

Real-World SOA: Definition, Implementation and Use of SOA with CentraSite™

White Paper

Getting Started: Your Guide to SOA Success

1 SUMMARY	2
2 INTRODUCTION TO SOA	3
2.1 DEFINITION OF A SERVICE-ORIENTED ARCHITECTURE	3
2.2 SOA FEATURES	3
2.2.1 Business considerations / aspects	4
2.2.2 Technical considerations / aspects	4
2.3 SOA BENEFITS	6
3 IMPLEMENTING SOA	8
3.1 IMPLEMENTATION ON THE BUSINESS SIDE	8
3.1.1 Step 1: Establish an SOA Vision	8
3.1.2 Step 2: Building an SOA Roadmap	8
3.1.3 Step 3: Establish an SOA Methodology	9
3.1.4 Step 4: Link SOA with Key Business Initiatives	9
3.1.5 Step 5: Create Architectural Blueprints	10
3.1.6 Step 6: Assess Risks	10
3.1.7 Step 7: Create an SOA Risk Mitigation Strategy	11
3.1.8 Step 8: Process-Driven Integration	11
3.2 IMPLEMENTATION ON THE TECHNICAL SIDE	11
3.2.1 Step 1: SOA Enablement	11
3.2.2 Step 2: Synthesis—Single Views and Service Orchestration.....	14
3.3 SOA GOVERNANCE.....	16
3.4 SOA SECURITY	16
3.4.1 Web Services Security Server	17
3.5 SOA VISIBILITY AND GOVERNANCE WITH CENTRASITE	18
4 SOA CASE STUDIES	19
CASE STUDY 1: LARGE NORTH AMERICAN BANK	19
CASE STUDY 2: US-BASED INSURANCE COMPANY	20
CASE STUDY 3: MAJOR TELCO SERVICE PROVIDER	21

1 Summary

This paper provides an overview of SOA, describing basics of web services and how to approach an SOA project. Key SOA technology is described including tools to build those services, repositories and governance policies to manage those services, and registries to advertise them for discovery. The paper concludes with three sample case studies and common scenarios where SOA can be used to solve some of today's common issues, and how CentraSite™ is a powerful enabler for delivering SOA solutions.

2 Introduction to SOA

More and more organizations are turning to a Service-Oriented Architecture (SOA) to increase productivity, enhance operational efficiency and agility, and align their IT infrastructures with business strategies.

This high-level white paper gives you insights into key considerations that can help you successfully launch an SOA. Use this guide to understand best practices that can get your organization off to the right start.

2.1 Definition of a Service-Oriented Architecture

The SOA landscape is strewn with buzzwords, conflicting technologies, architectures and pitfalls. There's no single approach or "solution." So exactly what is an SOA—and what does it provide?

SOA is an application architecture, in which application components or "services" are well defined using common interfaces, utilize a contract to define how services will be invoked, and interact in a loosely coupled manner.

In this paradigm, the terms "client" and "server" are purely situational. At one moment, an application could act as a client by calling an external service, while moments later, it may act as a service-provider when called by another application to perform a task.

When properly implemented, SOA promises to end the building and maintenance of point-to-point integrations. Using SOA, businesses will be able to generate new services in a flexible and agile way by combining existing logic and exposing it via reusable services.

SOA isn't new. In fact, technologies like CORBA and DCOM promised SOA in the 1990s. Both CORBA and DCOM introduced a high degree of complexity, and services based on these protocols were not truly interoperable by being vendor-dependent. Thankfully, today XML and Web services offer a truly standards based approach to developing services that can be consumed by many different types of applications—regardless of the application development technologies at work.

It's important to note that not every SOA is based on Web services. Rather, many different technologies, protocols and standards can be at play within an SOA. Web services are but one example of a standards-based implementation.

2.2 SOA Features

Competitive requirements for **greater business agility, flexibility to respond to change, reusability to cut costs and increase efficiency are driving businesses to SOA.** SOA holds the promise of a standards-based application infrastructure that encourages the development of modular, reusable components (or "services") that can support end-to-end business processes or composite applications, solve problems created by legacy applications that are hard to integrate.

In the past, with BPM being driven by the business and SOA is driven by IT, the market is now shifting where BPM in conjunction with SOA provides a platform that unites and aligns business and IT around business processes. With a core platform that enables automation of processes and visibility into processes that brings together both human workflow and business integration, the combination of BPM and SOA make end-to-end BPM possible.

2.2.1 Business considerations /aspects:

An organization needs to consider many different factors that are critical to SOA success before undertaking a project of this scope:

- Business case
- Best practices
- Organizational structure
- Guidelines and policies
- Process granularity
- Communication

and decide on an agreed roadmap within your company / organization. The temptation might be to immediately start building Web services. Many organizations take this route. If the scope of the project is only to create a number of Web services, this approach might work. But when designing and implementing an SOA, it is best not to just dive right in. The initial methods used to create, deploy and consume Web services usually establish a pattern of behavior, which might be difficult to change later. It is best to first establish practical guidelines and best practices. As the needs of every organization and their response to those needs are different, both business and technological aspects of SOA must be understood.

2.2.1.1 Process Driven Integration

Process driven integration is all about end-to-end BPM capabilities so business users have complete visibility into their business operations and can quickly respond to changes in their environment. BPM combines business process modeling and execution with advanced performance management capabilities to monitor and measure against operational targets.

The solution does not rely on creating new code, but entirely on models which are independent of the underlying systems. Business processes should be represented as services and then exposed as services so that different applications can consume and compose them in a loosely coupled manner.

Business process models create interactions between people, information models and service orchestration models to determine how information should be transported between systems. This provides a uniform and model-driven approach to designing, implementing, executing and supporting processes, information and data across all areas of the company.

To achieve business agility, you must break down legacy stovepipes into modular components that can be reused in multiple business processes. Applications are no longer built as tightly coupled, monolithic code but rather composed by assembling modular services. For example, a service might be a single software function such as CreditCheck or EmploymentCheck, that can be orchestrated as part of a larger business process a bank might use to approve a loan request. BPM is the end-to-end loan process and now other LOBs within the bank can reuse those software functions or services.

Each LOB can publish its processes as services and subscribe to other services, which in turn can become part of the process. A service registry/repository is where service compositions and orchestrations are stored and registered so users can not only find the services but also any processes that composed of those services.

2.2.2 Technical considerations / aspects:

2.2.2.1 Enterprise Service Bus (ESB)

The term ESB has been coined by analysts to describe a standards-based infrastructure solution that offers a core set of capabilities, which may include:

- Full XML and Web services support
- Load balancing
- Content-based routing
- Logging and message persistence
- Validation and transformation
- Security

The goal of an ESB is to provide the infrastructure necessary to offer value-added services to an enterprise SOA without forcing IT organizations to rely on tightly integrated or custom-built solutions. ZapThink, an analyst firm specializing in SOA, observed that an ESB isn't a solution to provide SOA in itself. But rather: "ESBs will combine with other integration approaches such as Business Process Modeling (BPM) and Service-Oriented Integration (SOI) to provide an optimal implementation of an SOA."

2.2.2.2 Role of the ESB

The ESB as an important component of the overall SOA. The implementation and use of ESB is much more than just technology. It is also an architectural approach. The ESB is intended for designing and assembling loosely coupled applications and services.

Mediation, defined in the dictionary as “negotiation to resolve differences conducted by some impartial party,” is a key concept in such systems. Just as in the dictionary definition, the ESB is well suited to resolve different assumptions about data format, data location and exchange protocols among applications or components in a distributed environment.

For example, consider an online store that employs the services of a credit card company, a shipping company and various suppliers of the goods sold. The basic process of selling goods, getting paid and shipping them may be fairly stable. But the details of which companies we deal with, what data formats they use and the precise Web addresses at which they can be reached are very likely to change on short notice. By building the application with an ESB, these differences can be resolved as they become apparent, without actual Java programming, or even recompilation and library deployment.

The ESB can't “negotiate” these differences in the dictionary sense of the word. But since the ESB builds on XML, Web services and Web standards, it uses skills learned in other domains (such as XPath, XSLT and XML schema construction). While ESB processes may fill the same role as Java or .NET code in an application, users can interact with them as they would other metadata in an enterprise, modifying values via GUIs and forms much as other online data are managed.

The true value of an ESB stems from its ability to allow users to manage the entire process of receiving SOAP requests and XML documents, manipulating them according to explicit rules and ultimately routing them to the intended destination. An ESB could facilitate these processes:

Conversion of incoming scanned documents: Incoming non-XML documents can be converted into a standard XML format as described by a specified schema definition, or incoming XML documents can be converted to different XML schema definitions.

Document validation: Often the first step in a typical business process would be to validate an incoming XML document structure against the defined XML schema to ensure the correctness of the document. The ESB can determine the document type received and can use the correct XML schema to do the necessary document validation checking.

Content validation: The next step often is to validate the actual data values via interaction with external systems. As long as an ESB is based on open standards, it can leverage open-source components extensively as part of its internal architecture. This results in an extremely flexible design time environment, allowing the user to create custom components, such as content validation routines, which can be made available from the design time palette. These components can be “dragged and dropped” into a sequence providing maximum reuse of existing components.

Data cleaning: Java classes, XSLT transformations, XSL-FO and custom transformations can be invoked to format data into the expected format. For example, inputs that are not well-formed XML could be run through a “tidy” program if a validation step fails.

Aggregation of multiple inputs: While an ESB process can be triggered by a single-request message, multiple inputs could be retrieved and consolidated. For example, we could build an RSS/Atom aggregator. The sequence could go through a number of steps to identify the basic type of data (RSS 0.91, 1.0, 2.0, Atom ...), perform the frequently necessary cleaning to make it well formed, transform it to a standard XML format or an HTML display format, and archive the cleaned data in an accessible location.

Persistence: Incoming XML documents can be stored to RDBMS or native XMLDB databases. This can happen at various stages of the business process and allow for the reuse of the XML documents in other sequences of business processes at a later stage. This caching mechanism can be used to improve performance and throughput.

External gateways: An ESB usually provides a number of external gateways or adapters allowing a business process to send XML documents and communications to customers, end users, external Web services or applications. These gateways allow a very flexible and effective means to communicate with the end user and external Web services as part of the business process. This could mean sending detailed messages or notifications to the end user or even sending complete XML documents. More importantly, it is the ability to orchestrate various Web services via the SOAP gateway that allows the ESB to fulfill an important role in the overall SOA.

Transformation: A transformation of the XML document could be invoked at any time during the business process. This is very helpful when the end user/customer requires the data to be in a specific format.

Content-based routing: The contents of an XML document, in process, can be evaluated. Based on this evaluation, content decisions can be made as to whether the document should be stored or to which customer it should be routed.

Replication: The ESB allows for documents to be replicated within a business process. These replicated documents could be complete copies of the original XML document or subsets of specific portions of the original XML document.

Aggregation: Various XML document instances could be combined into a single XML document instance. The ESB provides the ability to design rules for how this aggregation process should work. This is extremely useful when orchestrating various Web services. Each of the Web services invoked will reply with a proper SOAP response document and, through the aggregation process, will be able to select the responses or selected parts of a particular response that should be used to build the final result.

2.2.2.3 Role of Semantic Integration

A semantic integration tool enables organizations to build enterprise views of information important to the business, such as customers, products or employees. An enterprise-level view aggregates and materializes data in real time from source systems, presents relevant information in the user's terminology and gives different perspectives into the same information. Here are some examples of how this capability can be used to describe aspects of a company's business:

Customer: Tower Research Group, for example, states that "developing an enterprise view of the customer is one of the primary requirements of a CRM-based sales and service strategy." Semantic integration can pull together information from an organization's CRM and ERP systems and other sources to present a complete and consistent real-time view of any customer. For a call center operator, that might include the customer's order history, a summary of critical support calls the customer has logged and any items they have returned for refund.

Products: An enterprise view of products might aggregate information from the product catalog, the sales and service system and return materials authorization information so the COO can see what the most "troublesome" products are. **Employees:** An enterprise view of employees might aggregate information from the human resources and payroll systems, the corporate travel system, the employee calendar system and the company's employee directory so everyone in the company can locate their colleagues, know the best way to contact them, who to contact in their absence, and so on.

2.3 SOA Benefits

Service-oriented Architecture is a development approach that focuses on leveraging the investments that organizations have already made in technology by providing the tooling required to expose both business logic and data in existing or new systems in a standardized way.

- No need to rip and replace existing systems: SOA allows the reuse of all types of existing applications (both packaged and custom built) with a new GUI
- SOA and Web services standards help to ease the pain of integration
- SOA can help businesses respond more quickly and cost-effectively to the changing market conditions: Increased agility and flexibility
- Possibility to source business processes (insourcing / outsourcing)

SOA places a layer on top of existing IT assets which allows existing IT assets to be turned into 'services' that can be reused in support of optimized business processes. The standards used make these 'services' available for reuse regardless of the original technology or platform that the application resides on

The greatest benefit of SOA is its infinite flexibility. The greatest drawback of SOA is also its infinite flexibility. As you move beyond initial proof-of-concept SOA and BPM projects, you quickly begin to build out hundreds of services that are used to support hundreds of processes. How do you know what processes are using what services? Or what underlying systems support each service? How do you find what services are available for reuse? Without a centralized repository of information, you cannot answer these types of questions easily, if at all. Clearly, this isn't a good scenario from a management or compliance perspective. CentraSite is the solution.

CentraSite helps you manage integration components within a Service-oriented Architecture. CentraSite is an open, next generation SOA registry and repository. It promotes greater collaboration between business and IT by uniting metadata from service-oriented integration products.

As an SOA repository, CentraSite manages SOA metadata assets, enabling maximum visibility and re-use of Web services components within and across organizations.

CentraSite also provides reporting on those metadata assets and supports the sharing of the assets in the context of an SOA.

CentraSite integration partners (otherwise known as CentraSite Community Sponsors) store services and SOA assets supporting the business processes and services in CentraSite. Ultimately, CentraSite becomes the magic decoder ring that shows how your process driven integration is built out. Questions such as "tell me what business processes will be affected if I change this system" or "describe all of the different applications that support the accident claim process" can be answered through the metadata stored in CentraSite.

CentraSite provides you with a sophisticated process management platform supported by Service-oriented Architecture products. CentraSite provides unparalleled governance and management to this flexible environment.

3 Implementing SOA

3.1 Implementation on the business side

3.1.1 Step 1: Establish an SOA Vision

The first step to creating an SOA is establishing a clear vision of what the SOA will be and what value it will provide. Too often, companies rush to implement an SOA without clearly identifying the business value or the ideal end-state. Once expectations are misaligned, the success of the overall SOA implementation can be jeopardized.

3.1.2 Step 2: Building an SOA Roadmap

When creating services, it is important to start with a services-design specification. By starting here, you can identify:

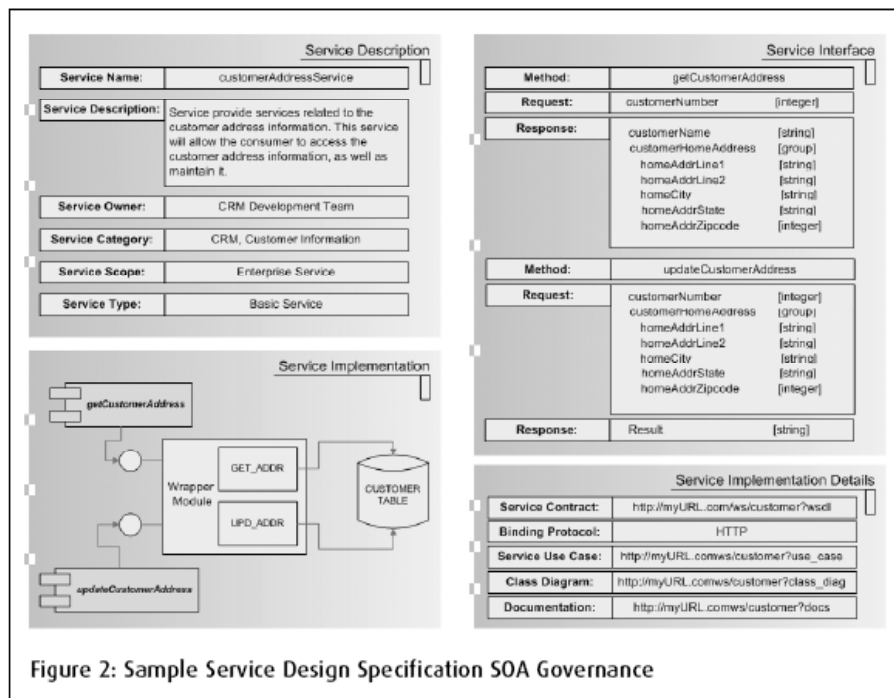
- Services needed
- What their interfaces should look like
- Scope of each service
- Granularity of each service

Often services are created, and no consideration is given to these factors. This has a direct impact on how usable the service will be to potential consumers. By following the services-design process, you can evaluate the requirements for different services and how they will be used. A good services-design process yields a highly reusable set of services.

Figure 2 below shows a service-design specification and its various aspects.

Generally, avoid these flawed approaches:

- Unnecessary re-engineering of existing applications
- Unnecessary extension or change to existing standards
- “Big-bang” approach to implementation rather than focusing on a tactical project



3.1.3 Step 3: Establish an SOA Methodology

Establishing a clear and consistent methodology is equally important to an SOA initiative. For instance, exposing a specific function in an application as a Web service may be possible. But, will it serve the needs of the overall SOA? Your IT organization needs to establish core principles surrounding the SOA then consider potential applications. It may be useful to build a set of best practices over a period of time. These practices will become the core of a proven methodology.

Other important questions to consider:

- What constitutes a service?
- Will you use a consistent toolset for building and deploying services?
- Will different groups be able to make independent implementation decisions?
- Would it make sense for the organization to establish a cross-functional SOA work group to develop and share best practices?
- How will the organization measure progress, and how will it judge end results as successful or unsuccessful?

3.1.4 Step 4: Link SOA with Key Business Initiatives

The vision of SOA promises a number of high-level business benefits, such as:

Efficiency: Business processes can be transformed from isolated silo and replicated processes into highly leveraged, shared services that cost less to maintain.

Responsiveness: Rapid adaptation and delivery of key business services can meet market demands for increased service levels to customers, employees and partners.

Adaptability: Changes can be made throughout the business with minimal complexity and effort, saving time and money.

These promises are certainly attractive. Yet what's most important is establishing the tangible value of an SOA by linking its initial implementation to a specific business initiative.

At the start of your project, you must ask this fundamental question: “How will the development and deployment of services help meet a stated business objective?”

A definitive answer to this question will help define initial steps related to purpose and goals.

3.1.5 Step 5: Create Architectural Blueprints

Next, you will need to define an architectural roadmap and blueprints. These blueprints are extremely useful in guiding the development teams by addressing specific design areas within their applications.

Blueprints may include:

- Common security model
- Service orchestration model
- Metadata management
- Process integration model
- Web services compliance model

Too many times development teams have to “reinvent the wheel” to solve a particular problem. By solving these problems once and documenting them by creating architectural blueprints and/or best practices, your team can achieve a certain measure of reuse. Establishing these blueprints is one area where reuse can have a direct effect on how services are created and on their reusability within the established SOA infrastructure.

3.1.6 Step 6: Assess Risks

As with any new IT initiative, an SOA implementation will introduce some level of risk into the organization. Ultimately, the organization needs to evaluate and assess the risks that will be assumed. Only then can you develop a clear plan for risk mitigation.

Security: Security is a critical component of any SOA implementation. Yet, often security is an afterthought. Security should be a key concern because the amount of XML documents will increase dramatically as your organization begins to introduce a larger number of Web services. The increase in XML traffic will raise questions about the vulnerability of the data that is shipped between endpoints. Security and risk are not always limited to outside exposure and may even play just as an important role internal to the organization.

You will need to verify some basic properties of these documents, such as:

- Where a particular document originated
- Who created the document
- How documents were received
- If documents have been tampered with or modified in any way
- Whether the content is accurate or correct
- Whether the content is safe
- How the documents are routed to mission-critical applications

What impacts would occur should applications receive malformed documents

What impacts would occur if a document contains an incorrect or false date

How the inflow and outflow of XML and Web services are tracked and audited across the enterprise

These are just some of the topics that must be addressed. We will discuss some of the options for security later in this document.

Interoperability: Interoperability is another key risk of an SOA. Despite the promise of Web services, organizations will discover that not all Web services are truly compatible. For example, Web services created in .NET and Java have slightly different implementations based on different interpretations of existing standards. Moreover, software products that function as “Web service adapters” for existing applications may not always interoperate with other Web services.

As your organization moves towards SOA implementation, you should monitor the efforts of the WS-I organization. The core mission of the WS-I is to be: “an open industry organization chartered to promote Web services interoperability across platforms, operating systems and programming languages.” The WS-I develops test suites such as the basic profile, which provides an interpretation of a core set of Web services specifications (such as SOAP, WSDL and UDDI) to promote interoperability among implementations. In practical terms, the WS-I produces test suites that can be run on Web services to ensure their adherence to the basic profile.

While such efforts help prevent instances of incompatibility in the future, you will need to consider how to handle instances in which services do not interoperate properly. How will you deal with these exceptions? What are the best practices for ensuring interoperability in the future?

Approaches to failure: As with any initiative, some approaches will be deemed overly risky or doomed to failure. As part of a risk

3.1.7 Step 7: Create an SOA Risk Mitigation Strategy

As change is introduced into the IT infrastructure, new risk is introduced as well. SOA is not a magic solution, no matter how easy technology vendors make it sound. There are some clear and recognizable risks. Identifying these risks upfront and creating a thorough mitigation strategy are important steps to limit risk. The value of following these steps should not be underestimated.

3.1.8 Step 8: Process-Driven Integration

Historically, Business Process Management (BPM) has not been easy to implement or integrate into the enterprise. BPM implementations have proven to be costly and complex undertakings, fraught with risk, partly due to the complex nature integrating business processes that span multiple applications. With every application interaction, the problem became more complex, since many different technologies were required to enable the interaction.

SOA provides an excellent foundation for implementing process-oriented integration scenarios that solve complex business process management and orchestration problems.

Today’s BPM solutions provide a rich toolset that facilitates modeling and engineering business processes while, at the same time, leveraging services exposed as part of the SOA. SOA is becoming a key enabler for BPM, providing a flexible architecture that’s easy to extend and also capable of adapting to changing requirements. Through the use of SOA, BPM processes can be shielded from the underlying changes within the services infrastructure.

Just as business processes can leverage the services within the enterprise, these same business processes also can be exposed as services to be consumed from within applications. The end result is that BPM becomes part of the SOA fabric, in which the business processes are viewed as nothing more than a new kind of service. If you are doing SOA, your design starting point should be what business processes you want to execute.

Implementation on the technical side

3.2.1 Step 1: SOA Enablement

When taking the first steps toward an SOA, we nearly always start with the familiar. Because legacy systems form the core of most mission-critical applications, we want to know how we can reuse these systems to leverage your existing investments. We then identify which parts of these systems need to be “exposed;” that is, which would provide the most value when transformed into services.

Next, we determine the types of information typically requested, so we can determine what functionality would solve the most pressing information demands. Only then can we determine the different approaches that can be followed to expose the identified functions as services.

Generally, there are three ways to integrate with these legacy systems:

- Session level
- Transaction level
- Data level

By exposing legacy applications as services through one of these approaches, we can make existing application functionality reusable by other applications. This saves time and money by taking advantage of existing resources.

3.2.1.1 Session Integration

Many z/OS applications are only accessible through terminal data streams, typically referred to as “green-screens,” and 3270 or 5250 terminals. This implies that applications are written in such a way that business logic and presentation interface are not cleanly separated through the use of callable routines.

There are a number of reasons not to modify or re-engineer such existing monolithic applications into modular versions. Many organizations, for example, lack the knowledge and technical skills required. For some organizations, the risks and costs associated with such a change could be too great. However, allowing these applications to participate in an SOA can be extremely valuable. In this case, an integration solution can provide flexible access to terminal sessions and allow them to participate in an SOA.

Session integration is the ability to intercept and interpret the screen information that is passed back and forth between a client and the server (for example, z/OS, AS/400 and UNIX). The terminal session or screen information can be packaged using different protocols, such as 3270, vt100 and 5250. These protocols describe the data related to the user interface and how that data should be interpreted and rendered by the receiving application (terminal emulator).

With session integration, it's possible to intercept the terminal emulation protocol data and render it in non-traditional ways. For example, the 3270 session data can be displayed in HTML format to be rendered within a Web browser.

In the past, this type of integration was called “screen scraping.” Screen scraping is an inefficient technique that is tightly coupled to the screen layouts. The moment a screen layout changes, the client application has a problem interpreting the data correctly. Session integration, on the other hand, can interpret the data received more efficiently via the emulation protocol. Rather than looking at the data in a fixed or positional manner, it actually parses the data and recognizes patterns (or data, fields, constants and screen identifiers) by locating them wherever they are within the data stream. This allows more flexibility in how the data can be manipulated and represented to the consuming application.

Session integration is often used to expose z/OS applications as Web services or XML documents. The use of either XML or Web services allows the consuming application a wider margin of flexibility in dealing with the session data received. The ability to abstract the consuming application from the actual implementation on z/OS is a powerful concept. It provides the means to expose existing z/OS applications as Web services to a new generation of applications—without requiring any changes to the existing z/OS applications.

Session integration clearly offers value in a number of scenarios. However, it does have some notable limitations. And because of them, there may be scenarios where session integration is not sufficient.

Here are a few limitations to keep in mind:

- Session integration is “one way” only. In other words, the z/OS applications can be exposed as services. But they cannot participate fully in an SOA by consuming other services that aren't on the z/OS platform.
- Session integration can expose and encapsulate only functionality that exists in the original application. There is no ability to extend the application with this approach.
- Session integration cannot access BATCH applications or processes. It can access only those that are exposed via a user interface or callable routine.

3.2.1.2 Transaction Integration

In other scenarios, applications may in fact be well structured with separate and distinct layers for data access, business logic and presentation. Ideally, the transactions that contain the business logic could be accessed as a Web service in an SOA. However, several challenges still remain. The application could be written in any number of languages, including Natural, COBOL or PL/1. In addition, the application may have been written to operate either in BATCH or online modes. Any viable solution will need to access a myriad of transactions, regardless of their language or mode of operation. It is equally critical this solution allow the application components to participate in an SOA without introducing change or risk into the environment.

Therefore, these legacy transactions need to be “wrapped” in such a way that they are callable as Web services without disrupting the original state of the application. Transaction integration refers to a style of integration in which existing transactions, such as BATCH programs or online CICS transactions on z/OS, can be accessed from distributed platforms. External applications should be able to call these transactions as methods or procedures without having to know that they may in fact reside on the z/OS platform.

Transaction integration allows this to happen by wrapping the existing transaction and exposing it as a service, whether it is a Web service, a Java object or a .NET object. To do this, a transaction integration solution needs to be able to handle the difference between the z/OS platform and the open-edition platforms, such as UNIX, Linux or Windows. For instance, on z/OS, data is encoded in EBCDIC, while on the open-edition platforms, data is encoded in ASCII. Transaction integration helps make this conversion transparent to the applications.

Here’s another example. Natural defines numeric data as numeric, unpacked decimal, packed decimal or integer formats. In a programming environment like .NET, numeric data can be defined as byte[], float, double, sbyte, short, int and decimal. Again, transaction integration helps perform the character translations between the respective platforms and also does automatic marshalling of the data, ensuring the data is properly converted between the various applications.

Transaction integration is also valuable because it offers the potential for “two-way” integration. Instead of only offering z/OS functionality to the outside world, transaction integration allows z/OS applications to consume external services without having to know they may be Web services. Just as a CICS transaction can be wrapped to look like a Web service, a Web service can be wrapped to look like a legacy transaction.

Despite the benefits of this approach, transaction integration has its own set of limitations that must be considered:

- Transaction integration assumes that existing applications are well structured with callable subprograms or procedures. If not, these applications must be re-engineered to use this approach.
- Depending on the nature of the original application, services created using transaction integration may be too fine-grained to be of value to an SOA.
- Organizations may want to get directly to data sources without having to call existing business logic. Transaction integration allows access only to applications, not data sources.

3.2.1.3 Data Integration

In some scenarios, organizations need to gain access to operational and transactional data residing on z/OS databases without going through business logic. Typically, new applications are developed outside of the original z/OS platform. These new-generation applications need to access and often times even update data stored within the z/OS databases.

In this case, you’re faced with two new challenges:

- You need to provide a standard of connectivity (via ODBC or JDBC) to the z/OS databases and data sources
- You must be able to encapsulate SQL statements or stored procedures as Web services

Data integration is the ability to provide a standard level of connectivity (typically ODBC or JDBC) to disparate data sources. This functionality is important for legacy databases that do not support SQL or provide ODBC connectivity natively. The ability to offer standard connectivity to these data sources allows data to be accessed in new ways and for new purposes. For example, an organization may have data in a database on z/OS but would like to make that data available to business intelligence tools. Data integration provides the facilities to make this possible.

Data integration has its own set of limitations. Some of these are:

- Accessing data directly may introduce risk by circumventing the business logic typically used to access it. This might not be desirable, since the integrity of the data of the z/OS database cannot be ensured when multiple different applications update the same data. It might be preferable to implement one set of business logic on the z/OS to enforce the correctness of the data stored within the database. This type of application should be exposed as a service that can be consumed by the new-generation applications.
- Data may be stored in a format that is unintelligible outside the context of the application built to access it.
- Accessing data directly may imply that business logic may be replicated on different platforms. This may defeat the principle of an SOA.

Service Characteristic	Screens and Session	Data	Transaction Business Logic
Granularity	Coarse e.g. a user interaction on many screens is turned into a Web service	Coarse, fine	Fine e.g. developers can pick a particular piece of code—not limited to a screen scope
Code Change	NO	NO	Yes/No Depending on the application design/modularity—might require reengineering
Extensible (Functionality)	Limited Restricted to original functionality	Dependent Extensibility depends on business logic	Enhance Coding of new functionality
Communication	One-way (z/OS can only provide services)	Two-way (z/OS can provide data and consume data)	Two-way (z/OS can provide services and consume services)

Figure 3: Comparison Between Integration Approaches

3.2.2 Step 2: Synthesis—Single Views and Service Orchestration

Once you’ve implemented Web services and taken the first steps to an SOA, you’ll typically find that XML and Web services introduce additional complexities into your environment. For instance:

- In many organizations, Web services first were employed at a departmental level. Each department may have taken a slightly different approach in its implementation. It is important to introduce enterprise standards without re-implementing existing Web services.
- Web services created from legacy transactions may be too fine-grained to be of value to other applications in the enterprise. It may be valuable to orchestrate several fine-grained Web services into a composite Web service that performs a defined business function.
- SOAP/XML documents may need to be transformed via XSLT.
- Web service requests may need to be routed to specific endpoints based on content.
- In a typical organization, information describing key items of interest, such as customer, product and financial condition, are scattered across multiple systems. It may be better to use a model-driven semantic-integration tool rather than creating and maintaining ad-hoc programmatic Web services to create single views of such items.

In short, you will need to ensure that your SOA remains agile, loosely coupled and maintainable. To do this, you'll need an infrastructure that offers value-added services to resolve conflicts and manage the SOA environment. For this purpose, a combination of service orchestration and semantic integration is needed.

3.2.2.1 Service Orchestration

Web services are created to address a variety of needs, and different Web services may perform business functions with different levels of granularity. For example, you may find it necessary to define Web services with fine-grained functionality, especially when such services are created to expose or wrapper functionality in legacy applications. Yet, such services are too fine-grained to be consumed directly by other parts of the SOA.

In such cases, you may need to execute multiple fine-grained Web services in sequence (with additional business logic inserted between the steps) and from them “compose” a new, more coarse-grained “orchestrated” service. This is service orchestration. It allows us to create a single, reusable and consistent definition of a business process that may be comprised of a number of lower-level Web services calls to the underlying systems. A composite Web service generated from such a process is referred to as an orchestrated service.

The use of Enterprise Services Bus (ESB) for service orchestration is needed.

3.2.2.2 Semantic Integration

When we pull data from multiple systems to create a single view, say, of a customer, we can identify three distinct levels of information integration:

Level 1 – technical integration: The first level, technical integration, is concerned with providing universal access to data regardless of the platform, database, operating system or location of the data. The familiar Internet and World Wide Web protocols and standards we use every day, such as TCP/IP and HTTP, are well suited to this task.

Level 2 – syntactic integration: Once we can access information in disparate systems, we have to agree on a common syntax for representing this information. In most integration scenarios today, this common syntax is provided by XML and HTML. Web services and service-oriented integration further define the level at which syntactic integration operates.

Level 3 – semantic integration: Once we have universal access to our information systems and a lingua franca for exchanging information between systems, we can focus on formally mapping the meaning of information between one system and another, resolving terminology, representation and even data accuracy or currency as we proceed.

Until recently, specialized tools for model-based semantic integration were lacking, and instead this task was handled by creating a set of foundational Web services that programmatically resolved these differences. Unfortunately, without a model-driven approach, creating and maintaining these foundational Web services is a chore in itself—especially when changes in one underlying system create a ripple effect across multiple Web services that access information contained within.

Semantic integration tools provide a model or extended dictionary of an enterprise's information. In addition to simple metadata, the model allows the creation of rules that govern how information is validated, rationalized and even combined with other information to allow inferencing.

Inferencing is a technique by which we can derive additional information by looking at the relationships between items. For example, if our hypothetical customer data included information about close family relationships, like parent and child relationship and sibling relationships, we could use rules to derive extended-family relationships between people such as “uncle-nephew” and “sister-in-law” and then ask questions such as: “What is the family relationship between these people?”

This model-based approach connects data from disparate data sources to a common business model that provided a unified, real-time view of information using dynamic data aggregation. The information model is capable of providing information about all connected data sources, while being flexible enough to accommodate changes. This provides the users an ability to receive different perspectives into the same information in their preferred terminology. Web services that provide information from such a model-driven approach are referred to as information services.

A semantic integrator product in the EII (Enterprise Information Integration) category provides modeling capabilities and also auto-generates information services.

3.3 SOA Governance

SOA governance is a key concept to address when starting your SOA initiative. Here's why: In large, complex SOA implementations, service management problems can arise after the first services have been deployed. The larger the number of services created and the more people involved, the more complicated the problem is likely to become.

SOA governance is about how the SOA efforts should be managed and controlled. It is about how the different groups, participants and services should operate within the larger SOA framework. Core considerations for SOA governance include:

Adhering to standards: It is best to create a common standard to be followed by every service provider. For example, in the case of SOAP, different binding protocols can be used. But the standard should dictate a preferred method.

Publishing and identifying services: Each service should be published in a common services repository. This repository can be used to discover the existence and location of services.

Monitoring, logging and tracking services: Due to security concerns, industry and governmental regulations, it is prudent to track which services are being used, how often and by whom. Keeping track of this information can be crucial for future reference and audit events.

Creating and enforcing contracts and service level agreements: Due to the loosely coupled nature of an SOA, it's a good idea to investigate new technological developments that can help specify and enforce basic service levels between consumers and producers. For example, a service such as trade date routine might be invoked by many different applications. Due to the financial nature of many of these consumers of the trade date routine, it would be reasonable to expect a minimum level of service. For instance, if the maximum expected response time for the trade date routine is 500 ms, any scenario in which response time exceeds this value might be indicative of a problem and should be dealt with accordingly. Unless a prior contract exists between the consumer and the trade date service itself, there would be no way to measure and enforce such a service level agreement.

Managing services and metadata related to each service: Initially you may create only a small number of services. But as the SOA becomes more widely accepted and understood within the organization, the adoption rate and corresponding creation of services can rise dramatically. You will need to manage these newly created services. To do this, you must determine: whether a particular service exists; whether there might be duplicates of the same services; and who owns a particular service. These issues can be addressed by using a central repository to store and publish existing services.

Secure services management: When you create viable SOA governance policies early on, your organization can cultivate good habits, and avoid less desired habits or approaches. The SOA initiative, in all likelihood, will be very visible within the organization. With sound governance policies and procedures in place, you can limit the risk of potential failure or conflict due to bad design, conflicting approaches, and poorly documented and publicized services.

Governance is important in the mainframe world and in many other areas within the IT infrastructure. The same holds true in the SOA world. (See next section on SOA security for details.)

3.4 SOA Security

XML and Web Services Security Challenges Application-level security is an important aspect of the overall security program. As new technologies and protocols become available and are implemented in production situations, you should give careful consideration to the overall security architecture. With the advent of Web services and XML, there are new ways to exchange data and interact with dynamic online services.

The requirement for security is directly related to the extent of the services provided, as well as the reputation of the service provider. It is crucial that the end user or business partner trust the service provider. Likewise, the service provider must establish a trust relationship with the end user or business partner. This trust relationship is crucial when conducting e-business.

With the introduction of XML and Web services, hackers are finding new ways to attack and exploit the corporation's internal systems. This occurs even though services may not be commercial in nature. XML and Web services use standard protocols such as HTTP and TCP/IP and technologies such as Web servers and applications servers. These protocols and technologies are the frequent target of hackers, and the cases of security breaches are numerous. It is, therefore, essential that you give particular care to how these technologies and protocols are implemented and used. You can mitigate many vulnerabilities, simply through the correct configuration of Web servers, application servers and other components.

3.4.1 Web Services Security Server

There are a number of commercial products and companies in the marketplace that focus exclusively on Web services security. The role of the Web services security server is to protect XML and Web service traffic according to a set of configurable security rules. It protects XML- and Web service-based applications by ensuring that only authorized users and applications are allowed to send data or connect to the services provided, that the appropriate level of encryption is applied and so on.

Web services security servers can be devices or software, and they will play an increasingly important role in the adoption and implementation of XML- and SOAP-based applications within the enterprise.

Here's a general overview of the features and functions of Web services security servers:

Policy-driven security: A Web services security server can be configured with appropriate policies to combine content filtering with identity-based rules. This allows both the control of user identities (accessing the XML and Web service-based applications) as well as the data they transmit. The server blocks unauthorized access, malicious attacks and malformed data.

Identity management: Web services security servers usually integrate or synchronize with the existing identity management infrastructure, such as LDAP, to perform authentication and authorization for XML- and Web service related traffic.

Real-time monitoring: Web services security servers also provide real-time alert and intrusion detection. This information can be useful when monitoring XML- and Web service-related traffic and performance. The servers log incoming traffic as well as any intrusion attempts. They also notify security personnel of attacks in progress so personnel can act appropriately. In addition, this information can be directed to a network management tool using SNMP.

Threat awareness: Web services security servers include support for a wide range of rules related to content filtering and traffic analysis. These rule sets are designed to protect the XML and Web service applications from malicious or malformed XML and SOAP data. In addition, the Web services security server plays an important part in guarding against:

- XML denial-of-service attacks
- Buffer-overflow attempts
- Malformed or invalid XML
- Unexpected MIME-Types in SOAP attachments
- Service scanning
- Brute-force "flooding" denial-of-service attacks

These security servers also can perform blacklisting functions and alert network firewalls further upstream by reporting the IP addresses of senders of malicious or malformed XML data.

Audit capability: Today's businesses are faced with a growing number of regulations, statutes and standards. To comply with these mandates, organizations have to provide audit capabilities for security and application monitoring.

Web services security servers can provide comprehensive auditing capabilities related to XML and Web services traffic. The information can be stored in a database, such as DB2 or Oracle, and typically details who accessed which services and how frequently. Each message sent and received can be logged, which allows for a complete audit trail of actions and corresponding results.

Standards-based: Web services security servers also provide support for a variety of security and XML-related standards. Technologies such as SSL, WS-Security and SAML are some of the important specifications and standards supported and implemented by many of these Web services security servers.

3.5 SOA Visibility and Governance with CentraSite

Many organizations start their SOA initiatives by creating ad hoc Web services. The existence of these Web services is usually known to select developers, and the information related to these Web services is usually shared in an informal way, such as e-mail. This approach usually works when a small number of Web services exists, and the group of consumers is small. In this case, there usually is no pressing need to manage the artifacts related to the Web services (such as WSDLs, XML schemas and XSLT), and there usually is an informal agreement about how to deal with upcoming changes in the service contracts.

As we have seen, SOA enablement, services orchestration and BPM are great tools for building a flexible and reusable IT architecture that enables a business to react quickly to new market conditions and customer expectations. This approach soon breaks down, however, when the adoption of services expands to dozens or hundreds of services in the organization.

Suddenly, the organization is faced with a number of new problems. For example, an organization may ask:

- Where do we go to determine which services exist within the organization?
- How do we determine whether the service contract is the most recent version?
- How do we determine the consumers of a particular service?
- How do we determine the potential impact of a change to a services contract?
- How are new services documented, and where do we publish a newly created service?
- How do we enable and maximize reuse to build composite applications and new processes and to shorten development time?
- How do we coordinate different teams of business analysts, architects and developers, and compel them to respect the norms set by the enterprise architecture?
- How do we govern and set policies over the deployed services and other components that will proliferate across the organization?

These considerations will have a direct impact on the reusability of existing services. This is precisely why a services repository can play a prominent role in the success of SOA projects. Organizations have found that to properly manage services within the SOA infrastructure, they have to have a central services repository, where all the services can be published and documented.

The services repository provides standardized interfaces, such as UDDI, through which service producers can publish their services. These repositories also allow service producers to document their services by providing additional metadata that help consumers find appropriate services via different classification and search mechanisms. Consumers can be assured that whenever they bind to a service, they will do so with the latest service contract. Likewise, service producers gain the ability to track how their services are used and by whom.

The resulting implementation of the services repository greatly increases the communication between the service producer and consumers, as well as the development teams. It is the central mechanism from which the various development teams can obtain the latest information regarding the service they need. Most IDEs today provide support for UDDI or provide plug-ins that allows developers to browse the services repository without having to leave their IDE environment. This ease of use inspires "integration" between the various parties involved in the SOA.

The usage of metadata to bring discipline and visibility to IT projects is nothing revolutionary. The data integration (ETL) and warehousing worlds have been leveraging metadata based on relational standards for years. But, things have been different for application integration, plagued by proprietary technologies until now. Unlike proprietary integration, SOA creates the opportunity to easily analyze metadata of models, processes, integration flows and Web services that are standards- and XML-based. The self-describing nature of XML and the ability to query XML files (XQuery/XPath) make metadata analysis feasible. XML-based metadata also offers the opportunity to represent a wealth of information extracted from different SOA management tools, such as XML firewalls and services-monitoring tools. You can then centralize this information into a central repository that holds the "DNA" of the entire SOA.

By bringing total visibility and governance over different SOA assets through a central point of control, organizations are more likely to reap the benefits of SOA and BPM. In concrete terms, this means that business analysts, who need to create new processes or integrate new systems in existing processes, can determine which systems can be reused via Web services. They also can determine what to expect from them in terms of information, logic provided and performance.

On the other side of the fence, a developer, who must update the Web service interface to comply with a new standard, can be aware of the processes that use the Web service and may be disrupted because of the change. Metadata analysis brings the needed benefits to well-planned and cross-functional integration: discipline, and horizontal and vertical collaboration.

4 SOA Case Studies

Case Study 1: Large North American Bank

Challenges:

With recent mergers and acquisitions, external compliance issues, globalization and consolidation, the banking industry is faced with accelerated expansion, process inefficiencies and increased competition. In addition, many banks are faced with complex, siloed technology that have been pieced together and patched up over the years, and which are now partially or even completely outdated. And to go back and re-engineer those legacy, siloed applications reduces a company's ability to cope with change as they seek to become more responsive to customers.

Solution:

SOA enables banks, which are facing pressure to improve profits on existing investments, to open up services to provide a better customer experience by offering flexibility and a faster response. These abilities are essential in a business environment where industry consolidation and outsourcing have combined with greater competition in recent years. Acquisitions and mergers mean that business processes are increasingly fragmented across a bank, making it difficult to adapt processes and be responsive.

To create business agility in the face of continual change, increased operational efficiency is required. These legacy stovepipes must be broken down into modular components that can be leveraged across business lines and units. Mortgage operations, card and payment operations, operations around retail banking must be modularized into core banking services.

Once these core modular services are created, business processes can be composed by assembling standard service building blocks which can be executed on request by any system, regardless of its operating system, programming language or geographic location.

A standards-based registry/repository plays a key role in development of business processes that integrate business services across different platforms, lines of business and geographies within organizations. You need a place to not only find the reusable business service you need across the entire bank in a shared infrastructure, but you also need a place to manage the associated metadata required as part of that business service (eg. policies, application integration models, business process descriptions, information models and Web service descriptions.). Lifecycle management is also important as business services change requiring change management and change notifications, as well as impact analysis.

Using BPM to monitor processes and repeat the BPM cycle, they were able to continuously improve core banking processes. As an overarching technology that spans all departments and functions, BPM was used strategically alongside middleware, messaging and CentraSite to become an essential layer in the IT architecture for the bank. CentraSite represented an open foundation for management and governance, providing unparalleled flexibility, visibility and control over the shared infrastructure.

Benefits:

By integrating multiple banking channels, banks can get a total view of their relationship with each customer, enabling them to improve their service. This means that customers can be better managed and can have a seamless experience of their interactions regardless of whether they communicate with the bank through the branch, call centre, internet or ATM.

While shortening development cycles, adopting flexibility and speed reduces operational costs, long-term objectives of efficiency, revenue-generation and customer growth are also key benefits. Automation will allow banks to devote more resources to customer retention and acquisition efforts. Organizations with the most efficient operations and most comprehensive data will be able to develop new products and services faster, speeding them to market for competitive advantage.

Case Study 2: US-based Insurance Company

Challenges:

With the number of insurance products being maintained over the years, including property and casualty, health and life insurance and annuities, agents were left to struggle through multiple systems and use their own judgement to try to find the right steps to come up with answers. The company's strategy of offering a highly diversified product line had created a raft of specialized systems that weren't necessarily linked together, preventing call center reps from providing quick answers to customers' questions. In addition, the insurance customers were expecting convenient and consistent service from the multiple ways of access—eg. call-center, Web self-service and others.

The insurance company needed ready access to a complete view of customer information; however, this can be a challenge as information often resides on disparate systems where even if they did do technical integration, the data was in different formats. The company also supports both agency management systems and insurance administration systems, but the two platforms had not been connected, meaning information captured at policy quotation had to be rekeyed at policy issuance.

Solution:

SOA helped integrate the company's multiple systems—a mix of imaging systems, group policy systems, individual policy systems—to pull up everything an agent needed to answer a customer's question without transferring the call. Integrating information from many sources is required to create a seamless experience across different channels. Different screens can be displayed to show different views. For example, one part of the screen might show the actual image of the policy and coverage and an invoice that was sent to the client. Agents get real-time customer information along with customer insight and product advice in an integrated, single-screen view and one-click policy and customer updates to improve key call center metrics like handle time, call blockage and first call resolution.

There are also federal and state regulations that must be enforced. For example, if you're not the primary insured, there are certain things the call center rep can and cannot say to avoid privacy issues. So the information that is displayed is based on who the caller is.

Transmitting precious customer data around raises security issues, maintenance and management issues. Who manages the services? Do you allow the individual lines of insurance product lines manage their own components? How was the service provided?

Benefits:

Results showed that 84 percent of customer issues are resolved on the first call, up from just five percent before the project. They also reduced the number of transfers to second-tier agents by 60 percent and reduced call referrals to staff outside the call center (such as sales staff or underwriters) by 90 percent. The company has also been able to reduce policy-processing time from weeks to hours and has experienced a growth rate of 48 percent over the past 12 months, which attributes to speed in processing.

In solving customer service problems caused by disparate systems, the company wanted to make sure that quality problems didn't inhibit the 360-degree customer view. CentraSite was used to provide better visibility into the myriad of services being developed and deployed. The company went from not being able to measure such basic information as call time to now having dozens of detailed call measurements. In addition, call centers were able to see whether anything unusual was going on when SLAs for certain services were not being met.

Providing a broad range of data and better insight into customer information gives insurance companies a 360-view of the customer which can lead to better decision making, increased customer satisfaction and loyalty, as well as improved cross-selling. Because information flows through components, rather than within silos, call-center reps can recognize where another opportunity might be, the same way that Amazon.com makes recommendations based on previous purchases.

Case Study 3: Major Telco Service Provider

Challenges:

The Telecom industry is far more dynamic than retail or manufacturing. Its services, customer demands, networks and systems change in a matter of weeks, days or even hours.

Yet those dynamic changes are difficult to accommodate when IT systems have been around for many decades, some have used for inventory management for perhaps 80 years. Because carriers generally have a very IT- and network-centric view of their businesses, they focus more on enhancing their networks and fixing legacy problems rather than improving the inherent architecture.

Additionally, operators have grown in numerous ways through acquisitions and high-growth booms. Inflexibility is rampant, and with so many different systems there are multitudes of stovepipes that house vital customer data. That has made coordination across functional areas very difficult for top-to-bottom reengineering and management of business processes.

Solution:

An SOA makes fits into existing environments because it enables providers to piece together new offerings with those of third parties and integrate them quickly with their internal, mainframe-based billing, provisioning, and other support systems.

In addition, this service provider is wanted to explore new business models that would enable them to productize their networks for those pushing content or traffic, where different parameter and measures of the service could be priced.

Other than the operator's service inventory consisting of a handful of services, most of which have been around for decades, this carrier is now thinking about other content such as videos, music downloads or games. There will be hundreds of different products to manage where the operator might collect a piece of the revenue with copyright holders, for selling copyright material over its network or upsell and cross-sell opportunities through portals or self care applications.

The possibilities are infinite. Not only does is the service provider trying to embrace a retail model (eg. ring tones, hosted messaging, accounting, and other business services), but the underlying infrastructure must be adapted to be like successfully retail companies where investment in IT and process will be the only way to gain visibility of partners and customers. SMB broadband customers become more savvy and the Internet becomes more integral to their business they'll eventually start moving business processes online in an ASP model.

Also like many successful retailers, they rely on partners to be suppliers in a supply chain. For example, if you go into Dell's web site, Dell has visibility into all its partners' inventory.

Benefits:

As thousands of services evolve through suppliers and partners, management of Web services is critical for success. Services need to be managed to control who uses them, monitor where they are being used, and provide a compliance-related control mechanism. The ability to categorize services becomes important to help people find and connect the services required.

And once they are deployed, the issue of managing services becomes critical in maintaining service levels to consumers and managing changes that could break consuming applications. CentraSite plays a key role as all of their SOA assets as well as supplier assets are managed in CentraSite. Governance and life cycle management is important when suppliers need to update their web service. It's important to identify who is using that web service and how it will affect other related services.

CentraSite is a central SOA registry/repository, that gives you visibility and control of SOA assets as never before. Jointly developed by Fujitsu and Software AG and continuously enhanced by the CentraSite Community Sponsors, you can take advantage of improved business agility, reduced cost and a high return on investment.

UNITED STATES

Fujitsu Computer Systems Corporation

1250 East Arques Avenue
Sunnyvale, CA 94085, U.S.A.

Tel: (408) 746-6300

Fax: (408) 746-6360

Toll-free: (888) 248-9273

E-Mail: info@interstage.com

URL: www.fujitsu.com/interstage

EUROPE

Fujitsu Computer Systems Corporation

FEL: Fujitsu Europe Limited
Enterprise Software Division
Hayes Park Central, Hayes End Road,
Hayes UB4 8FE

Tel: +44 (0) 208 606 4902

E-mail: es_support@uk.fujitsu.com

Software AG, Inc.

North America Headquarters

11700 Plaza America Drive
Reston, Virginia 20190

T: 703-860-5050

T: 877-724-4965

www.softwareagusa.com

Software AG

Corporate Headquarters

Uhlandstraße 12.
D-64297 Darmstadt/Germany

T: +49-61 51-92-0

www.softwareag.com