



Case Study

**A Sustainable
eBusiness Platform**

Using Service Oriented Architecture (SOA)

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

Table of Contents

1. EXECUTIVE SUMMARY	4
2. INTRODUCTION TO THE VISION	6
2.1 KEY CONCEPT.....	6
2.2 BUSINESS VISION.....	8
2.3 TECHNOLOGY VISION	9
2.4 GOVERNANCE VISION.....	10
3. BUSINESS GOALS AND BENEFITS.....	10
3.1 GOALS	10
3.2 BENEFITS	11
4. ARCHITECTURE STRATEGIES AND REQUIREMENTS	12
4.1 ARCHITECTURE STRATEGIES	12
4.2 ARCHITECTURE REQUIREMENTS.....	12
5. GUIDING PRINCIPLES	13
6. CONCEPTUAL ARCHITECTURE VIEW.....	14
6.1 DESKTOP.....	17
6.2 PORTAL.....	18
6.3 PRESENTATION LAYER.....	19
6.4 BPM AND BUSINESS PROCESS SERVICES.....	19
6.5 APPLICATION-SPECIFIC COMPONENTS/SERVICES	21
6.6 COMMON BUSINESS SERVICES.....	22
6.7 UTILITY/BUSINESS SERVICES.....	22
6.8 DATA ACCESS LAYER.....	22
6.9 COMMUNICATIONS LAYER.....	23
6.10 REGISTRY SERVICES	23
6.11 SERVICE BUS	25
6.12 SECURITY INFRASTRUCTURE	28
7. DEPLOYMENT VIEW.....	30
7.1 DEPLOYMENT ARCHITECTURE VIEW	30
7.2 RUN-TIME GOVERNANCE.....	30
8. NON-FUNCTIONAL CONCERNS	31
8.1 PERFORMANCE.....	31
8.2 AVAILABILITY	32
8.3 RELIABILITY	32
9. TECHNOLOGY	32
10. GOVERNANCE	33
11. SUMMARY.....	34
APPENDIX A: GLOSSARY.....	35

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

Table of Figures

Figure 1.	Key SOA Terms.....	15
Figure 2.	Service Operations	15
Figure 3.	eBusiness SOA Platform Architecture.....	16
Figure 4.	Web Services Technology Stack.....	17
Figure 5.	Portal Architecture	18
Figure 6.	BPM Modeling Tool	20
Figure 7.	Service Bus	26
Figure 8.	Event Driven Architecture	27
Figure 9.	eBusiness Security Scenarios.....	29

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

1. Executive Summary

Agility, competitive advantage, operational efficiency — all are strategic Company goals. The objective of the transformation plan outlined in this document is to support those goals and to enable Company to maximize and sustain its market leadership.

Keys to achieving these goals, and to realizing maximum sustainable benefits from IT operations, include:

- Alignment with the business
- Agility to evolve quickly to capitalize on new opportunities (translates to time-to-market)
- Ability to scale appropriately to accommodate growth — in new business volume and new business features
- Improving ROI on IT investments
- More effective leverage of IT assets

In this document, we define a strategy for transforming the Order Management (OM) platform and, ultimately, Company’s enterprise IT environment by implementing a services-based eBusiness platform, supported by strong governance and development processes. Best industry practices, and the professional experience of members of the team who crafted this vision, make a strong case for the benefits, business opportunities and ROI that result from adopting a services-based IT ecosystem – commonly know as a Service Oriented Architecture (SOA).

Gartner definition of a Service: As a gross generalization, [from the business’ perspective] a service is a repeatable task within a business process. So, if you can identify your business processes, and within that, the set of tasks that you perform within the process, then you can claim that the tasks are services and the business process is a composition of services.

Adopting this strategy means adopting a new IT perspective on the business — a *process-centric* view rather than a traditional application-centric view. Business-to-IT alignment becomes easier to achieve because business processes are implemented as IT “services” designed to deliver specific functionality — in fact, the alignment becomes a natural consequence of the platform architecture. This means that changes are implemented as 'business changes' and 'IT changes' at the same time, so they are aligned.

Traditional applications are a series of business processes, tightly coupled with each other and with the data required to support those processes, designed to deliver a well-defined outcome. The sequences in which the application processes are executed, the policies and rules that define different paths through the sequence are all “hardwired” into a monolithic application structure. Any change to processes, rules, or policies impact the entire application.

The eBusiness platform defined here turns that monolithic structure inside out and exposes each discrete business process. New mechanisms are introduced into the environment that enable the business to reconfigure existing process flows or implement new process flows (business models), complete with the rules and polices that govern and constrain those models. Maximum agility!

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

However, when thinking about a services-based approach, it's important to consider the opportunity it presents as part of a larger business model. SOA is not just software or an IT discipline. The services model can be applied to the entire business by thinking about what the company does on a day-to-day basis as being composed of a series of repeatable business tasks or components.

“The component view of a business is very different from the traditional business model of rigid, hierarchical organizations and static processes. It is a fundamentally new way to view your entire business model, and it will help you find new ways to enable flexibility and innovation in all aspects of your business. This componentized business model is more flexible, responsive and dynamic and hence is the ideal model for the new hyper-accelerated, networked world.” – Driving Exponential Change, IBM

Best Practice: Successful SOA initiatives deliver more value to the enterprise if they include the participation of cross-functional teams including business, development, operations, and security. These key stakeholders may not be directly involved on a daily basis but it is critical that they learn from early experience and understand the challenges and benefits of SOA adoption.

In the sections that follow, we describe the vision that guides this initiative and a roadmap to achieve the vision. In addition to a technology-agnostic architecture blueprint, we recommend certain key technologies and “mechanisms” that, in the practical context of Company’s experience and existing assets, are leading candidates for enabling the vision.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

2. Introduction to the Vision

Order Management is positioned to lead Company IT on a mission to transform itself into a more strategic and valuable asset and partner with the business by providing a flexible eBusiness Platform that can adapt quickly and cost-effectively to new business requirements. Every mission needs a clear vision of its goals – a vision that motivates and guides those working to achieve it. The objective of this document is to define that vision.

An example of high-visibility company that recently undertook a similar transformation is Federal Express. FedEx is widely recognized as a company that has leveraged information technology in innovative ways to constantly enhance their original model. Yet even though they are positioned as a role model for effective use of IT for eBusiness, in 2003 FedEx felt the need to undertake an IT transformation – a three-year business-process and technology-infrastructure initiative called 6x6 Transformation which included enhancements to their services-based and event-driven system architecture.

Why did FedEx feel the need for this initiative? According to Robert Carter, FedEx EVP and CIO, “Because our customer needs have changed, and we think they'll continue to change exponentially in the next few years... The vision of the 6x6 Transformation initiative is simple: FedEx IT will dynamically align resources to the corporation's critical priorities while improving cycle time and return on investment.”

In 2007, Company IT goals are similar to those that drove the 6x6 Initiative at FedEx:

1. Satisfying changing customer requirements
2. Working as trusted partners with the business and clients
3. Offering greater opportunities for IT employees
4. Improving our ability to deliver results quickly
5. Establishing a consistent environment with enterprise-wide IT infrastructure standards and common processes
6. Simplifying information access

This document describes the vision for transformation of the Company business-IT ecosystem, and specifically, for the OM eBusiness Platform. It defines the blueprint for an architecture that will support current business requirements and ensure that we can accommodate future requirements quickly and with minimal impact to the application systems and platforms.

2.1 Key Concept

At a high-level, we define a business and IT ecosystem consisting of business architecture, a process and service model, and a service-based IT platform. Because the theme of this document, and the foundation for the OM vision, is a service-based architecture (also referred to as Service Oriented Architecture (SOA)), a working definition of this key concept would be helpful.

Gartner predicts that by 2008, more than 60 percent of enterprises will use SOA as a "guiding principle" when creating mission-critical applications and processes.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

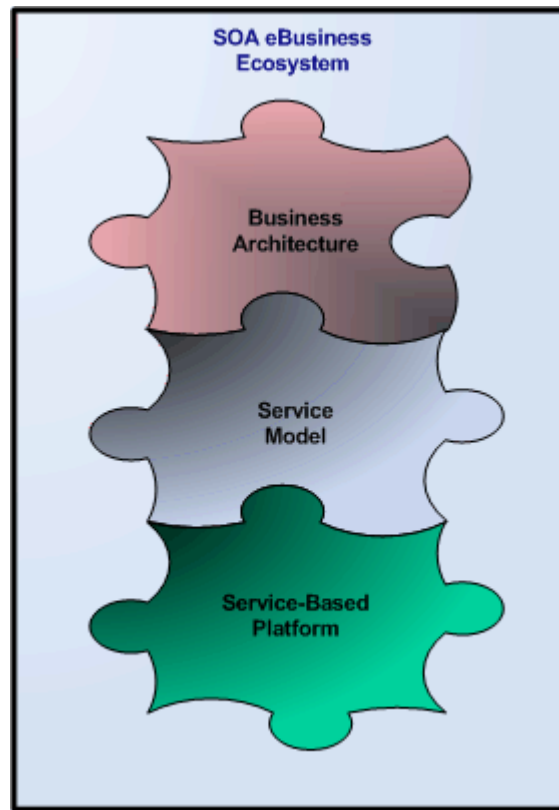
Even though the concept of a “services-based” IT environment is simple, the industry has struggled to reach consensus on a definition. One of the reasons is because it means different things depending on your perspective:

- From the point of view of the business, SOA is a set of services that are configured to form composite applications with dynamic and flexible process flows. Those processes and services can be exposed to customers and partners, or to other parts of the organization
- From the point of view of an enterprise architect SOA is an architectural style that promotes the concepts of business processes and the orchestration of enterprise-level business services. It is also a set of architectural principles, patterns and criteria which address characteristics such as modularity, encapsulation, loose coupling, separation-of-concerns, reuse and composability.
- From the point of view of a project manager SOA is a development approach supporting highly productive parallel development.
- From the point of view of a tester and/or quality assurance engineer SOA represents a way to simplify overall system testing.
- From the point of view of a software developer SOA is a programming model supported by standards, tools and technologies including, but not limited to Web Services.

One theme is common to all definitions of Service-Oriented Architecture: it is a way to decompose enterprise IT systems into smaller, more manageable software artifacts called services. Through service orchestration, these services are assembled to implement enterprise business processes. It provides a process-centric view of the enterprise rather than the siloed application patterns that have proven rigid and costly to maintain and change.

A visual illustration of the synergistic components of the vision, as realized through a service-based platform, might look like the following diagram of closely fitting pieces of a puzzle.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008



As the first document in a continuum of architecture artifacts that will fully define the eBusiness Platform, from enterprise architecture through solution design, this document defines the reference architectures¹ and will explain the reasoning and motivation for decisions related to the design patterns and technology stack on which the vision is based. It will describe how the architecture will support Company standards for scalability, reliability, availability, security, manageability, and other fundamental architectural principles.

2.2 Business Vision

Company's vision is to provide a powerful and flexible eBusiness Platform, a platform that provides secure and appropriate access to eBusiness processes and the data that supports those processes. To ensure that the business has optimum flexibility the platform will be configurable to connect processes, people, and information within the organization and across organizational boundaries.

As Company's business landscape evolves, the ability to launch a robust eBusiness platform from which to offer new products and services can provide significant competitive advantages. Enhanced customer self-service will streamline ordering and fulfillment processes. Software as a Service holds promise as a radical new business model and product offering.

A service-based architecture enables applications to be designed for flexibility, interoperability, and change. Case studies show business benefits like faster time to market, reduced application development cost, and more-competitive electronic service offerings – for example, Software-as-a-Service.

¹ A predefined architectural pattern, or set of patterns, partially or completely instantiated, designed, and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

The extent to which business requirements are fulfilled is directly related to the accuracy with which business process flows and logic are expressed and automated by the business' application software (see, *6.4 BPM and Business Process Services*). Because applications have traditionally been designed to address immediate or tactical requirements, there is always a challenge in keeping applications in alignment with business needs when the requirements and direction of the business changes.

Service-based computing introduces a design paradigm that promotes abstraction on many levels. One of the most effective means by which functional abstraction is applied is the establishment of service layers that accurately encapsulate and represent business models. By doing so, common, pre-existing representations of business logic (business entities, business processes) can exist in implemented form as physical services.

Because services are intrinsically interoperable, configurable and replaceable, this approach directly facilitates business change. As business processes are augmented in response to business climate changes – new competitors, new policies, new priorities – services can be reconfigured into new compositions that reflect the new business requirements. The architecture evolves in alignment with the business.

2.3 Technology Vision

The technology vision is a robust and flexible platform that is linked to and aligned with business strategy and is also modular and capable of adapting to rapidly changing requirements at the lowest possible cost and highest possible quality. Company's services-based (SOA) eBusiness platform is the foundation for realizing this vision. Defining characteristics of the platform follow.²

1. The new OM platform is based on a distributed component architecture. Service components are transparently located inside (or outside) the enterprise and are accessible as services through a stack of universally supported, interoperable remote procedure call and messaging protocols. On the wire protocol interoperability – as opposed to code portability – is the centerpiece of SOA component interactions, enabling universal access and platform independence. Callers are unaware of a component's underlying implementation technology.
2. SOA components encapsulate functionality and enable reuse at the level of abstraction modeled by business analysts in formal business models. This improves alignment by mapping more directly between an IT function and the business function it supports.
3. Declarative, machine-processed contracts enable processes to access the services that an SOA component provides. These contracts explicitly state functional characteristics as well as non-functional (quality-of-service, or QoS) capabilities and requirements. SOA components document their operations using WSDL. Sequences of operations on a component are defined using Business Process Execution Language (BPEL).
4. SOA components can be dynamically found, selected, and bound by means of their declarative properties, and integrated using composition mechanisms to form complex business processes.

Our roadmap will enable us to reach the vision by implementing the new platform in increments – with significant business value added at each incremental milestone. According to a Forrester Research finding, “Although you need the guidance of a strategic vision for SOA, it is not wise to build a strategic SOA platform in a top-down, strategic build-out fashion. Instead, you should

² Some content provided by: *IBM, SOA programming model for implementing Web services*

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

evolve your SOA platform a step at a time based on current day business needs — and do this simultaneously with ongoing development of your strategic vision.”

2.4 Governance Vision

Sustaining the business and technology visions requires a clearly defined, rational set of governance processes, with management sponsorship and support to provide accountability and control. Architectural governance is one aspect of enterprise IT management, and includes:

- Architectural oversight, via a COE or steering committee specifically chartered to
 - ✓ Establish and enforce architectural standards and procedures
 - ✓ Track and monitor new technologies
 - ✓ Provide training and mentoring
- Architecture blueprints and standards
- Technology standards
- Architecture-driven SDLC that includes mentoring and participation by project architects

A key to establishing enterprise architectures that evolve over time and continue to support the business’ changing environment and requirements is a feedback mechanism that accepts input from both the business and IT. Although most of our guiding principles (see 5 Guiding Principles) are immutable, the actual realization of architectural patterns and components will certainly evolve over time. The process of architectural governance must be designed to facilitate that evolution.

Other relevant domains of IT governance include:

- Corporate/Business (to establish goals and strategies)
- Solution Development
- Operations
- Support

Because of the depth and breadth of the topic, specific, detailed governance policies and procedures are documented separately.

3. Business Goals and Benefits

OM’s architectural approach is driven by a number of overarching business goals. Achieving these goals will result in significant benefits for the business and for IT development and operations.

3.1 Goals

Because an application solution is of value only if it supports the business’ goals and mission, the OM architecture is designed to support the following business goals:

- Increase business and IT agility – Enable Company to meet evolving business requirements, seize new opportunities, and maintain our position as industry leader
- Business-model focus (process view) – Choreograph and orchestrate processes and policies that implement innovative and competitive business policies and models

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

- Reduce incremental spending
- Achieve increased ROI
- Drive IT investments based on business objectives
- Simpler and more effective IT governance
- Improve process efficiency – Enable more efficient process flow with a clear view of all orders, from all sales channels, through the sales and fulfillment lifecycle
- Improve quality of application releases
- Leverage existing assets and investments – Leverage high-value software and infrastructure assets as components for the new OM platform.
- Reduce operational overhead

3.2 Benefits

In addition to the benefits that the business will realize as a result of achieving the goals outlined above, the architectural approach defined in this document will deliver the following benefits:

- Improved time-to-market for new business models – As described in Section 6, the new platform will enable definition and deployment of new and revised business models much more quickly than the current OM environment
- Reduced application lifecycle cost (Total Cost of Ownership) – A component and service-based architecture will result in a significant savings in costs for new development and on-going maintenance
- Higher productivity – Development responsibility can be assigned based on the functionality that each component provides, and the skill sets required to implement that functionality. Implementation of new features is faster because changes are isolated to the component to be replaced
- Higher quality – Defects, defined as missed requirements and errors in application logic, are dramatically reduced as a direct result of the architectural approach and supporting SDLC processes
 - ✓ Requirements are easier to understand and implement
 - ✓ Encapsulation of logic into components ensures that the impact of change to a component is isolated to that component. Other aspects of the system are not exposed to the risk of new defects
 - ✓ Developers are able to focus on specific components of logic responsible for the logic or process being changed or implemented
 - ✓ Full test coverage is easier to attain
- Longer useful life – Because the system can more easily evolve over time as new functionality is required
- Reuse of common functionality – Locally, and on an enterprise scale
- High availability and reliability – Enabled by deployment of redundant components

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

Company will realize these benefits – and others -- because a component-based architecture is based on fundamental design principles, referred to as “separation of concerns” and “loose coupling.” Separation of concerns dictates that each component is designed to provide a specific type of service (a specific responsibility). Loose coupling refers to a design approach that minimizes the dependencies between components.

4. Architecture Strategies and Requirements

The OM architecture will enable Company to implement application functionality and platform infrastructures that provide a robust, reliable, and high-performing solution.

4.1 Architecture Strategies

The strategies that will enable the OM project to achieve its goals and realize its benefits include.

- **Component-based.** The architecture is defined as a collection of components that are responsible for specific, distinct business functions.
- **Service Oriented.** Service-Oriented Architecture is an architectural style that physically separates functionality into services that can be used by multiple application and system functions.
- **Plugable architecture.** By using service-based architecture and open standards for integration mechanisms and protocols, business functionality can be added, changed or removed without impacting the system overall. Thus, the design allows the business community to alter the basic nature of the functions performed without changing the overall architecture and infrastructure of the systems.
- **Data abstraction.** By providing mechanisms to define metadata, the solution space is isolated from changes occurring in the virtual data-models of the order and fulfillment systems. The same meta-data and universal data-dictionary are used in the access, manipulation and presentation of data and hence will provide a much higher level of flexibility to the business community as a whole.

4.2 Architecture Requirements

The architecture of the OM platform provides the context within which the application is developed, and defines the guidelines and constraints within which we will make design and implementation choices. The architecture, therefore, establishes the technical requirements for the final realization of the platform. Requirements for the OM platform include:

- **Scalability.** Establishes the ability of an application or platform to provide additional capacity over time. Adding capacity will often require the addition of resources, but should not require changes in the architecture, design, or implementation. There is no inherent limit to the scalability of an application implemented in compliance with this architecture. The platform can be scaled horizontally (e.g., to accommodate increases in user load) and vertically (e.g., to accommodate greater processing complexity).
- **Performance.** The OM solution will meet its required Service Level Agreements (SLA). The OM architecture ensures that application code will be implemented in a way that will enable the solution to meet SLA performance requirements. The application architecture addresses performance concerns by enabling the most effective use of caching, connection pooling, object granularity and coupling, and other techniques that improve throughput and reduce latency.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

- **Reliability.** This attribute is a measure of the reliability of the individual underlying components. System reliability describes the likelihood of component failures at any level – application, server, or network. The architecture supports platform configurations that provide redundant, highly available processes.
- **Availability.** The modular and redundant characteristics of the OM architecture will enable the system to maintain a high percentage of availability. Most system maintenance can be performed without shutting down the application.
- **Security.** The scope of security requirements extends from the edge of the solution domain through fine-grained application functions. It includes the levels of authentication supported, the granularity of role-based authorization, the mechanisms for auditing, the techniques used to ensure data integrity, and the resistance of unauthorized access (intrusion detection). Other considerations may include security credential cache and session timeout, LDAP (Lightweight Directory Access protocol) performance and administration.
- **Flexibility.** The flexibility of a system is the extent to which the system can be modified to meet the changing needs of a business. To be flexible a system has to be simple (in design, not necessarily in functionality), consistent, and modular (component-based). Individual components should be capable of being reorganized to provide new services. Encapsulated service components and careful adherence to “separation of concerns” at all levels of the solution will help enable flexibility.
- **Maintainability.** Ensures that the system can be repaired and components replaced, with minimal impact on availability and level of service. The modular and redundant features of the OM architecture will enable routine maintenance to be performed in compliance with the availability requirements, stated above.
- **Interoperability.** Is dependent on the framework that enables components to work with each other, and the ability to add new components. The OM architecture is based on an open, interface- and service-based design. The result is that OM functionality that may be used by external applications can be made available as a service to those applications.
- **Reusability.** Software applications represent a valuable corporate asset. Component architectures provide an opportunity to create a portfolio of software assets that may be used by multiple projects or applications. The result is a decrease in application lifecycle costs and an improvement in productivity and quality.

5. Guiding Principles³

Architecture principles are used to guide architecture decisions, including project scope, solution design, software and technology purchases, and deployment options. For example, a principle stated as, “should conform to enterprise data standards” guides database design. “Should have the fewest number of physical servers possible” guides deployment decisions. “Should deliver business value in quarterly increments” guides project scope decisions.

The OM conceptual design complies with the following principles:

- **Separation of concerns** – Each component is designed to provide a specific service (a specific responsibility). This principle applies at all levels of granularity. At the

³ For more information about OM architecture guidelines refer to the document, “*OM Architecture Principles and Guidelines*”

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

finest level, a component may be specifically concerned with computing some result from a pair of input values. At a larger granularity of service, the ESB (for example) is concerned with guaranteed delivery of messages and tracking related events. Large-grained components or services are usually composites of smaller services.

- **Loose coupling** – A design approach that minimizes the dependencies between components. By implementing the ability for a component to locate other components, whose services it requires, and to discover at run time what interface protocol is required to use those components, our architecture enforces loose coupling.
- **Process View** – Business is viewed as a series of processes that can be choreographed to form complex business models.

As the OM architecture evolves to detailed levels of design and implementation, we will establish additional design and implementation guidelines and standards for the project.

6. Conceptual Architecture View

The practice of deriving an information system architecture from a business architecture, using a process-centric perspective, results in a Service-Oriented Architecture. The business processes and the tasks from which they are composed can be thought of collectively as services. Thus, the business design is a composition of services, and the policies and rules that define and constrain their use, which form the information system design.

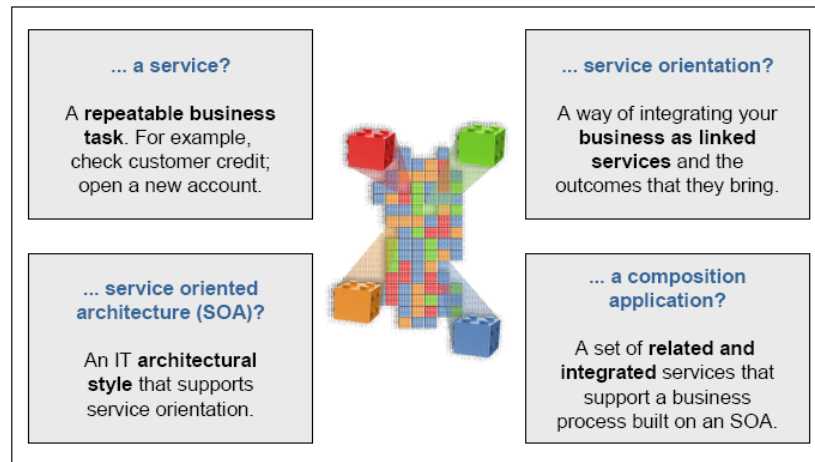
Note: Perhaps the most common misconception about Service-Oriented Architecture (SOA) in the marketplace is that SOA and Web Services are the same thing. This point of confusion is widespread and affects architects and developers, consultants and vendors alike.

Our architectural vision includes the use of Web Services where appropriate but our definition and implementation of “services” is not dependent on the use of Web Services technology and protocols.

Our view of the OM eBusiness platform is a modular, component and service-based architecture (SOA) that decomposes the system into discrete components that provide specific functionality. Figure 2 illustrates some of the terminology used to describe SOA architecture.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

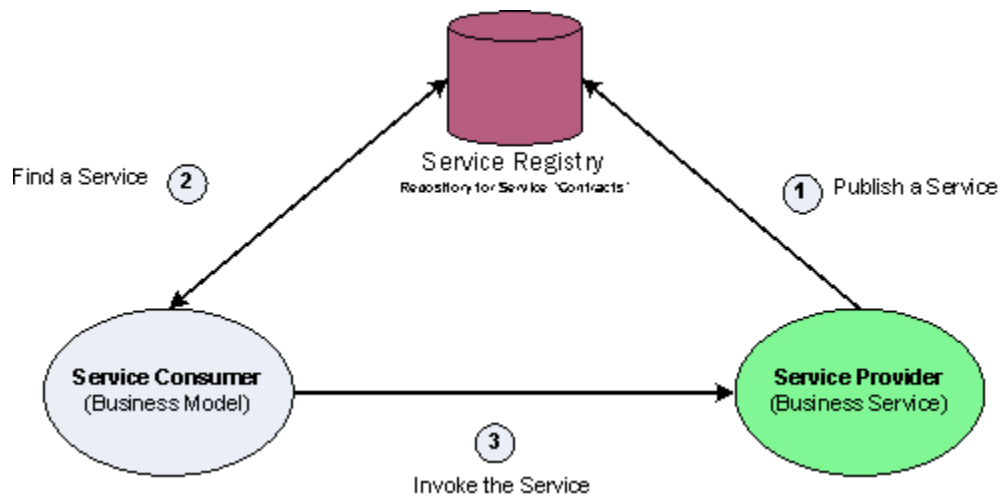
Figure 1. Key SOA Terms⁴



A guiding principle is “separation of concerns.” Each block in Figure 4 encapsulates specific concerns (aspects) of the overall architecture.

Another key concept is that of “consumers” and “providers” of services. Service consumers invoke the services of a service provider, based on a published “contract” that defines how the service is called and what results (service) it provides. Figure 3 is a classic illustration of the process of registering, discovering and using a service.

Figure 2. Service Operations



This approach enables systems to be assembled so that when changes or improvements are required – for example, a better presentation technology is available, or a new business process is developed – the appropriate service component can be changed with minimal impact on the rest of the system. The Service Registry clearly defines each service’s interface so that other components can invoke services without regard to how the component is implemented and how it functions, internally.

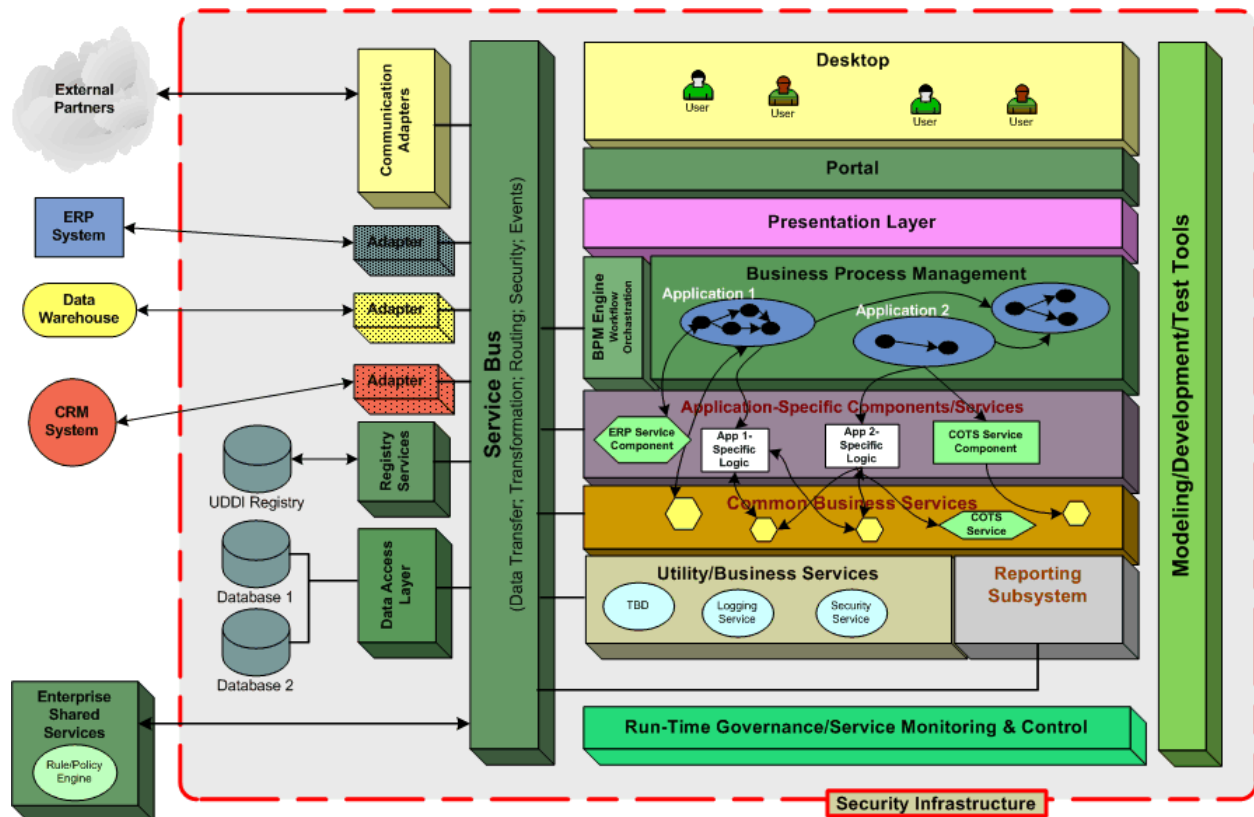
⁴ Illustration provided by IBM.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

The following diagram illustrates the major architectural aspects and suggests their physical relationship with each other. It provides a high-level architectural pattern. This high-level pattern is a composite of more detailed architectural patterns that define each individual aspect.

It is important to view this diagram as a representation of architectural mechanisms and capabilities, not as a physical view of the platform. For example, although there are many mature products available to instantiate each aspect, Company may elect to use any number of approaches – open source, commercial or internally developed – to implement any specific aspect. The significant takeaway from this diagram and discussion is that each aspect is important to realizing the benefits of the component and services-based approach.

Figure 3. eBusiness SOA Platform Architecture



Company’s reference architecture is based on state-of-the-art standards and technologies that are considered best practice in implementing service-based systems. The service technologies are vendor neutral and interoperable. They include a web services framework and associated technology stack (see following diagram) including:

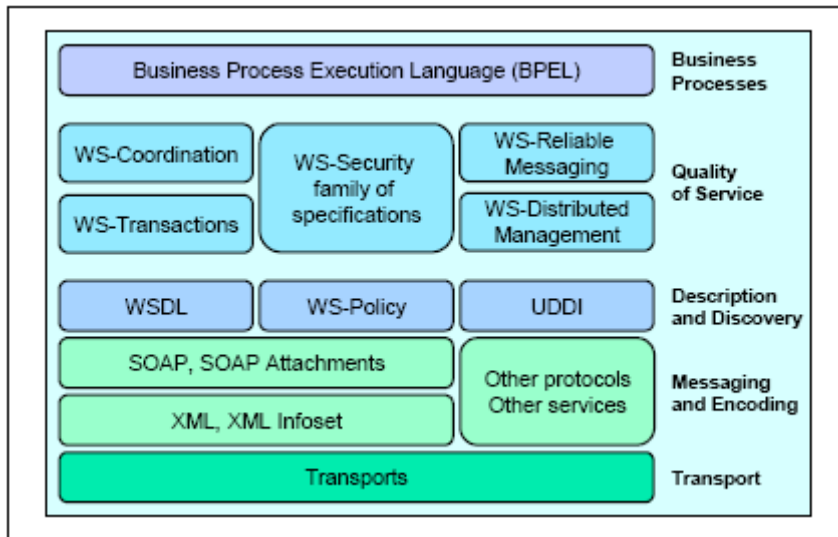
- Extensible Markup Language (XML) – XML is the markup language that underlies most of the specifications used for Web services. XML is a generic language that is used to describe the content in a structured way, separated from its presentation to a specific device.
- Simple Object Access Protocol (SOAP) – SOAP is a specification for the exchange of structured XML-based messages between the service provider, service consumer, and service registry. SOAP, in principle, is a protocol-independent transport.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

- Web Services Description Language (WSDL) – WSDL is an XML-based interface and implementation description language. It describes a service’s “contract.”
- Universal Description, Discovery, and Integration (UDDI) – UDDI is both a client-side API and a SOAP-based server implementation that can be used to store and retrieve information on service providers and Web services.

This reference architecture also includes a service bus (ESB) to ensure loose coupling and a Business Process Management (BPM) engine to provide flexibility and rigor in choreographing and orchestrating business processes.

Figure 4. Web Services Technology Stack



The following sections describe each aspect of the Business Platform architecture, the benefits each offers and alternatives for implementing each aspect, along with the tradeoffs associated with each alternative.

6.1 Desktop

OM’s Desktop environment provides consistent, integrated user access to all OM functions, for authorized users. This user-facing tier is a Web-based interface that provides a single point of access to a wide variety of data, knowledge, and services—at anytime and from anywhere using any Web-enabled client device.

6.1.1 Benefits

Because the Desktop environment is necessary for the delivery of services to users, its benefits are obvious. There are, however, specific benefits related to each implementation alternative (see below),

6.1.2 Products and Technologies

Alternative technologies employed to provide the Desktop environment may include thin-client HTML browser pages or rich client applications, based on an Ajax framework (JavaScript and XML), which uses local resources and XML Web Services and can be deployed and updated from a centralized server.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

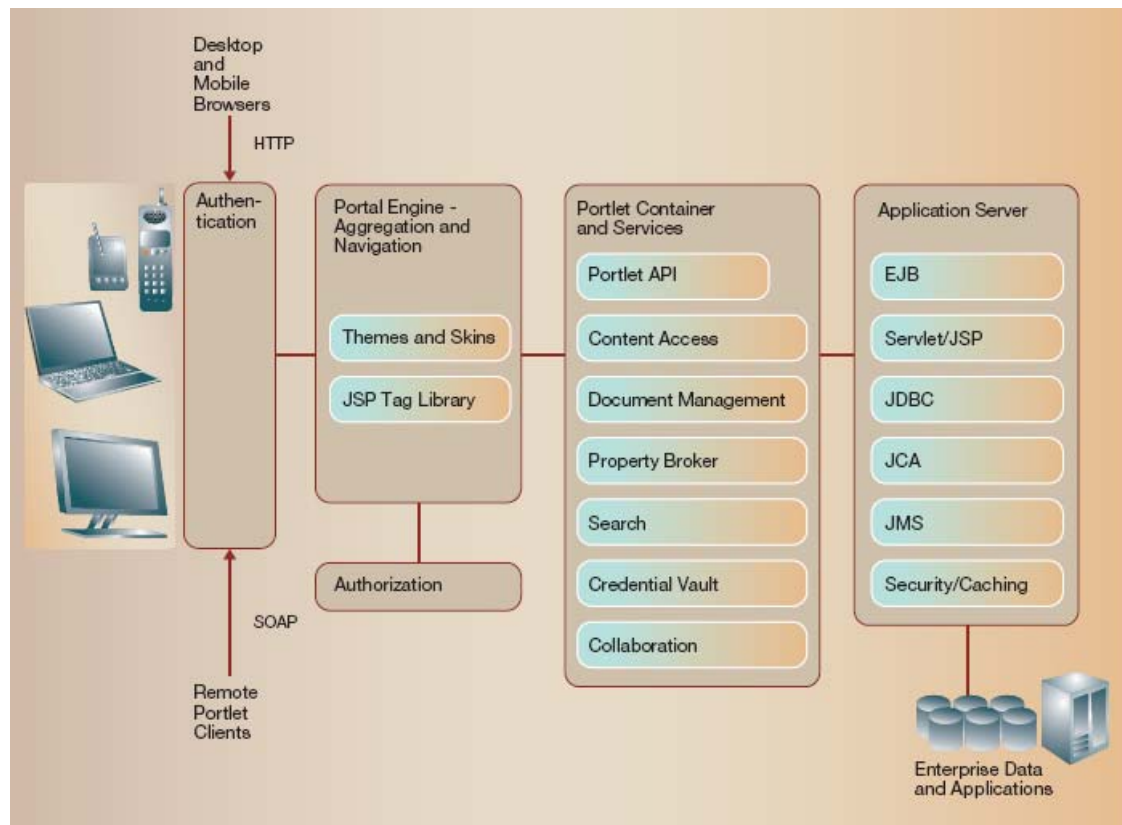
A purely browser-based presentation allows full control of page composition to remain on the Web server. Load on users' clients is low because they are only responsible for rendering the HTML sent by the server. Most user interactions, however, require that the server compose a new page and send it back to the client. For heavily interactive applications, the result is a significant amount of network traffic and load on the server.

Rich internet applications (RIA, and an aspect of Web 2.0) like those provided by Ajax, execute logic on the client – in the form of JavaScript – that can perform many routine tasks locally, without requiring a round-trip interaction with the server. When a specific piece of data is required from the server, the messages between client and server are much smaller than a full Web page, and therefore much faster. Because logic executes locally, applications tend to be faster and appear more responsive. SLAs for interaction may be easier to achieve and sustain.

6.2 Portal

A web portal aggregates applications and content to allow Company users, partners, employees and customers to tailor their user experience, with personalized applications based on role, context, actions, location, preferences and collaboration needs. As the front end to a service-based architecture, portal software provides a component-based model that supports reusable software assets. Figure 5 illustrates the high-level architectural view of a typical portal.

Figure 5. Portal Architecture



Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

6.2.1 *Benefits*

A Web portal interface provides each user with a personalized, unified view of the enterprise and access to specific business functions – or topics – from a single access point. From an architectural perspective, it provides a single point of focus for implementing and controlling consistent and secure access to enterprise assets.

6.2.2 *Products and Technologies*

There are numerous commercial portal products. Because Company has invested in XXX’s suite of products, [XXX], is a logical choice.

6.3 **Presentation Layer**

Presentation layer services support the interaction between applications and end-users. If required, presentation logic can orchestrate the interface to non-human interfaces.

Regardless of to what or whom the service is interfacing, every external interaction shares one thing in common – the ability to project a view of the system tailored to the specific interaction fidelity, frequency of interaction, and presentation composition that best fits the needs of the end user or device.

Interactions are tailored to role-sensitive contexts – adjusting what is seen and the behavior presented to the external world based on whom the user is and what role they are performing. Authentication, authorization, and proximity may all be significant to what a user can do and how.

6.3.1 *Benefits*

Isolating all responsibility for presenting information to the user allows the business to change to the way in which the information is presented – flow, format, validation, etc. – without requiring changes to the underlying business logic. Conversely, it also allows changes to business logic and data models without impacting the users’ view.

6.3.2 *Products and Technologies*

OM currently uses a Struts MVC framework, one of the original web-application frameworks and the most pervasive JEE MVC framework deployed today. An alternative gaining wide acceptance is the Spring Framework. Both Struts and Spring are open source products.

The two viable options for use in the new architecture are:

- Struts 2.0
- Spring MVC

Struts and Spring can be used as complementary frameworks – can co-exist. If OM decides to implement a Spring framework, we will define a roadmap that enables both frameworks to exist until the re-implementation is complete.

6.4 **BPM and Business Process Services**

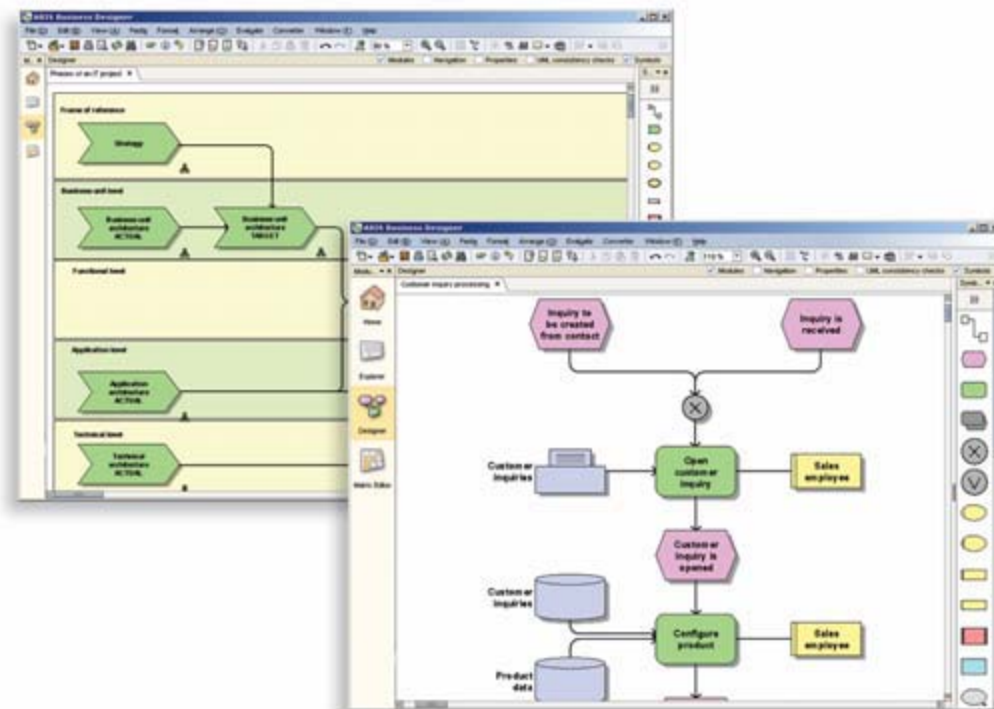
This aspect (or, component) of the architecture provides the realization of “applications” from the vertical view – a set of processes that enable the business to perform a related flow of processes. This new process-centric paradigm maximizes the business’ ability to define new, and redefine existing, business models to accommodate virtually any orchestration of processes necessary to

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

support evolving strategies. In the event that a new model requires new processes, those processes are implemented as services and plugged into (wired into) the process flow.

The following diagram illustrates an example of a typical BPM process modeling UI.

Figure 6. BPM Modeling Tool



Process services include various forms of compositional logic – the most notable of which are business process flows and business state machines. Both kinds of composition mechanisms, plus other forms such as business rules and decision tree processing may be to be equally valid approaches to choreographing service composition.

Business processes are governed by rules which determine how tasks are routed, which task will be executed next, and how long the process can wait for a response from another system or human being. BPM requires two types of management:

- Operational process management
- BAM (business activity monitoring)

The former provides a mechanism for dealing with exceptions, understanding the current state of processes and dealing with in-flight as well as long-running asynchronous processes; the latter provides the means by which both business and process efficiency is managed, optimized and understood.

6.4.1 Benefits

A BPM suite enables business users and business analysts to collaborate to model process flows and the rules and policies that govern that flow. Analysts and developers build forms and

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

processes, simulate process execution, compare simulation results with production data, “playback” the model for testing, and optimize the process.

In a study conducted by the Aberdeen Group in 2006, “Business Process Monitoring systems are achieving an **average 9% improvement in revenues**. Other key business value metrics enterprises have achieved by using Business Process Monitoring systems include **a median 12% decrease in process-related expenses, a 10% return on assets (ROA) and an 18% return on investment (ROI)**. Some 94% of respondents’ companies are either moderately or very pleased with the overall improvement Business Process Monitoring has delivered.”

Some specific benefits relevant to Company include:

- Optimization. A significant BPM benefit is the optimization of business processes across departments and even cross-enterprise. There is a direct correlation between more efficient business processes and specific cost savings.
- Business Agility. Another area where BPM provides a significant return is in improved agility. Business parameters and market conditions change rapidly, and managers need to be able to respond to those changes quickly and effectively. Using BPM, will allow Company to create “on-demand” business environments we can adapt quickly and easily at the business level to build a competitive edge.
- IT Agility. Experience with SOA is demonstrating that, as an enabler to respond to business climate changes, by adapting technology infrastructures to be more responsive, SOA is a best-practice approach to building and sharing applications that run the business. With SOA, Company can build components, or services, of applications that business models will reuse and share across the enterprise to bring cohesiveness to business process.

At Company, specific business requirements and strategic objectives dictate the need for some form of business process management. An evolving, complex multi-channel business model, with complex relationships and variable policies, is one example.

6.4.2 Products and Technologies

Because of *assets already in place*, key candidates for the choice of a BPM product for OM would be:

- XXX
- YYYYYYY

Other potential candidates include . . .

6.5 Application-Specific Components/Services

This aspect of the architecture contains functionality specific to a Company business model (function), implemented as services. An example of a business-specific function implemented in this aspect is GOLD. Application components are not, generally reusable. They contain logic specific to one business function or business model.

These components are created specifically as services within a business model and represent the basic building blocks of business logic – services that are not decomposable within the business model, but that can be composed to form higher-level services.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

6.5.1 *Benefits*

These services may be thought of as the “root” of an application model – defined at the Business Process Services layer (6.4 BPM and Business Process Services). Providing these components as services enables flexible definition of new business models, and enables the business to define (via the BPM) composite applications that consist of multiple, logically related business models.

6.5.2 *Products and Technologies*

As with all of the service layers, these services may be implemented using Web Service technologies, Java RMI protocols or other appropriate mechanisms.

6.6 **Common Business Services**

The common business services aspect contains business-centric services and bases its functional boundary and context on one or more related business entities. These are highly reusable services because they are agnostic to most parent business processes. As a result, a single entity service is leveraged to extend the functionality of multiple application-specific business processes.

Examples of services in this category might include licensing, pricing and access to CPM.

6.6.1 *Benefits*

Encapsulation and isolation of logic; loose coupling; re-use.

6.6.2 *Products and Technologies*

As with all of the service layers, these services may be implemented using Web Service technologies, asynchronous messaging, or Java RMI protocols.

6.7 **Utility/Business Services**

It is beneficial to establish a set of utility services that are not business-centric. This results in a distinct, technology-oriented service layer. Utility services are dedicated to providing reusable, crosscutting utility functionality, such as event logging, notification, and exception handling. They are generally application agnostic in that they can consist of a series of capabilities that draw from multiple enterprise systems and resources, while making this functionality available within a very specific processing context.

6.7.1 *Benefits*

Encapsulation and isolation of logic; loose coupling; re-use.

6.7.2 *Products and Technologies*

As with all of the service layers, these services may be implemented using Web Service technologies, asynchronous messaging, or Java RMI protocols. In specific cases, a utility software product may be deployed as a service at this layer.

6.8 **Data Access Layer**

The Data Access Layer provides an abstraction between the physical data sources required by OM, and the application code that uses that data. The physical structure of the data source is separated from the business logic so that the impact of changes to either the physical database or to the business processing logic is minimized.

In this SOA architecture, the data access layer is defined as a data “service.” Doing so allows OM to make its data available to any consumer – internal service or external application – as

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

appropriate. Definition of a canonical data model is a best practice to support the abstraction of data services.

6.8.1 *Benefits*

A primary benefit is isolation of the physical data structure from consumers of the data. Services or applications are provided a “view” of data the correlates with their intended use.

Providing access to data via encapsulated services also enables effective and flexible security policies. Access and authorization is controlled at the source.

Data is, of course, a critical corporate asset. Providing “data as a service” maximizes the use and value of that data and enables the business to leverage the capability to enhance its strategic value.

6.8.2 *Products and Technologies*

One responsibility of the data access layer is object-relational mapping (ORM). OM is currently considering two of the leading products in this domain:

- Hibernate – an Open Source project and a component of the JBoss Enterprise Middleware System (JEMS) suite of products. Hibernate is one of the most widely used ORM frameworks.
- TopLink – a commercial product from Oracle. TopLink was one of the first ORM products on the market.

6.9 **Communications Layer**

The OM solution is required to send and receive data to a variety of external “end points.” To allow the application to focus only on the data that is to be transmitted, and the logical destination, the Communications Layer takes responsibility for transforming communication requests into physical communications protocols, and manages the communications sessions until the transmission is complete.

This architectural approach enables either end of the physical communication to make changes to the means of communication without impact on the application.

6.9.1 *Benefits*

Primarily a separation-of-concerns benefit – responsibility for preparing messages for physical transport and routing, as well as selecting protocols compatible with the endpoint are isolated in this architectural component.

6.9.2 *Products and Technologies*

The choice of specific communications products will be finalized during the detailed design phase and will probably be selected from the technology stack provided by the platform – e.g., WebSphere.

6.10 **Registry Services**

Governance and the need for a point of control within the SOA environment demands that service information artifacts be stored and managed in a consistent manner that allows enforcement of organizational policies. This is the role served by a registry service in an SOA deployment.

The following example describes a registry-repository using a library metaphor:

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

- A registry-repository is like a local library.
- It has a repository that contains all types of electronic assets, much like the library bookshelves.
- It has a registry that contains metadata describing the electronic artifacts, like the library's card catalog contains information describing the published content on its bookshelves.
- The registry and repository are administered jointly. In a library, the card catalog information and books on the shelves are administered jointly.
- Across the enterprise, any number of registry-repositories should be able to work together to offer a federated, unified service, much like multiple libraries can participate in a cooperative network and offer a unified service.

An SOA registry service should provide governance capabilities that enable organizations to define and enforce organizational policies governing the content and usage of the artifacts throughout their life cycles. Since organizational policies vary – e.g., security and authorization – an SOA registry-repository should enable organizations to enforce custom policies for the governance of any type of service throughout its life cycle. In particular, it should enforce conformance to such policies when a service information artifact is published or updated.

Most service registries comply with UDDI (Universal Description Discovery and Integration protocol) protocols. The following description is taken from the Oasis UDDI Specification.

“For Web services to be meaningful there is a need to provide information about them beyond the technical specifications of the service itself. Central to UDDI’s purpose is the representation of data and metadata about Web services. A UDDI registry, either for use in the public domain or behind the firewall, offers a standard mechanism to classify, catalog and manage Web services, so that they can be discovered and consumed. Whether for the purpose of electronic commerce or alternate purposes, businesses and providers can use UDDI to represent information about Web services in a standard way such that queries can then be issued to a UDDI Registry – at design-time or run-time – that address the following scenarios:

- *Find Web services implementations that are based on a common abstract interface definition.*
- *Find Web services providers that are classified according to a known classification scheme or identifier system.*
- *Determine the security and transport protocols supported by a given Web service.*
- *Issue a search for services based on a general keyword.*
- *Cache the technical information about a Web service and then update that information at run-time.*

These scenarios and many more are enabled by the combination of the UDDI information model and the UDDI API set. Because the information model is extremely normalized, it can accommodate many different types of models, scenarios and technologies. The specification has been written to be flexible so that it can absorb a diverse set of services and not be tied to any one particular technology. While a UDDI Node exposes its information as an XML Web service, it does not restrict the technologies

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

of the services about which it stores information or the ways in which that information is decorated with metadata.”

6.10.1 Benefits

As an indication of the importance of a service registry in service-based platforms, a Gartner opinion in 2006 stated, *“In 2006, lack of working governance mechanisms [e.g., registry] in midsize-to-large (greater than 50 services) post-pilot SOA projects will be the most common reason for project failure (0.8 probability).”*

Primary benefits include:

- Enables and supports reuse of services across the enterprise. By enabling discovery and self-describing contracts for usage, a service registry makes the goal of software asset reuse a reality
- Provides a mechanism for governance of asset usage and security.

The registry is a central element of the service-oriented approach to software design. By enabling policy-based distribution and management of enterprise Web services, a registry delivers significant business value.

6.10.2 Products and Technologies

Generally, service registries are embedded in other mechanisms, like application servers, ESB and workflow engines. Some specific registry products include:

- IBM WebSphere Service Registry and Repository (WSRR)
- Systinet Registry – A mature and widely used SOA registry
- SAP Services Registry
- Sun Service Registry

The choice of a service registry for Company will probably depend on other components of the technology stack.

6.11 Service Bus

The enterprise service bus (ESB) is a silent partner in the SOA logical architecture. Its presence in the architecture is transparent to the services of your SOA application. However, the presence of an ESB is fundamental to simplifying the task of invoking services – making use of services wherever they are needed, independent of the details of locating those services and transporting service requests across the network to invoke services wherever they are deployed in the enterprise.

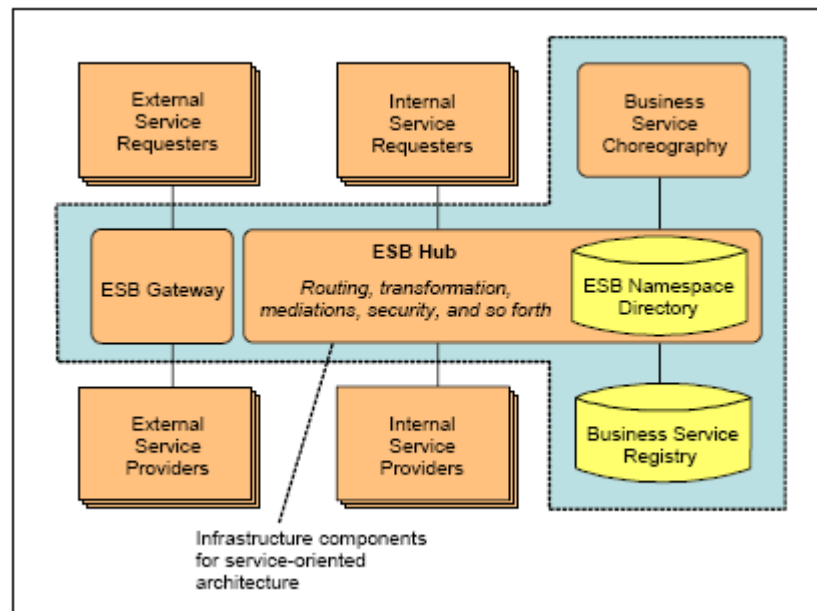
An ESB provides an infrastructure that removes the direct connection dependency between service consumers and providers. Consumers connect to the bus and not the provider that implements the service. This connection further decouples the consumer from the provider. A bus also adds value by providing capabilities such as security and guaranteed delivery. It is an advantage to implement these capabilities centrally within the bus at an infrastructure level rather than within the application. The primary driver for an ESB, however, is that it increases decoupling between service consumers and providers.

In the following diagram, an ESB is depicted as a logical component in a service-based architecture as the mediator between the service consumers and service providers. The service

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

providers and service consumers never interact directly. The ESB provides services to resolve differences in protocol and format, and decouples the service consumer from the service provider.

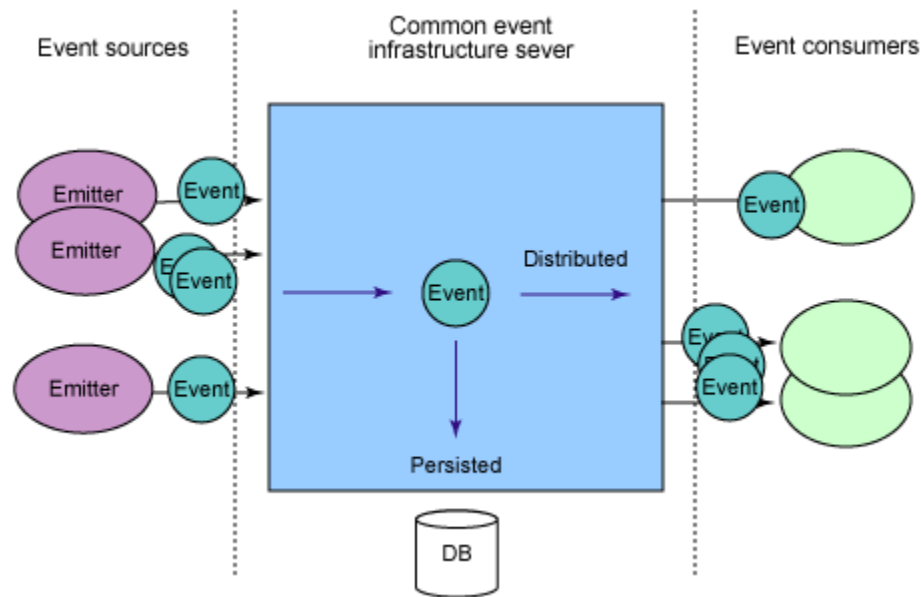
Figure 7. Service Bus



A service bus supports implementation of an “event driven architecture” (EDA). EDA defines a methodology for implementing applications and systems in which events transmit between software components and services. EDA complements SOA by enabling long-running asynchronous processes. An EDA node (application or service) posts events to the ESB and does not depend on the availability of consuming service service. It is decoupled from the other nodes. EDA is often referred to as "event-driven SOA".

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

Figure 8. Event Driven Architecture



An ESB is an architectural pattern and can be implemented by many different products across the enterprise, and assembled together to act as a federated enterprise bus. Many vendors are now offering a complete product to fulfill enterprise integration needs and there are several open-source enterprise-capable products.

6.11.1 Benefits

A service bus enables faster, simpler, and more flexible integration, which enables the business to implement services quickly. It reduces the number of interfaces and improves the reusability of interface components to cut cycle time from design to deployment.

An ESB provides features that improve responsiveness, customer service, transaction time, and partner interactions. An ESB provides capabilities that enhance both direct connection between applications and routing requests between applications.

An ESB supports the concepts of a services-based implementation by:

- Decoupling the consumer's view of a service from the implementation of a service
- Decoupling technical aspects of service interactions
- Integrating and managing services in the enterprise

Decoupling the user's view of a service from the implementation increases the flexibility of the architecture. It allows substitution of one service component for another (for example, if a new service is required to fulfill new business requirements) without the user being aware of the change or without the need to alter the architecture to support the substitution.

From a practical, business perspective, what are the benefits of an event-driven architecture? Users view virtually all activities as "events" – contract in place, order completed, fulfillment complete are a few obvious examples. Enabling the system to respond automatically, based on policy and workflow rules, provides a powerful process governance and support capability.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

At the system level, event generation and monitoring enable run-time monitoring and control to support consistent service levels. For example, we may define a five-second threshold for response to a user request. In the event that those transactions begin to fail to complete within the specified SLA, an event can be generated to cause a reallocation of resources.

6.11.2 *Products and Technologies*

Service buses are one of the most mature and long-standing of the architectural mechanisms define in this architecture. Some obvious candidates for Compnay’s use include:

- XXX
- MULE – Open source product from MuleSource
- WSO2 – Open source product based on Apache Synapse and Apache Axis2 projects
- YYYYYY Fabric ESB

It’s worth noting that there appears to be a new generation of light-weight, high-performance service bus products emerging.

6.12 **Security Infrastructure**

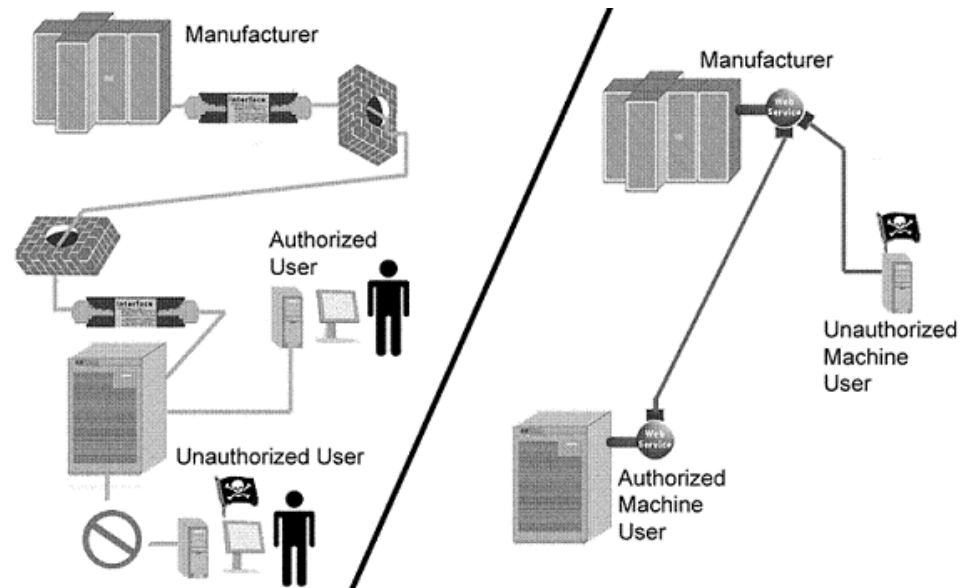
Security concerns in the first generation of OM have primarily focused on user authentication (GSSO) and authorizations (Entitlement). As OM evolves to an eBusiness platform, with external customer-facing features and business-to-business capabilities, additional dimensions of security begin to surface.

In a traditional security model, a system's security mechanisms, such as a firewall or virtual private network (VPN), screen out unauthorized (human) users. An externally facing e-business platform, however, requires that the architecture be open to access from multiple systems and external non-Company users. If services are exposed without an appropriate security mechanism, a hacker can configure a machine to impersonate an external partner or customer’s system and make fraudulent service calls.

The following diagram illustrates the two aspects of eBusiness security – human to machine interaction and machine-to-machine interaction.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

Figure 9. eBusiness Security Scenarios



Security is an issue in service-based environments because the SOA stresses machine-to-machine interaction, while most IT security is based on human-to-machine interaction. Authentication and authorization become more challenging in this environment. It is virtually impossible to block unauthorized use of web services in an unsecured SOA. It is relatively easy to intercept a message in an unsecured SOA and reroute it or transform its content for illicit purposes.

There are solutions to SOA security that can address all of these issues. As we determine an appropriate level of security – during the physical architecture design – we will consider implementing a security solution that enables SOAP message monitoring; federated authentication; application proxy; contract management; certificates, keys, and encryption; and audit logging.

6.12.1 Benefits

The benefits of a secure eBusiness environment are best described in terms of risk management and containment. No commercial enterprise would want to launch a Web-based platform, and particularly a customer facing platform with rigorous security in place.

6.12.2 Products and Technologies

Implementing appropriate levels of security for OM may include:

- Public/private key encryption (PPK)
- Encrypted XML messages. SOAP messages used to communicate with web services are encrypted to ensure privacy.
- Security Assertion Markup Language (SAML) – an assertion (token) that expresses the authenticity of the user in a way that will be accepted by the web service that the user is invoking.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

7. Deployment View

A key benefit of a component-based system is that it enables great flexibility in deploying software components and services to achieve maximum performance and resilience.

7.1 Deployment Architecture View

We will provide a physical deployment view as we define the physical architecture, in subsequent releases of this document. The architectural approach defined herein enables a great amount of flexibility in how all aspects of the architecture are implemented and how they may be deployed..

7.2 Run-time Governance

Componentized and services-based environments are most effective when monitored and controlled by an effective run-time governance infrastructure. There are numerous commercial products available that provide extensive features for ensuring that the run-time environment stays healthy and performant.

Governance can allow the business to establish flexible SLA policies to cover a variety of circumstances. For example, on an eBusiness platform it may be desirable to ensure that external users (customers) are given priority in the use of resources. The governance infrastructure can react to degrading performance by throttling internal users' access and giving priority to customers.

Some other driving requirements for a run-time monitoring and control include:

- Visibility – What services are deployed in the environment
 - ✓ What services, SOA components, and infrastructure are installed and in use
 - ✓ Who is using them
 - ✓ Are they meeting expectations / SLAs
- Control – Taking action to prevent or correct Issues...
 - ✓ Define and enforce runtime policies – make sure proper policies are active
 - ✓ Diagnose failures / prevent them
- Ensuring Integrity – Ensuring changes do not impact the application environment...
 - ✓ Automatically check for the correctness of the running system
 - ✓ Detect and validate changes before they impact users and partners
- Exception Management – Identify and react to exceptions in real-time...
 - ✓ Transaction tracking and failure events
 - ✓ Alerts when processes or transactions do not complete
- Enforce Service Level Agreements – Dynamically take corrective action...
 - ✓ Enforce agreements based on business criteria, e.g., “Gold” users, Accounting systems at the end of quarter
 - ✓ Activate more resources (blade servers, load balancing, etc.)
 - ✓ Redirect traffic
 - ✓ Make lower-priority users wait until high-value users complete transactions

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

Successful, scalable management of SOA requires the ability to take action and intervene to ensure the run-time environment is under control.

8. Non-Functional Concerns

In addition to the functional aspects addressed by this architecture, there are non-functional requirements that any system or platform must meet. These requirements must not be lost or subsumed to the functional elegance of the architecture. They are equally – sometimes even more – critical to meeting business goals and requirements.

8.1 Performance

Performance is a well-known issue when architecting, and implementing, a service-based architecture. Early adopters, over-enthusiastic about the obvious benefits that SOA represented, failed to consider performance early enough (at the conceptual architecture level) and ended up with SOA implementations that failed miserably because they could not perform, or were inordinately costly because the hardware infrastructure had to be scaled considerably beyond the budgeted levels. As often happens in the early stages of the adoption of new technologies, those failures created a general impression that it is too expensive and complex to implement a high-performance SOA architecture.

8.1.1 Latency

Transaction latency is a primary focus area when designing and implementing services-based solutions. The nature of componentized software applications introduces more potential bottlenecks in the end-to-end flow of data from source, through transformations and processing, and to the consumer (user). Depending on the specific source of latency, there are a number of techniques to reduce or virtually eliminate delays.

When the performance cost of loosely coupled components is too high, tighter coupling may be appropriate. Although counter to the principle of loose coupling, there are situations when a compromise can significantly improve performance. One example is avoiding the use of the service registry to “find” a service. If the service definition can be “known” in advance, overhead of invoking the service may be much lower (also see, Service Granularity).

8.1.2 XML Overhead

XML has emerged as the standard language of service-oriented infrastructures, forming the basis for all current SOA standards. The performance issues in SOA deployments due to verbosity and size of XML, combined with chatty transport protocols is an area that requires specific focus. Different approaches have been adopted to deal with this performance bottleneck, key among which are:

- XML compression into binary format
- Caching
- Offloading via accelerators (appliances)
- Transport-level optimization
- Optimized XML processing

While the first three approaches listed above are common in commercial deployments in different products and custom implementations, the latter two – transport level optimization and optimized XML processing – are still new to the SOA product community.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

8.1.3 Service Granularity

Probably the most obvious performance factor in a service-based architecture is related to the granularity of services. While finer granularity optimizes our ability to orchestrate composite services, and to change them easily, careful analysis of our service model and its map to the business process model, often reveals that there is not practical need to decompose certain functionality to the degree that is physically possible.

As a general guideline, services implemented as Application-Specific services (see, Architecture Figure 3) are very course grained. These services encapsulate the largest amount of related functionality – they tend to be the equivalent of “application” systems in an application-centric view. Services in the next category, Common Business services, are finer grained components and services. The scope of their responsibility (concern) is much narrower. Utility services are implemented at the finest granularity; they have a very specific function, usually very narrow in scope.

The level of coupling often tracks inversely to the granularity. Large-grained business services are loosely coupled and may be implemented as Web services. The fine-grained components are accessed as local services, or even from a library.

8.2 Availability

Service-based platforms greatly enhance IT’s ability to provide 24/7 availability. Because application components and the infrastructure that supports them are location independent and can be distributed, it’s practical to implemented redundant services, on redundant clustered and load balanced servers.

When application components need to be replaced, the SOA architecture (specifically, the service registry) allows component versioning. Consuming applications don’t need to be stopped while services are updated.

8.3 Reliability

Reliability is not a new non-functional requirement. It has been one of the primary “...ilities” for years. Service-based architectures introduce some opportunities for enhancing reliability, and some challenges that must be addressed. Some of the specific characteristics and mechanisms employed to implement a service-based system that facilitate high reliability, include:

- Service Bus. My decoupling sender and receiver of messages, and providing guaranteed delivery (via persistent queues) the ESB prevents loss of transactions and protects against the failure of any specific node.
- Redundant services, on clustered servers ensure that the physical failure of any single node, or network segment, does not cause the system to fail.

A cautionary note is that, this architectural approach only *enables* deployment of a highly available platform. IT must provide the appropriate hardware and network infrastructure to realize the availability benefits.

9. Technology

At the conceptual level, the OM architecture is technology and product agnostic. In practice, however, there are obvious best practices and cost-effective approaches for implementing each architectural feature. For example, we could choose to implement our own service bus for messaging and event management – using technologies like Java, database triggers, and JMS. This “home grown” approach would not only be a more costly and risky implementation choice,

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

it would result in a long-term maintenance issue. The better choice is to choose one of the proven, supported products – commercial or open source – available off the shelf.

The following table defines key technologies that comprise the SOA stack, and lists potential candidates for each.

Item	Description	Tradeoffs
Details deleted to honor confidentiality agreements		

10. Governance

Service Oriented Architecture requires a shift in the way software is developed and deployed. The paradigm shifts from “Develop now, Integrate later” to a “Develop for Integration”. This new paradigm, combined with the technologies and standards created to support this shift, requires SOA to be implemented in a well-planned, well coordinated, and effectively managed way.

To ensure business continuity, reduce integration costs and complexities, limit corporate liabilities including security, and effectively compete in the marketplace, Company will govern the design, development, deployment, and operations of new services in the enterprise.

Governance is the ability to ensure that all of the independent efforts (whether in the design, development, deployment or operations of a Service) come together to meet SOA requirements. In the early stages of introducing an SOA ecosystem, the most effective Architectural Governance structure is a centralized authority able to provide singular focus on defining and establishing standards and procedures.

In order to effectively implement and manage the SOA environment, the answers to the following questions need to be answered and monitored:

- Policies – what are the policies that we have? Where are they implemented?
- Business Models – what are the business models to be developed?
- Business Services – what services, at what level, do we need?
- Compliance Status – to what degree do our services conform to our policies? Which interfaces are not in compliance? What is the impact of the noncompliance on the operations of the Services or our business operations (e.g. security breaches, service levels, etc.)?
- Impact Analysis - what happens to our SOA operations if we change our current policies? How do we add new policies?
- Interdependencies - how will operations be impacted by changes made to Services? Which critical processes will be effected or cease to operate?
- Exception Management - can we grant an exception to a defined policy for a certain

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

project? What will be the impact of that exception?

Governance applies to all stages in the services lifecycle.

- Service design and development practices
- Configuration management
- Release management
- Contract management
- Service monitoring and control
- SLA management
- Runtime mediation
- Incident management
- Change management

There are three levels of design-time governance that should be taken into account:

- Portfolio governance: what initiatives, projects, and deliverables (both services and applications consuming those services) get funded and the tracking of those projects' progress against funding and timeline objectives.
- Architectural governance: ensuring that the services and applications developed under SOA initiatives conform to the organization's business and technical architectures and best practices.
- SDLC governance: the day-in day-out application of SDLC best practices (e.g., unit test before code promotion, peer review of code changes, establish an SCM system with code promotion levels) when developing services.

Portfolio and SDLC governance have been around for quite some time, and a number of tools exist to support these governance activities. SOA, because of its loosely coupled nature places increased importance on architecture governance. If effective architecture governance is not instituted, there will be a high risk that service spaghetti will result.

11. Summary

The intent of this document is to communicate the architectural vision and to gain consensus with Company business and technical communities. The document fulfills that intent by

- Documenting the strategic business goals
- Defining the architectural goals
- Providing a conceptual design to support the attainment of those goals

The Architecture Vision document will continue to evolve to present a more detailed view of the OM architecture and design.

Case Study	Version: 1.0
SOA eBusiness Platform Architecture	Date: 1/20/2008

Appendix A: Glossary

Term	Definition
Enterprise Service Bus (ESB)	An enterprise service bus (ESB) refers to a software architecture mechanism, implemented by technologies found in middleware infrastructure products based on standards, that provides foundational services for more complex architectures via an event-driven and standards-based messaging engine (the bus). An ESB combines event-driven and service oriented approaches to simplify integration. It acts as an intermediary layer to enable communication between different application processes.
Service Registry	A catalog or index that acts as the “system of record” for the services within an SOA.
SAML	Security Assertion Markup Language (SAML) is an XML-based standard that provides a framework for describing security information about a user.
BPM	Business Process Management.
SDO	Service Data Objects (SDO) provide a framework for the design and use of business objects in an SOA. The fundamental concept in the SDO architecture is the <i>data object</i> . SDO is often used interchangeably with the term data object.
SCA	Service Component Architecture (SCA) is a model for application development that splits the application function from the implementation details. SCA defines <i>modules</i> and <i>components</i> that are connected using standard interfaces.
WebSphere Process Server	WebSphere Process Server is an SCA-compliant runtime element that provides a fully converged, standards-based process engine that is underpinned by WebSphere Application Server. It is, along with WebSphere Enterprise Service Bus, a strategic product for integration and modernization of IT assets, including core systems using SOA