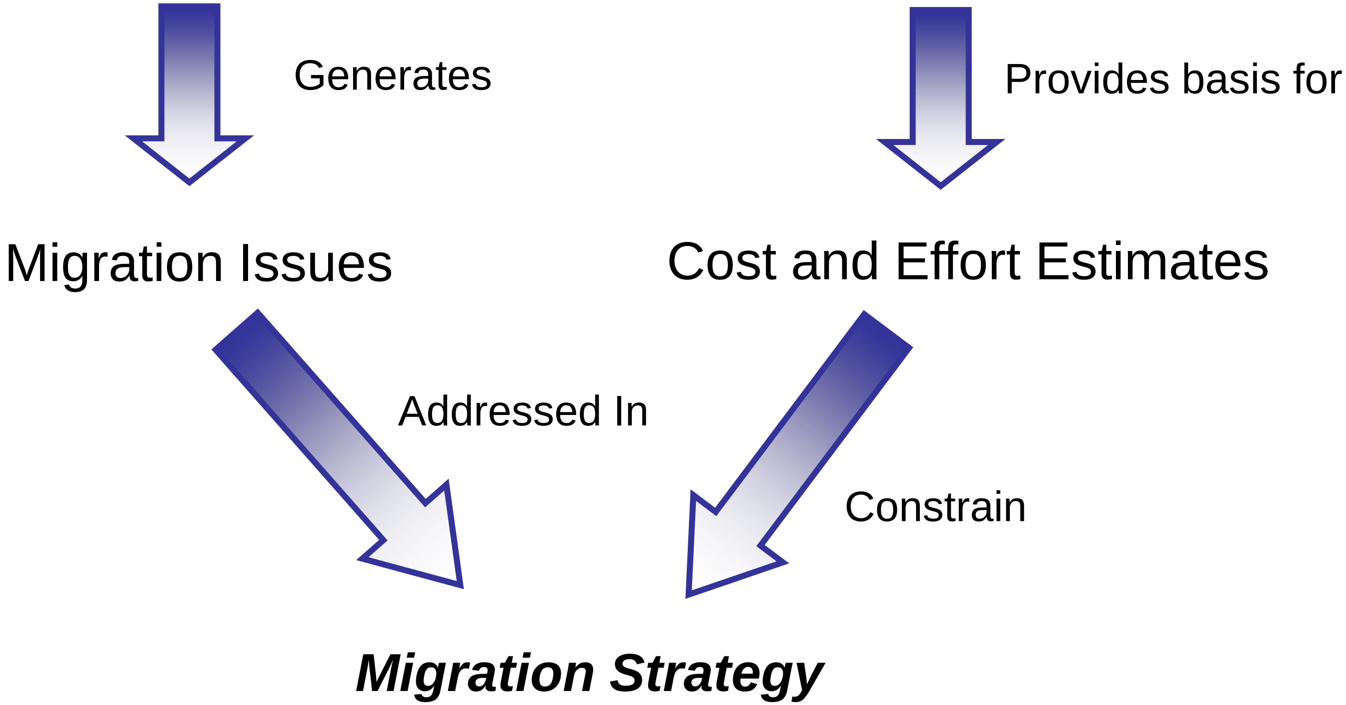
Rationale. Get started! The faster you produce results and start making services available, the faster people will start using them.

Make sure there is feedback between iterations.

- Incorporate lessons learned.
- Evaluate changes in technology.

Process Revisited

Information gathered during *Establish Migration Context*, *Define Candidate Services*, *Describe Existing Capability*, *Describe Target SOA Environment*



Generates

Provides basis for

Migration Issues

Cost and Effort Estimates

Addressed In

Constrain

Migration Strategy

Agenda

Introduction

- SOA Challenges
- Common Misconceptions
- Consequences of Decisions

Introduction to SOA Research Agenda

Pillars of Service-Oriented Systems Development

Challenges of Migration to SOA Environments

SMART (Service Migration and Reuse Technique)

Conclusions

Conclusions 1

SOA offers significant potential for

- Leveraging investments in legacy systems by providing a modern interface to existing capabilities
- Exposing functionality to a greater number of users

They accomplish this by promoting

- Assembly of consumers from existing services
- Platform and language independence
- Reuse of services through loose coupling
- Easy service upgrade due to separation of service interface from service implementation

Conclusions $_2$

End-to-end engineering approach for SOA requires addressing the unique challenges, risks, and technical issues of three different development perspectives.

- Service consumer developers
- Service developers
- Infrastructure developers

Reuse at the service level is more complex than reuse at module or component level.

- Designing reusable services requires a different approach, skill set, and mindset
- Bigger stakeholder community because services are typically reused at organization and sub-organization level

Conclusions $_3$

Cost of exposing legacy system functionality as services may be higher than actually replacing the system with a new service-oriented system.

- Detailed analyses are needed

Reuse in the services world requires

- Identification of requirements of the target SOA infrastructure
- Clear distinction between the needs that can be satisfied by the legacy system and those that cannot be satisfied
- Systematic analysis of changes that need to be made to work with target SOA infrastructure

SMART analyzes the viability of reusing legacy components as the basis for services.