An Analysis of Service Oriented Architectures

by

Rajiv Ramaratnam

B.S. Bangalore University, (1989)
M.S. University of Denver (1992)

Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Engineering and Management
At the
Massachusetts Institute of Technology

June 2007

Signature of Author

Rajiv Ramaratnam
System Design and Management Program
March 2007

Certified by

Michael A. Cusumano, Sloan Distinguished Professor of Management
Thesis Supervisor
Sloan School of Management

Certified by

Patrick Hale
Director
System Design & Management Program

Introduction

Corporations all across the globe and of various sizes rely on their IT systems for business processes, reduction of process lifecycle and management of resources. These systems house several applications for internal sales, purchasing, finance, HR and so on. In any such typical organization, IT systems are a heterogeneous mix of hardware, operating systems and applications. Many of these applications run on different operating systems like Windows systems, Linux and Unix systems, etc.

Oftentimes there is a need to consolidate data or access data from several such systems. The diversity among systems and applications makes these tasks difficult, time consuming and tedious.

Furthermore, there is also a need for synchronization of data across systems and applications to ensure that the data is accurate and up-to-date. The heterogeneous nature of systems and applications lead to high levels of redundancy of data, making data maintenance a huge overhead for organizations.

Today's organizations must also adapt to changes in environments both external and internal to them. Such changes could be changing market conditions, reorganization, change in business strategies within a company, addition or changing suppliers, partnerships, mergers and acquisitions and so on. There is also a growing need for integration across enterprise boundaries to facilitate fast and seamless collaboration between partners, customers and suppliers. All such needs entail changes to existing IT systems within an organization in a timely manner.

There is thus a growing need for integration of such systems for consolidated decision making, accurate, up-to-date system information, better performance and system monitoring. IT Systems must also be flexible to respond to changes in the environments of their organization.

Enterprise Application Integration is a process that aims to bring about such integration. The need for integration goes beyond the boundaries of an enterprise. Further, to successfully compete today, businesses need to be flexible. This means that their IT systems need to be able to keep pace with dynamic business conditions.

It is evident that any solution for multiple IT systems to integrate with each other and to provide flexibility, they must be able to communicate and coordinate activities in a standard way. For almost two decades, many companies have tried to use CORBA, DCOM and similar technologies but have had little success. None of these technologies, for many reasons have become global technologies.

The arrival of standards like HTTP and HTML helped linking together millions of humans across the internet but proved inadequate to link together computer systems. Moreover, internal and cross enterprise integration and coordination bring with them, security implications as both involve information exchange between organizational entities. As we will see later, the traditional methods of securing applications with firewalls prove inadequate for application security.

One insight that has come from failed attempts to consolidate and coordinate IT systems is that such efforts cannot be limited to IT alone. Decision making on how interdepartmental and inter-enterprise data must data must be exchanged must be made by leaders and opinion shapers at each level or division of the organization.

It is the goals of internal and external enterprise integration, flexibility of business processes, and enterprise data security that has led more and more organizations to adopt to Service Oriented Architectures (SOA).

The adoption, implementation and running of a SOA does not simply involve IT department heads to design and create a new architecture for the enterprise. It involves a holistic understanding of the nature of the entire enterprise, its business and internal processes, the corporate strategy for the enterprise, an understanding of the business processes of the enterprise, its partners, suppliers, subsidiaries, etc.

Such an undertaking is beyond the scope of a single department or division of the enterprise. The creation and running SOA architecture thus involves the coordination of all parts of the enterprise.

Service oriented architectures abstract business processes and do not abstract software functionality.

So, how can organizations benefit by adopting to SOA?

- A SOA can help an organization consolidate its data and eliminate and data redundancy. It can help leverage its existing assets like mainframe data and allow seamless data communication between disparate systems and applications. SOAs can also leverage existing applications through reuse. Besides the reuse of existing systems and their data, an SOA can also facilitate building of new functionality from existing systems in lesser time. Thus SOAs helps efficient building on existing IT infrastructure and leveraging IT investment of a corporation.

- With nimble and flexible SOA systems, businesses can adopt to changing market, business and regulatory conditions and help the organization deliver efficiently and effectively to market needs.

- By bringing together different parts of the enterprise to make decisions on the operation of the SOA, an organization can help IT personnel better understand the their business and the operations of each of its divisions. The divisions of a corporation, in turn, can also have a say on how the SOA system has to function.

- Such a coordination of efforts within an organization can thus enhance a firm's investment in IT. IT systems can thus cater optimally to the needs of all divisions within the organization and to the organization as a whole.
- IT systems can build new functionality relatively easily and can respond to new market or business environmental challenges.

We can thus define a SOA as follows:

A Service Oriented Architecture is an IT architectural style, typified by loosely coupled, invokable software functional units (services) to enable supporting the requirements of applications or persons. An SOA is language, platform.

It is often assumed that SOAs consists of only web services, but in reality SOAs may or may not use web services. The key to an SOA is a set of independent services with defined interfaces that can be called to perform tasks in a standard way. An SOA may be implemented in REST, RPC, DCOM, CORBA or Web Services.

Among these technologies, web services and REST are the most popular ones used to implement SOAs. The question of which is better is an ongoing debate among the proponents of the respective technology. By and large, most corporations implement their SOAs through web services.

As described above, Service Oriented Architectures are composed of loosely coupled invocable software functional components called **services**. The loosely coupled nature of services makes them independent. Thus any user of a service need not be concerned with how the service is implemented and must only be aware of how they operate and how they ought to be used.

Typically services abstract business functions of an organization or a business unit within an organization. This means that while designing and implementing a service, careful decisions must be made on the level of granularity for the service.

The functionality of services implemented within an SOA is typically invoked through an *interface* for the service that is typically placed in a well defined location. The interface of a service defines the parameters required to use functionality of a service and also defines the format the output of the service. The contributor of the service and its interface is usually called the *producer* and a user of the service is a *consumer.* The provider and consumer of a service are usually software applications or components of a software application.

The invokable nature of services enables consumers to be agnostic of the physical location of the service. The service could thus be local or remotely located to the consumer.

The service could be part of the same application as the consumer or be located remotely on the corporate intranet or even on a partner or supplier's system. This aspect of web services also facilitates removal of barriers to outsourcing of application maintenance for an organization.

Further the management and workings of a service fall under the responsibility of the system providing the service and not the consumer that invokes the service.

While there are many definitions for Service oriented architectures, OASIS an international non-profit consortium to drive the development, convergence and promotion of e-business standards has charted a reference model to "guide and foster the creation of specific service-oriented architectures, and to publish a reference model for SOA" . The reference model was approved as an OASIS Standard in October 2006[1].

The Oasis Reference Model provides an overview of SOAs and guidelines on how they must be implemented and how they ought to work. However, all the descriptions and examples of SOAs in the reference model are too broad and generic. Independent implementers of SOAs just based on this model are bound to come up with architectures incompatible with one another.

The language and platform independent of SOAs gives them enhanced flexibility in implementing services. However, to facilitate access to them and efficiently use their functionality, their interfaces must be well defined, well documented and adhere to certain standards.

Further, clear separation of business logic (interfaces) and implementation is a major imperative for services in an SOA. This is one of the reasons for web services to have gained popularity in most large corporations today.

Web Services

As described above, web services are the most popular of technologies used to implement SOAs. The Web Services Interoperability Organization (WS-I), is an organization composed of almost all major software vendors to establish and promote web services.

Unlike generic SOAs, web services, by definition must adhere to strict guidelines outlined in the Basic Profile 1.0[2], a standard specification from WS-I. WS-I aims to provide *interoperability* among software modules on different operating systems that have been built by different programming tools.

[1] http://en.wikipedia.org/wiki/OASIS_SOA_Reference_Model
[2] http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

Using a mechanism called UDDI, web services provide a standard way to locate, catalog, provision and deliver service functionality. UDDI thus allows easy maintenance and monitoring of services by clearly separating business logic and the underlying technology.

UDDI also encapsulates related functionality of services allowing consumers to effectively and efficiently use them. We will look at web services and UDDI in more detail in the third chapter, 'Technologies of Service Oriented Architectures'.

SOA Growth

While Gartner estimates that most application growth by 2010 will be in SOA, IDC predicts that spending in SOA will reach 33.8 Billion in 2010[3].

Summary

SOA is an IT architectural model that has gained widespread acceptance in industry today. It allows reuse of existing IT assets, quicker development of newer software, standardized access to IT functionality for users and applications within an organization or across corporate boundaries.

SOAs make businesses flexible and streamline their processes effectively via services. SOA architectures can quickly adapt to changing market conditions and can thus enable service providers(producers) quick access to more revenue. By facilitating reuse of existing functionality, they can reduce IT spending and risks by eliminating cycles of development and testing.

In the following chapters, we will look at the Historic evolution of SOA, the technologies used to build SOAs, (particularly web services), the technological benefits of SOAs and the security aspects of SOA. We will then look at the benefits SOAs provide to enterprises and the state of SOA adoption in industry at this time.

Next, we will build a business case for why enterprises must adopt SOA. We will then look at the risks associated with SOA adoption and how one must plan contingency measures associated with these risks.

We will then look at the future of SOA in industry and look at the numerous pathways that SOA technologies can take.

This thesis is intended as a manual for business leaders and architects considering adoption of SOAs in their organizations or enhancing existing SOAs in their firms to bring in more value from their IT investments.

[3] http://www.fool.com/investing/value/2006/09/18/soa-the-new-software-bazaar.aspx

History of Service Oriented Architectures

Tracing the origins of Service Oriented Architecture is a daunting task. Many factors contributed to the arrival of Service Oriented Architectures. These include, application development, on early computers, networking, the client server architectures, object oriented programming and component technologies, the standards wars, arrival of the internet, and advances in enabling technologies.

Networking

Networking as a discipline predates even the earliest computers and can be traced to the early telecommunication machines, and then to the calculation machines. The 60s saw major evolutions in networking. One was a linked network of teletypes and computers developed by ARPA in 1962. In 1964, timeshared systems to enable distributed users to access large computer systems were developed in Dartmouth and MIT. Kleinrock, Baran and Davis, each produced mechanisms for packet switching between computers during this time.

The first ARPA node went live at UCLA in 1969. UCLA, UCSB and the University of Utah were now a connected using 50 kbits/circuit. In 1978, the International Packet Switched service was formed by The British Post Office, Western Union International and Tymnet. This network grew across Europe, Australia and North America in the following years.

The IP protocol was developed by DARPA in the 1970s.While IP enabled transfer of information across LANS, it was not ideal for exchange of packets across computer systems. Thus a protocol to exchange data, TCP was also in development at DARPA. Kahn and Cerf, both employees of DARPA would soon develop a TCP/IP a common network protocol that used both TCP and IP. TCP/IP allowed work over preexisting networks, and thus allowed ease of growth.

The National Science Foundation (NSF) foundation of the United States would then produce the first TCP/IP network, NSFnet at the beginning of 1983. After the network was commercially opened in 1985, computers called gateways were used to connect other networks like Usenet, Bitnet and Janet to the NSFnet.

A large private network that allowed dial-up access then joined these networks in the 90s. This network then began to be called the internet.

Tim Bernes Lee created HTML, a mechanism that allowed publication and receipt of data in 1991 at CERN (The European Organization for Nuclear Research). Two years later CERN publicized the internet as the 'World Wide Web' and the first Mosaic internet browser.

The modern day internet is governed by joint contracts and by protocols about how data is to be exchanged over it. In addition to TCP and IP, it includes an **application protocol** to specify the format of data exchanged by applications.

Remote Procedure Calls

The evolution of the application protocol led to the need for Remote Procedure Calls. A Remote Procedure Call (RPC) is a protocol that allows a program on one computer to call a software module on the same computer or on another computer. It enabled a programmer to write the same code irrespective of whether the called module resided on the same machine as the calling module or if the called module resided on a different machine.

It was first described in 1976 in RFC 707. Early implementations of RPC were done by Xerox's 'Courier application in 1991 and a UNIX implementation on Appolo Computer's Network Computer System (NCS). The latter became the basic for the Distributed Computing model by OSF. RPC was subsequently adopted by Microsoft and used under the covers of its DCOM architecture. It would also be adopted in the evolution of CORBA by OMG. Unfortunately the evolution of multiple varieties of RPCs led to several incompatibilities among them.

Client Server Applications

In this paradigm, computing is spread across two entities the client and the server, typically on separate machines .Client Server communication mechanisms are typically established by RPCs. The server does the heavy computing like generating processing, storage and retrieval of data, etc. The client usually is used to display, format or edit data.

A typical client server communication model works as follows. The client sends a message to a remote or local server requesting it to execute a procedure. The client sends all the data required by the server as part of its requesting message to execute the procedure. The server processes data from the request and furnishes the client with a response message that contains results of the executed procedures. There are many protocols that can be used in the client server model. As described above, the incompatibility among RPC mechanisms forced clients and servers to be locked into a specific RPC technology. This often resulted in a 'tight-coupling' among servers and clients.

Object oriented Programming

Object oriented programming (OOP) is a programming technique that 'abstracts' models from the real world. By 'abstracting' real world entities as containers of information called objects OOPs can provide functionality such as modularity, inheritance, polymorphism and encapsulation.

Modularity is a cohesive grouping of procedures and data. Modularity enables *encapsulation* (data hiding) and thus hides design. This helps changing the internal implementation of a module without having it affect the external workings of the module. Modularity thus allows the clear separation of implementation from the interface of the module.

Inheritance is a generalization or an 'is-a' hierarchical relationship between objects. For instance, 'Stick_Shift_Car can be a generalization of an object Car'. This implies that Stick_Shift_Car is a specific case of the object 'car' and may inherit all the properties of the object 'Car'. Stick_Shift_Car is a derived object from the base object car. Stick_Shift_Car is an inheritance of the object-type car.

Polymorphism is the means of allowing derived objects respond to the methods of the same name as the parent object. For instance, a drive () method called on a Stick_Shift_Car object may operate differently from the drive () method on the base object 'Car'

The earliest traces of Object Oriented programming can be seen in the 60s with PDP 11 developed at MIT (1963) and Simula 67. In 1970, Xerox PARC developed Smalltalk an object oriented language that used objects and messages. In the 1990s, C++ an extension of C became immensely popular, helped by the growing popularity of Graphical User Interfaces.

Object Oriented Programming became even more popular, fuelled by the growing popularity of video games. Object oriented features were added to many existing languages like ADA, Lisp, Basic and Pascal.

Another evolution that arose from the growing popularity of object oriented methodology were design patterns, design by contract and modeling languages like UML.

In the 90s, many technologies evolved from traditional object oriented methodology, each attempting to use the strengths and eliminating the weaknesses of most object oriented technologies. Four of the most common component technologies are CORBA, COM/DCOM, Java and .NET

CORBA (Common Object Request Broker Architecture)

CORBA was one of the earliest attempts to solve the interoperability issue among heterogeneous systems. When the Object Management Group (OMG) introduced CORBA 2.0 in 1997, developers could use its standardized protocol and C++ and Java mapping to implement communication among heterogeneous applications. It was touted as "*a vendor-independent architecture and infrastructure that computer applications use to work together over networks.*[4]"

CORBA uses a protocol called IIOP (Internet Inter-ORB Protocol), an efficient protocol where a client can make a request to a server and then allowing the server to call back the client on the same connection.

A CORBA application is a software entity (object) and communicates with other objects through an OMG IDL, to a CORBA object request broker (ORB). A client object typically specifies the operation it wishes to perform on a server object through the IDL. Every object in CORBA has its own unique reference. CORBA objects written in different programming languages can communicate through the IDL. In distributed environments, a client can call a server on another machine, through its ORB. In this case the ORB enables requesting and sending of the information to another remote machine communicating with the ORB on the remote machine. The CORBA communication between a client and an object is shown below. [5]

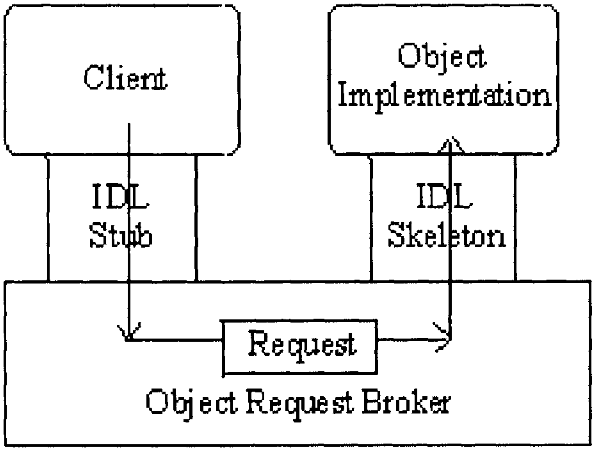


Figure 1: A request passing from client to object implementation

[4] http://www.omg.org/gettingstarted/corbafaq.htm
[5] Source: Object Management Group.

As the internet began exploding, however, CORBA did little to cooperate with popular web technologies like HTTP, Java and EJB. OMG tried to push for a component model for CORBA but this turned out a disaster and faced numerous compatibility problems with other technologies.

Further, Microsoft never adopted COBRA and was pushing its own DCOM Model. The late 90s also saw the emergence of XML as a silver bullet for the entire computing industry. Microsoft jumped on this bandwagon by collaborating with Developmentor and publishing SOAP, an internet protocol to provide interoperability. SOAP was passed to W3C and became a standard for distributed communication.

According to an article by Chris Britton and Peter Bye, Iformatit.com,

".. possibly the major reason that CORBA never took off was the rise of component technology [like Java and COM]."[6]

COM and DCOM

Early in 1991, Microsoft introduced OLE (Object Linking and Embedding), a technology to link and embed one application within another. Changes made in one application would automatically reflect in the other.

For instance, an excel document could either be linked to or embedded within a word document. OLE was messy and a large consumer of memory and other resources. With further improvements, Microsoft evolved OLE to form the basis of a programming model called Component Object Model (*COM*).

Programs wanting to use COM components could look it up in the system registry, a stable feature in Microsoft Windows Operating systems like Windows 95 and Windows 2000. The registry was a database modeled as tree structured directory used to look up COM components and system information on the respective computer housing the operating system. In the late 90s, a lot of applications were built with COM.

COM had the limitation in that it could only allow communication between applications within the same operating system. Microsoft thus provided DCOM as an extension to COM to enable lookup and use of COM objects on other Windows operating systems on a network by accessing the registry of another system.

The registry in Microsoft Windows systems however, became a major security threat. The registry contained application information about COM components that included the actual physical location of the application on the hard drive. Many viruses often exploited this vulnerability of windows systems causing great threats to such systems and their

[6] http://www.informit.com/articles/article.asp?p=345781&rl=1

stability. This was one of the reasons that Microsoft discontinued the use of the registry in their .NET framework, the successor to DCOM

Java

Sun presented Java, an object oriented programming language in the mid 90s. After years of unsolved programmer woes with languages like C and C++, Java was aimed to relieve the programmer from problems embedded in the nature of the languages themselves.

Java introduced the notion of a virtual machine (VM), a software abstraction of an actual machine. The Java applications themselves reside on and operate within the boundaries of this virtual machine. The VM is a pure software module with a strict set of specifications. Programs written with Java code are converted to Java Byte Code in a VM.

A Virtual Machine itself was a layer on top of the operating system of a computer. As the specifications of the VM are strict, the same Java program written on one operating system would run on another which runs another operating system, as this only entails moving the program across two VMs that both operated on Byte code. The code within the VM converts the Byte code to machine code specific to the respective operating system that houses the VM.

This approach provides numerous advantages like providing a Sandbox for Java applications and thus provides security from malicious code. The VM performs thorough checks on the application to ensure that that the security of the local machine of the VM is not compromised.

To make the job of programming easier as compared to other languages like C and C++, Java eliminated rarely used, hard to use features in its implementation and added simpler features to make programming and linking programs of an application less confusing. Developers of Java also made great strides in making Java more humanly readable, and facilitate maintenance of code written by several programmers.

Java does not incorporate many C++ features like user-defined operator overloading and multiple inheritance with classes. Java replaces this multiple inheritance among objects with the concept of an interface and thus eliminates confusion caused by polymorphism.

Being a pure object oriented language; Java prohibits the creation of functions that exist outside of a class. It facilitates distributed client-server programming through strict encapsulation, data hiding and abstraction making code cleaner and more portable.

In addition to checking for malicious code, the java compiler also rigorously checks the types of all data in the code, the legality of class-casts and checks if all arrays are within bounds. Java also eliminates one of the biggest bugbears in C++, pointers. C++ programmers have long been haunted by pointer errors and Java totally eliminates them.

The removal of pointers in Java eliminates the possibility of a program either intentionally or accidentally, to modify another object's memory.

Java has several other security features like the inability of Java programs called applets to make connections across the network. Information thus cannot be stolen and sent to a remote computer. Further, Java uses class loaders that ensure that programs and applets cannot access other parts of the system unless they are explicitly given permission to do so. The class loader also ensures that assumptions made about code placed in a namespace will always be true. Java is thus one of the most popular programs used for distributed computing used today. It provides numerous libraries and pre built packages to help the programmer.

.NET

Released first in 2002, Microsoft.NET was Microsoft's retaliation to the growing popularity of Java. It is an extension that can be added to Windows Operating Systems to enable them to run newer Windows programs. Like Java, it provided large libraries of code to target common issues. .Net programs run in a Common Runtime Environment (CLR), a feature, no doubt inspired by Java's virtual machine. Like the virtual machine the CLR provides libraries with security mechanisms, memory management and exception handling. The CLR allows programming in a number of languages like Visual Basic.NET, C#, C++.NET and so on.

XML (Extensible Markup Language)

The coming of HTML helped in the rapid growth of the Internet. Documents and images could be shared by individuals anywhere in the world. However, HTML had inherent flaws. One of them was its inability to differentiate between the presentation of data and the data itself. XML developed originally Sun Microsystems, thus emerged as a solution to this flaw and other limitations of HTML.

XML, in addition to transmitting information could also interpret the information. The greatest value added feature of XML was the fact that each information subject sent through XML, be they stationary bills, tennis scores or medical information have its own 'Vocabulary'. As described as in the article by John Markoff[7] in 2000, "XML designers began to glue together entire industries with their vocabularies". Microsoft too realized the value of XML and promoted its own technology built on XML, SOAP. XML became one of the key technologies that formed the basis of SOA. We will look at HTML in greater detail in the next chapter.

[7] http://www.nytimes.com/library/tech/00/06/biztech/technology/07mark.html

SOAP

SOAP was first conceived by Microsoft in 1998.When the first version of SOAP was introduced in 1998, there was no agreed upon methodology to describe XML structures or data.

The older systems assumed a Microsoft COM-like system for remote communication. Dave Winter, immediately released a XML-RPC specification as a superset of the SOAP specification. The next phase of SOAP came at the end of 1999 and illustrated how to model typed references and arrays in the W3C XML Schema type system.

The post 2001 Post SOAP era saw the stabilization of the XML schema. With this stabilization came the formation of a working group. SOAP today forms the backbone of web services, one of the most popular ways to build SOAs. In the next chapter we will look at the SOAP syntax in greater detail.

Recurring Themes

As we trace the history of computing and SOA specifically, we see the following themes re-emerge in different manifestations from time to time.

Loose coupling

Interfacing problems between discorporate software modules have plagued IT systems almost for ever. Loose coupling offers a way out of such woes.Loose coupling is a feature of two or more software modules by which each communicating module specifies a specific explicit way for other modules to interact with it. Further the modules make no assumption about how other modules of the coupling interact. With more diversity of platforms, operating systems and languages, it is clear that software modules need to be designed to allow loose coupling.

The best working definition for loose coupling can be found on ViewPoint.[8]

"Loosely coupled is an attribute of systems, referring to an approach to designing interfaces across modules to reduce the interdependencies across modules or components – in particular, reducing the risk that changes within one module will create unanticipated changes within other modules. This approach specifically seeks to increase flexibility in adding modules, replacing modules and changing operations within individual modules."

[8] http://www.johnhagel.com/view20021009.shtml

Encapsulation

This is an old concept that can be traced back to one of the main goals behind object oriented programming. Encapsulation of similar functionality within a software module makes it easier to understand and more intuitive for a programmer using that module. Encapsulation helps draw a clear line between implementation and the interface. Data hiding and abstraction are key benefits of abstraction. Encapsulation facilitates the process of loose coupling and helps keeps software systems modular.

Interoperability, Language and Platform independence

Interoperability is the ability of software and hardware systems to work in conjunction to accomplish tasks and to have the ability to share data. For interoperability between systems, they must be able to transmit and receive data in such a way that they understand each other's communication mechanism. They must also be able to comprehend the format used by one another to represent data. Interoperability is one of the key necessities that led to the development of the TCP/IP protocol and the ASCII character set.

Another key factor of interoperability is that terminology used in one system or software must not be misinterpreted on another system. Builders of interoperable systems must ensure that there is no ambiguity among commonly used terms (Author, composer, creator) and must offer documentation with strict definitions of each term.

Language independence is yet another recurring need that keeps returning from time to time. When interacting software modules are built using different programming languages, it is necessary that each module expose access to the functionality it provides in a language neutral way. The functionality provided by a module must also be usable by other modules written with either the same language or another programming language.

Reuse

Reuse is another recurring theme in the evolution of software systems. It refers to the use of a software entity built to accomplish a task be used with little or no modification to accomplish the same task or another task even in an entirely different context. System development done over time has proved that effective use of reuse reduces significant development and testing efforts in extending existing systems or building newer systems.

The motivation to reuse existing code has led vendors like Sun and Microsoft to provide their development tools with extensive pre built libraries that programmers can easily incorporate to build new components or applications.

Simplified deployment

Deployment is the process of installing one or more modules of a system at a certain site to make it operational at that site. When a system is composed of more than one component, deployment usually involves installation of the system component in a strict sequence to enable the components interact with each other in the correct fashion. Deployment also refers to updating software systems with newer versions, security packs and bug fixes.

As software applications grow in size and complexity, deployment becomes more complex and cumbersome. Improper deployment of systems can cause enormous wastage of effort and time and can delay or impeded the operation of critical systems.

The need for standardized deployment is thus another recurring theme in the evolution of software systems. The coming of the J2EE architecture is a major step towards such standardization.

With all J2EE compliant app servers like Tomcat, Weblogic and websphere, Java web applications are packaged as a war file (web archive files) and all other supporting complied java components are packaged as jar files.

Software built for the enterprise must be easily deployable in the shortest time frame and cause very little interruption of service. This is yet another recurring theme in IT today.

Security and Access Control

The need for security and access control has been revisited over and over in IT and is more important today than ever before. Security issues range from issues of data privacy protection, access control and malicious attacks on personal and enterprise systems.

These are just some of the recurring needs of software systems that led to the evolution of Service Oriented architectures.

Standardization

As IT infrastructure becomes more and more complex, marked by a mix of operating systems, devices from multiple vendors, the rise of Linux and Windows based systems, standardization of networking protocols, applications and software becomes more and more necessary.

Adhering to standards facilitates interoperability, reduces costs, simplifies corporate systems and promotes uniformity. Standardization reduces the need for specialized skills among employees of a corporation. It also facilitates transfer of information and proper coordination within and across corporate borders.

The Evolution of Service Oriented Architectures

Earlier in this chapter, we saw the evolution of popular technologies and emergent themes in software systems. It was clear that one vendor or architectural model could not address all needs across organizations. Some reasons for the lack of a comprehensive system were the limitations of existing technologies. The other reason for this limitation was political. The larger corporations have long been involved in a heated war about promoting their own software solutions and standards and in the process caused much of the heterogeneity that exists in enterprise toady

All these factors led to the evolution of Service Oriented Architectures OR SOA. The term "SOA" was first used in a 1996 research paper by Gartner analyst Yefim V. Natis.[9] SOA was thus an evolution in IT that happened to address all the recurring needs of computing, to consolidate the knowledge gained from previous technologies, leverage advantages of existing technologies and mitigate the risks of system malfunction in all systems. Service Oriented Architectures are not tied to any one technology but provide many options to build robust, interoperable systems. The next chapter will examine the technologies used to build SOAs and will focus on Web services in particular.

[9] http://objectwebcon06.objectweb.org/xwiki/bin/download/Main/DetailedSession/A-Trenaman-SOA.pdf

Technologies of Service Oriented Architectures

A Service Oriented Architecture is merely an architectural style characterized by loosely coupled components. It makes no dictates on the underlying technology used to implement it. Moreover, the reference model provided by OASIS is so general that it can be implemented by a variety of technologies.

Indeed SOAs can be built on RPCs Jini, CORBA or other technologies. However, most SOAs today are built through web services and REST. Web Service and REST have gained broad acceptance as they are standards based and allow for more interoperability. Web Service Specifications are mandated by WS-I an organization formed by most leading IT vendors. This allows for building SOAs without being tied to a certain vendor.

In this chapter we will examine some popular standard technologies used for implementing web services.

RPC

Remote procedure call (**RPC**) is a mechanism through which a program or software module on one computer can communicate and interact with a program on another computer. By using RPC, a programmer avoids the need to explicitly code the interaction between the modules. When used with object oriented technologies, RPC is often referred to as Remote Method Invocations (RMIs) or Remote Invocation. As mentioned in the previous chapter, many flavors of RPC exist today that are not entirely compatible among each other.

This means that software modules that implement RPCs need to be locked in with a certain vendor. Software modules using RPC to communicate to one another have to be tightly coupled with each other.

Systems that implement software communication through RPCs can easily violate the loose coupling aspect of Service Oriented Architectures. As RPCs often need a language specific interface, they are not a popular choice in building SOAs. The WS-I Basic Profile[10] disallows the use of RPCs to build web services.

Web Services

[10] http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

Web Services are among the most popular ways to implement an SOA. Though the term 'web service' can have several connotations, the WS-I Basic Profile[11] describes it to have a very specific definition:

"A web service is a software application that conforms to the Web Service Interoperability Organization's Basic Profile 1.0"[12]

Web Services are built of the following standard technologies:

- **XML**: XML forms the basis of all web service technologies. In fact, SOAP, WSDL and UDDI, the technologies to build web services are all XML-based. XML is the key protocol to describe the message payload.

- **HTTP**, the protocol of the internet is used by web services to send and receive messages. Often web services may use **HTTPS**, a more secure form of HTTP to send or receive confidential messages.

- **SOAP** is a standard XML-Based protocol to exchange XML data. Web Services exchange data using SOAP over HTTP.

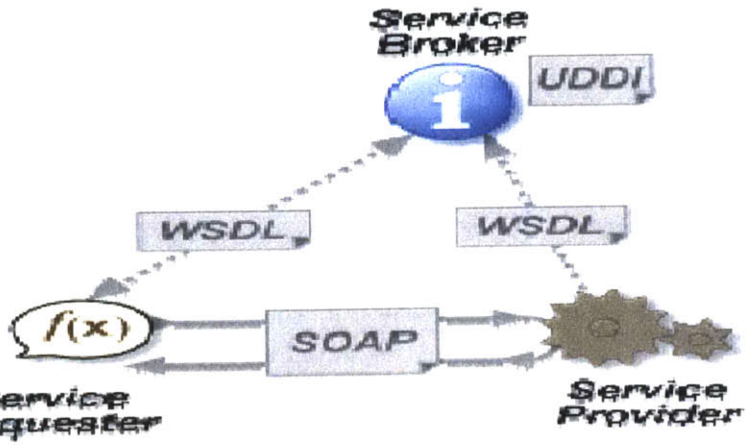
- **Web Service Description Language (WSDL)** is an XML-Based format used to describe a web service, the functionality the web service externally provides, the protocols supported by the web service and the format requirements of a message to communicate with the web service.

- **Universal Description, Discovery and Integration (UDDI)** is an XML-based registry database that an XML provider could use to describe the web service and allows users to locate a web service.

The figure[13] below shows the interactions among the component technologies of web services.

[11] http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

[13] Source: http://en.wikipedia.org/wiki/Image:Webservices.png



To summarize, XML is the protocol used to describe web service data, SOAP is the communication mechanism, WSDL is the language used to describe the functionality provided by the service, and UDDI is the registry to enable publishing of the web service.

The service provider publishes a description of a service on a UDDI registry provided by a service broker. A service requester looks up the description of a service through a UDDI registry. It uses the information provided in the registry to communicate with the service provider. The communication between the requester and the provider is done through SOAP messages.

XML (Extensible Markup Language)

XML is the basic building block of web services. All technologies used in web services are built on XML and the W3C XML Schema. XML can best be described as a meta-language, a language used to describe the format of other languages. In its essence, it is a set of rules through which one can describe XML Markup Languages.

The XML language describes a set of tags, generally paired and used to organize and describe text. The pair of tags are usually a start tag and an end tag. Any entity within these tags is an element. XML documents also may contain an optional declaration that declares the version of the document.

A Simple XML Example
The snippet below is an example of a simple XML document

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<!--
Author: Rajiv Ramaratnam
-->

<note>
```

```
<to>My daughter</to>
<from>Dad</from>
<heading>Happy Birthday</heading>
<body>Many happy returns!</body>
</note>
```

In the above example,

- The 1st line <?xml version="1.0" encoding="UTF-8" ?> is the xml declaration.

- Comments in the document are enclosed between the<!—and --> signs.
- 'My daughter' and 'Dad' represent elements.
- 'My daughter' is enclosed between start and end tags, <to> and </to> respectively.
- 'Dad'is enclosed between start and end tags, <from> and </from> respectively.

Hypertext Transfer Protocol (HTTP)

HTTP is a mechanism to convey information on the World Wide Web. It is the most common protocol to send and receive HTML pages on the internet. It is request/response protocol for Clients and Servers and is usually sent on TCP/IP. Typically data, pictures and control messages are sent and received using HTTP by browsers on computers.

Typically a HTTP Client establishes a connection through a Transmission Control Protocol (TCP) through a port with a remote server or host. The server responds by a status line indicating that the request was processed successfully or with an error message and some usually a supporting message of its own. It may also send along other information.

HTTP allows for request and response messages. Both of them are composed of an initial line, some optional header lines and then the body of the message. The body can contain text or a query or the results of a query.

There are three common HTTP requests, a GET command to request resource, a POST to put data on a server for processing or a HEAD to send to send additional data as part of a request. The initial line of a HTTP request is used to specify the type of request. The following is an example of a GET message:

```
GET /my_directory/my_subdir/file.html HTTP/1.0
```

The HEAD message is similar to the get message and uses the same format. Here is an example.

```
Header1: a_value, another_value
```

Below is a post message:

```
POST /my_directory/my_subdir/data.cgi HTTP/1.0
```

```
From: rajiv@ramaratnam.com
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

arg1=Test&arg2=flies
```

The response has 3 parts, separated by spaces. The first part specifies the HTTP version, a status number that gives the result of the request, and a line called the reason phrase describing the status code. Here is an example of the response.

HTTP/1.0 200 OK

The table below shows the most common responses.

Status Code	**Reason Phrase**	**Comments**
`200 OK`	The request succeeded, and the resulting resource (e.g. file or script output) is returned in the message body.	The request was successfully processed by the server.
`404 Not Found`	The requested resource doesn't exist	The server could not find the requested page.
`301 Moved Permanently`		The requested page has been permanently moved to another location.
`302 Moved Temporarily`		The requested page has been temporarily moved to another location.
`303 See Other` *(HTTP 1.1 only)*	The resource has moved to another URL (given by the `Location:` response header), and should be automatically retrieved by the client. This is often used by a CGI script to redirect the browser to an existing file.	This message tells the requester that the resource is elsewhere and that it can be accessed by the path specified by 'Location'

`500 Server Error`	An unexpected server error. The most common cause is a server-side script that has bad syntax, fails, or otherwise can't run correctly.	This usually means that the server encountered and error while trying to furnish the request.

For more information on the HTTP Protocol, see RFC 1945.[14]

SOAP

Originally an acronym for Simple Object Access Protocol, SOAP 1.1 is XML based standard protocol used for messaging. It is used primarily to communicate between two applications (A2A). It is also used in B2B communication.

A SOAP XML document is a SOAP message and is used as a payload for other network protocols. Web Services can use either one way to two way (Request/Response) messaging. SOAP messages are commonly exchanged via HTTP and sometimes via SMTP (Simple Mail Transfer Protocol) or FTP (File Transfer Protocol).

A SOAP message is similar to the traditional postal service. The root element or the main element of a SOAP XML document is a Envelope element. Envelope may contain an optional Header element and must have a Body element.

An XML document may also contain an XML namespace, a qualified name for an element or attribute. It provides the location that describes the document that describes the elements and attributes the namespace belongs to.

Here is an example of how a client might format a SOAP message requesting product information from a web service by a (no-existing)wholesale stationary store. The client needs to know which product corresponds with the ID 122212:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <getProductSpecifications
      xmlns="http://products.my_stationary_store.com/ws">
       <productID>122212</productID>
     </ getProductSpecifications >
   </soap:Body>
 </soap:Envelope>
```

[14] http://ftp.ics.uci.edu/pub/ietf/http/rfc1945.html

Here is a possible response to the above request.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     < getProductSpecificationsResponse
        xmlns="http:// products.my_stationary_store.com/ws">
       < getProductSpecificationsResult>
         <productName>Cross Classic</productName>
         <productID>122212</productID>
         <description>
          Black Cross Fountain Pen
          </description>
         <WholeSale_Price>30.50</WholeSale_Price >
         <Available>true</ Available >
       </ getProductSpecificationsResult >
     </ getProductSpecificationsResponse >
   </soap:Body>
 </soap:Envelope>
```

It describes a product, a Cross Pen in this example as a response to the information request on product ID 122212.

WSDL (Web Service Description Language)

As mentioned before, one of the key advantages to using Service Oriented Architectures is loose coupling. This means that external functionality provided by software components that are used by other software components must be well known. The functionality, data and protocols provided by such a component must be well documented and must be easily understood to all other systems and individuals using that component.

To describe the Web Service, a standard XML format, WSDL has been adopted by the Web Services Community. In essence, a WSDL file describes how a web service can be used. Technologies like JAX-RPC or other tools within IBM Websphere or Microsoft .NET can use a WSDL file to generate network stubs or interfaces using a WSDL. AS WSDL is highly modular, one can use its artifacts(XML tags) to describe more than one Web Service.

A WSDL file must include the following to a service:

- An interface definition outlining all the functions a web service provides, and the parameters needed by them and the format of the results each function provides after it has done its processing.
- The format and structure of the data shared by requesting and response messages.
- Information with regard to the transport protocol used for the messages
- Information on how to locate the service.

The WSDL document contains seven key elements: types, import message, portType, operations, binding and service. All of these are described in a definitions element, the root element of the WSDL document.

The **definitions**: This is the root level element of the entire WSDL document and envelopes the entire WSDL file.

The **types** element**:** All complex data types that are used within a WSDL document must be declared inside this element.

The **import** element: WSDL definitions from other WSDL files can be included using this element. This element lists all other WSDL files used in the respective WSDL file.

The message element: This element describes all the data types used in the message payload while accessing a service. The payload can have simple XML data types, complex data types and datatypes within the *types* element or types from files specified in the import element

The **portType** and **operations** elements: These elements are used to define the web service interface and all the external functions described within the web service.

The **binding** element: This element is used to map out the portType and operation elements to a certain protocol

The **service** element: This element maps the binding to an internet address.

The **documentation:** This element provides textual information on the WSDL document.

The following is an annotated example of a WSDL file[15].

```
<?xml version="1.0"?>

<!-- root element wsdl:definitions defines set of related services -->
<definitions name="StockQuote"
             targetNamespace="http://example.com/stockquote.wsdl"
             xmlns:tns="http://example.com/stockquote.wsdl"
             xmlns:xsd1="http://example.com/stockquote.xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

    <!-- wsdl:types encapsulates schema definitions of communication
types; here using xsd -->
    <wsdl:types>

       <!-- all type declarations are in a chunk of xsd -->
       <xsd:schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">


           <!-- xsd definition: TradePriceRequest [... tickerSymbol
string ...] -->
```

[15] Source : http://www.w3.org/2001/03/14-annotated-WSDL-examples.html#xsd_ns_example_com_stockquote
Used with permission of author, Eric Prud'hommeaux

```
           <xsd:element name="TradePriceRequest">
              <xsd:complexType>
                  <xsd:all>
                      <xsd:element name="tickerSymbol" type="string"/>
                  </xsd:all>
              </xsd:complexType>
           </xsd:element>


           <!-- xsd definition: TradePrice [... price float ...] -->
           <xsd:element name="TradePrice">
              xsd:<complexType>
                  <xsd:all>
                      <xsd:element name="price" type="float"/>
                  </xsd:all>
              </xsd:complexType>
           </xsd:element>

       </xsd:schema>
    </wsdl:types>
    <!-- request GetLastTradePriceInput is of type TradePriceRequest --
>
    <wsdl:message name="GetLastTradePriceInput">
        <wsdl:part name="body" element="xsdl:TradePriceRequest"/>
    </wsdl:message>

    <!-- request GetLastTradePriceOutput is of type TradePrice -->
    <wsdl:message name="GetLastTradePriceOutput">
        <wsdl:part name="body" element="xsdl:TradePrice"/>
    </wsdl:message>

    <!-- wsdl:portType describes messages in an operation -->
    <wsdl:portType name="StockQuotePortType">

    <!-- the value of wsdl:operation eludes me -->
        <wsdl:operation name="GetLastTradePrice">
           <wsdl:input message="tns:GetLastTradePriceInput"/>
           <wsdl:output message="tns:GetLastTradePriceOutput"/>
        </wsdl:operation>
    </wsdl:portType>

    <!-- wsdl:binding states a serialization protocol for this service
-->
    <wsdl:binding name="StockQuoteSoapBinding"
                  type="tns:StockQuotePortType">

        <!-- leverage off soap:binding document style @@@(no wsdl:foo
pointing at the soap binding) -->
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>

        <!-- semi-opaque container of network transport details classed
by soap:binding above @@@ -->
        <wsdl:operation name="GetLastTradePrice">

           <!-- again bind to SOAP? @@@ -->
```

```
            <soap:operation
soapAction="http://example.com/GetLastTradePrice"/>
            <!-- furthur specify that the messages in the wsdl:operation
"" use SOAP? @@@ -->
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>

    <!-- wsdl:service names a new service "StockQuoteService" -->
    <wsdl:service name="StockQuoteService">
        <wsdl:documentation>My first service</documentation>

        <!-- connect it to the binding "StockQuoteBinding" above -->
        <wsdl:port name="StockQuotePort"
                   binding="tns:StockQuoteBinding">

            <!-- give the binding an network address -->
            <soap:address location="http://example.com/stockquote"/>
        </wsdl:port>
    </wsdl:service>

</wsdl:definitions>
```

SOAP Message Embedded in HTTP Request

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body>
        <m:GetLastTradePrice xmlns:m="Some-URI">
            <m:tickerSymbol>DIS</m:tickerSymbol>
        </m:GetLastTradePrice>
    </soapenv:Body>
</soapenv:Envelope>
```

Here is a SOAP request message, based on the above WSDL file.

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
```

```
    <soapenv:Body>
        <m:GetLastTradePrice xmlns:m="Some-URI">
            <m:tickerSymbol>MY_STK</m:tickerSymbol>
        </m:GetLastTradePrice>
    </soapenv:Body>
</soapenv:Envelope>
```

Here is a SOAP response message for the above request, based on the above WSDL file.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: YYYYY

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body>
        <m:GetLastTradePriceResponse xmlns:m="My_URL">
            <m:price>1000.5</m:price>
        </m:GetLastTradePriceResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

UDDI (Universal Description, Discovery and Integration)

UDDI is the standard XML format for providing a data model to store information on organizations and their web services. UDDI facilitates identification and characterization about businesses and their Web Services. This is done through a registry service that catalogs enterprises and the web services offered by them. The UDDI registry is a database that contains information about organizations and the technical requirements needed to access their web services.

UDDI registries offer access to their data through SOAP based web services. A UDDI registry can be setup for internal use within an organization or externally in the web marketplace. The UDDI Business Registry is jointly run by Microsoft, IBM, NTT and SAP under an umbrella organization called the UBR Operators Council. Any organization can register its business in the registry and web services or peruse the registry for free.

The following XML structures are provided by UDDI for organizations to represent themselves and to detail the services they provide using UDDI.

The **businessEntity** element: This element provides information on an organization that is listing its services in the registry.

The **businessService** element: This element represents a service provided by an organization, The **bindingTemplate** element: This element maps and binds a web service to its tModel.

The **tModel** element is a abstraction of either a XML technology or a number system. The numbering system represented by a tModel may be US Tax IDs or a D-U-N-S numbers. The XML technology may be SOAP or WSDL or similar technology.

A **publisherAssertion** is a mapping of the the relationship between two business entities.

The following is an example[16] of a UDDI database entry.

```
“<serviceDetail generic="2.0" operator="SYSTINET" xmlns="urn:uddi-org:api_v2">
 <businessService businessKey="9a26b6e0-c15f-11d5-85a3-801eef208714"
         serviceKey="860eca90-c16d-11d5-85ad-801eef208714">
  <name xml:lang="en">DCAmail</name>
  <description xml:lang="en">Get credit assessment by email</description>
  <bindingTemplates>
   <bindingTemplate bindingKey="f9274a50-c16f-11d5-85ad-801eef208714"
           serviceKey="860eca90-c16d-11d5-85ad-801eef208714">
    <description xml:lang="en">The address to which you should send the name
      and address of your credit report target</description>
    <accessPoint URLType="mailto">mailto:DCAmail@democredit.bar</accessPoint>
    <tModelInstanceDetails>
     <tModelInstanceInfo
       tModelKey="uuid:93335d49-3efb-48a0-acea-ea102b60ddc6">
      <description xml:lang="en">The smtp protocol is used when sending
        information</description>
      <instanceDetails>
       <overviewDoc>
        <description xml:lang="en">Describes how to use this
          service</description>
        <overviewURL>http://www.creditdemo.bar/DCAmail/howto</overviewURL>
       </overviewDoc>
      </instanceDetails>
     </tModelInstanceInfo>
     <tModelInstanceInfo
       tModelKey="uuid:25ddf051-c164-11d5-85a6-801eef208714">
      <description xml:lang="en">The namespace in which our credit numbers
        are used.</description>
     </tModelInstanceInfo>
    </tModelInstanceDetails>
   </bindingTemplate>
```

[16] Source: http://209.85.165.104/search?q=cache:CfT_v1u6vs4J:sern.ucalgary.ca/courses/CPSC/547/W2002/slides/uddi-wsdl.ppt+uddi+simple+example&hl=en&gl=us&ct=clnk&cd=9

```
  </bindingTemplates>
  <categoryBag>
   <keyedReference keyName="Personal credit agencies"
            keyValue="841416"
            tModelKey="uuid:db77450d-9fa8-45d4-a7bc-04411d14e384"/>
   <keyedReference keyName="Credit agencies"
            keyValue="8414"
            tModelKey="uuid:db77450d-9fa8-45d4-a7bc-04411d14e384"/>
   <keyedReference keyName="Netherlands"
            keyValue="NL"
            tModelKey="uuid:4e49a8d6-d5a2-4fc2-93a0-0411d8d19e88"/>
   <keyedReference keyName="France"
            keyValue="FR"
            tModelKey="uuid:4e49a8d6-d5a2-4fc2-93a0-0411d8d19e88"/>
   <keyedReference keyName="Belgium"
            keyValue="BE"
            tModelKey="uuid:4e49a8d6-d5a2-4fc2-93a0-0411d8d19e88"/>
   <keyedReference keyName="Business credit agencies"
            keyValue="841417"
            tModelKey="uuid:db77450d-9fa8-45d4-a7bc-04411d14e384"/>
   <keyedReference keyName="Luxembourg"
            keyValue="LU"
            tModelKey="uuid:4e49a8d6-d5a2-4fc2-93a0-0411d8d19e88"/>
   <keyedReference keyName="Germany, Federal Republic of"
            keyValue="DE"
            tModelKey="uuid:4e49a8d6-d5a2-4fc2-93a0-0411d8d19e88"/>
  </categoryBag>
 </businessService>
</serviceDetail>
```

The UDDI Inquiry API

UDDI requires support for SOAP 1.1 over HTTP. It uses a request response paradigm. There are two sets of operations associated with the Inquiry API. They are the find operations and the get operations. One uses the find operations when one needs to search the registry and get operations when one needs to get the data associated with an entry. The find operations are listed below:

1. find_business finds businessEntity based on specific criteria.
2. find_relatedBusiness finds publishingAssertion that match the given criteria.
3. find_service find businessService that match the given find criteria.
4. find_binding finds bindingTemplate entries that match the given find criteria.
5. find_tModel finds tModel entries that match the given find criteria..

Example:

To search for a company, say Acme, we would create a query within a SOAP envelope[17]:

```
<find_business generic="1.0“ xmlns="urn:uddi-org:api">
    <name>Acme</name>

</find_business>
```

The response for this query could be:

```
<businessList generic="1.0"
 operator="Acme Corporation"
 truncated="false"
 xmlns="urn:uddi-org:api">
 <businessInfos>
  <businessInfo
    businessKey="0076C468-EB27-42E5-DC09-9955CFF462A3">
   <name>Acme Corporation</name>
    <description xml:lang="en">
      Produce food for toones to keep them in shape
    </description>
     <serviceInfos>
<serviceInfo
    businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
      serviceKey="1EFE1F71-2AF3-45FB-B788-09AF7FF151A4">
     <name>Web services for smart toons</name>
    </serviceInfo>
```

[17] Source: The original example was found at http://209.85.165.104/search?q=cache:CfT_v1u6vs4J:sern.ucalgary.ca/courses/CPSC/547/W2002/slides/uddi-wsdl.ppt+uddi+simple+example&hl=en&gl=us&ct=clnk&cd=9. It was changed slightly here.

```
   <serviceInfo
     businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
     serviceKey="8BF2F51F-8ED4-43FE-B665-38D8205D1333">
    <name>Electronic Toon Business Integration Services</name>
   </serviceInfo>
   <serviceInfo
     businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
     serviceKey="611C5867-384E-4FFD-B49C-28F93A7B4F9B">
    <name>Volume Licensing Select Program</name>
   </serviceInfo>
<serviceInfo
     businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
     serviceKey="A8E4999A-21A3-47FA-802E-EE50A88B266F">
    <name>UDDI Web Sites</name>
   </serviceInfo>
   </serviceInfos>
  </businessInfo>
 </businessInfos>
</businessList> “
```

The get operations are listed below:

1. get_businessDetail gets businessEntity entries.
2. get_businessDetailExt gets BusinessEntityExt entries.
3. get_serviceDetail gets BusinessService entries.
4. get_BindingDetail gets bindingTemplate entries.
5. get_tModelDetails gets tModel entries.

The UDDI Publishing API

Organizations use the publishing API to add, change or delete information in a UDDI registry. The UDDI publishing API requires HTTPS (HTTP over SSL 3.0) for secure authentication. There are four basic types of UDDI operations. These include, authorization operations, save operations, delete operations and get operations. UDDI also specifies fault messages when something goes amiss.

Authorization Operations

Before one can publish to a UDDI registry, she has to enroll with a UDDI operator. For this, one must go to an operator website, fill out a form about the organization she represents and choose a user-id and password.

Once enrollment is complete the operator can use the user-id and password to login into the registry and add, modify or delete entries in the UDDI registry. There are two authorization operations:

1. get_authToken logs one into the registry.
2. discard_authToken logs one out of the registry.

Save Operations

Save operations enable one add or update information in the registry. They include:

1. save_business, used to add or update one or more businessEntity entries.
2. save_service, used to add or update one or more businessService entries.
3. save_binding, used to add or update one or more bindingTemplate entries.
4. save_tModel, used to add or update one or more tModel entries.
5. add_publisherAssertion, to add one or more publisherAssertion entries.
6. set_publisherAssertion, to update one or more publisherAssertion entries.

Delete Operations

Delete operations enable deletion of information from the UDDI registry.

1. delete_business, deletes one or more businessEntity entries.
2. delete_service, deletes one or more businessService entries.
3. delete_binding, deletes one or more bindingTemplate entries.
4. delete_tModel, hides one or more tModel entries.
5. delete_publisherAssertion, deletes one or more publisherAssertion entries.

Get Operations

Get operations are used to get summary data about data structures published by the client.

1. get_assertionStatusReport gets a summary of public assertion entries.
2. get_publisherAssertion gets a list of publisherAssertion entries.RE
3. get_registeredInfo gets an abbreviated list of businessEntity and tModel entries.

Representational State Transfer (REST) SOAs

In SOAs implemented with web services, services abstract business processes. Web services are agnostic to the resources the processes act upon. A service could be exposed as a service or vice versa. REST SOAs, as we will see, use the concept of a resource.

REST was first introduced by Roy Fielding to describe the web architecture. It is an alternative to using the typical Web Service model described above. **REST** or **RESTfull Web services** mimic the HTTP protocol by using the typical HTTP commands, GET, DELETE, POST and PUT. Unlike web services that focus on sending and receiving resources, REST focuses on interacting with resources and changing their state. A REST resource is represented through a URI (Uniform Resource Locator). Services and Service providers in REST are resources themselves, but consumers may or may not be resources.

With rest, all interfaces are limited to the four HTTP commands GET, DELETE, POST and PUT.

- The HTTP GET command is used to by a consumer to get the representation of a resource.
- The HTTP DELETE command deletes a resource representation
- THE HTTP POST command could with update an existing representation of a resource or create a new representation of a resource.
- The HTTP put command is used to create a new representation of a resource.

The messages sent must be in an XML schema language like XMLSchema, W3C or RELAXNG. One can encode a simple message using URL encoding.

Resources in REST are identified using URIs. REST does not use RPCs to send messages. All messages are in the context of a resource. A typical rest message may be to change a state within a resource.

REST directly works with the data structure of the resource, unlike web services that would call a function to work on a resource or data. For instance, a REST resource type Stock, may have a number of exposed URIs. If http://my_potfolio/mystock is one such URI, one would use the PUT or POST command to create an entry and then use GET to retrieve the record. REST would make an XML call to a resource called StockID. A web service in turn would use a SOAP request message getStock Price with a parameter StockID.

REST is agnostic of the underlying representation of the data and can support any number of tiers of an application, distributed or client server model. To be REST compliant, an application must support a HTTP GET operation. REST by nature is simple, asynchronous and event driven.

REST defines no standards in representing a Resource. Its messaging model is a simple combination of XML and HTML. The lack of standards in REST is often seen as a drawback in the SOA community. However, some believe that its simplicity will make REST a more common choice in building SOAs. Web services in turn have the disadvantage of being more complex but are more robust than REST applications.

The Amazon Affiliates network, the Linkshare network and the Google AdSense network are all examples of REST services. In all three of the above mentioned, a website builder needs to add a snippet of code provided by any of them into their website.

The website becomes part of a larger network of affiliates and can display products and goods from this network. The website can then get paid for all the sales generated for the respective network through its referrals.

The argument for and against using REST, as a choice instead of using web services is an ongoing discussion in the SOA community. As web services allow flexibility to build interfaces, REST proponents believe, they, web services do not lend themselves to scalability, reliability and the ability to cache intermediate results in the midst of an interaction. It is hard to determine if a web service SOAP request represents the fetching or the updating of a resource.

The asynchronous nature of REST and its inherent simplicity makes it less than ideal for reliability of service, transactions and selective encryption of data in a message. Web services provide the ability to carry messages via several transport protocols in their path, thanks to the header architecture of SOAP messages. REST on the other hand is tied to HTTP.

There are advantages and disadvantages of both types of architecture but builders of SOAs could possibly build an SOA with a combination of both.

Summary

In this chapter we have covered some of the key technologies used to implement SOAs. Web services are the most popular technologies used to build SOAs but REST implementations are also gaining popularity. It is not the intent of this chapter to provide a comprehensive description of these technologies but to provide a basic understanding of how SOAs can be implemented. In the next chapter we will look at the technical benefits of implementing SOAs.

Technical Benefits of SOA

In the last chapter, we gained an understanding of some of the important technologies to implement a SOA. In this chapter we will look at the advantages offered by SOA from an development perspective. If used properly, SOAs can be a boon to architects and developers of IT systems. The following are some of the technical and architecting benefits of SOA:

Reuse

As mentioned earlier, the loose coupling aspect of SOA software modules promotes reuse. Reuse is beneficial to organizations in the following ways:

- By eliminating of redundancy in code, reuse facilitates code maintenance.
- By eliminating duplication of efforts, reuse eliminates cycles of development and QA. As SOA is designed to facilitate coordination across different parts of the organization, organizational departments can work together to avoid duplication of programming efforts and thus be beneficial to each other and to the organization as a whole.
- As reused software modules have been tested before, it helps corporations reduce the risk of adding bugs during new development.
- When existing reusable components are well documented and cataloged in an organization, developers can quickly build new functionality from pre-built components.
- All of the above points mean that the organization becomes quicker in adding new robust functionality and large cost savings in software development.

Reusable software must be heavily tested for robustness and all possible boundary conditions that could cause it to behave abnormally.

It must be stressed that the workings of reusable software components must be documented in detail. This means that the documentation must warn the potential for the software module to malfunction if it does so, under extreme conditions.

Further, when components are reused, they must be treated as black boxes and that no assumption is made about their underlying implementation. It must also be ensured that software components work as documented.

Composite Applications

As mentioned before, for components of systems to be loosely coupled, the functionality of the components must be made explicit. SOA systems draw a clear line between implementation and interface. As we saw in the previous chapter, SOA systems built with web services make their interfaces explicit through a WSDL file.

This allows components built using heterogeneous tools and residing on different platforms to interact among each other. SOA systems help build composite applications through the use of interfaces.

One can harness the power of old existing software components with newer components which may be built with newer tools to provide the customer a composite application that incorporates the functionality of both, old and new systems.

Much of the woes that are experienced while trying to integrate applications, implementation and testing are eliminated through the use of interfaces. The use of interfaces also allows building of systems with components irrespective of where they are located.

Flexibility and Agility

The loose coupling aspect if SOA, wherein the invoker of a service is agnostic about the location or the underlying implementation of the service introduces a high level of flexibility of choice. To create a service component, one has the freedom of choosing from a variety of technologies and platforms. One could build a composite system by choosing the best tool and the most optimal platform while building a system component.

This benefit allows choosing the best platform or geographic location for a service. Computing intensive algorithms can be deployed where price/performance is optimal for that activity. One can also optimize other system parameters like network traffic to make a system robust and reliable.

Encapsulation

Traditional object oriented programming focused more on the encapsulation of objects based on related functionality. However, SOA prescribes the use of encapsulation around business processes.

A system encapsulating based on business processes can offer functionality from components is disparate parts of an organization, different geographical locations or from more than one organization. Organizations can thus exhibit greater efficiency when they present their business processes through SOA.

Standardization

As the invocation interface to SOA takes a standard form, a programmer creating a new service need not have to understand the technologies used by underlying services. This reduces the skill requirements in the development organization, leading to cost reductions.

Interoperability

SOA allows interoperability from several dimensions. From a software systems standpoint, interoperability addresses the ability for different system to work together. SOA implementations using REST and web services allow this type of interoperability. When users are thrown into the mix, interoperability means that in addition to having the ability for disparate systems to work together, there is no likelihood data of the system is misinterpreted or not understood either by a system or user.

Interoperability within an Organization

As we saw above, SOA allows interoperability between heterogeneous software components built on different platforms, using different technologies to communicate with one another without too much programming overhead or developmental costs.

Interoperability across Organizations

The location transparency aspect of SOA components allows the partners, suppliers or customers of an organization use the services if the organization without being aware of where these services are located and how these services are implemented. Conversely, an organization may use its partners' service by being agnostic about its implementation. This improves process efficiency and promotes automation across the enterprise value chain.

Semantic Interoperability

Semantic interoperability means that data can be understood unambiguously by users and software and that it can be processed in a meaningful way. For example, in the software world, words like 'Platform' and 'System' often have many interpretations.

Systems using SOA can be built in such a manner that they only work with data they can comprehend. The loosely coupled nature of SOA also allows scaling the architecture to accommodate more diverse data and to handle it in an efficient and flexible manner.

UDDI, a facet of web services provides a taxonomy of terms concepts, categories and keywords to help creators of web services to describe their lingo unambiguously.

Incremental Deployment

Loose coupling of components allows for incremental deployment. In large corporate systems it is commonplace to replace components of the system with newer components or with different components.

If the interfaces of components are well defined, then newer or different systems that implement and exhibit the same functionality can be used easily to replace the existing component without hindering the operation of the system for large periods of time. Time and frustration spent in fixing integration problems are also minimized if careful consideration is paid to backward compatibility in the development of newer systems.

The newer component could be at a totally different location and use different underlying technology from the older component.

Abstraction

Abstraction in SOA involves hiding the complexities in implementation of business processes through simple, clear easy-to understand interfaces. SOA focuses on sharing of business processes and knowledge both within and outside the boundaries of an organization.

When the service is delivered with the end customer (consumer) in mind, it exhibits an interface that the consumer of the service can easily understand. Users of a service need not understand all the intricacies of the workings of the service and can simply use the service to extract its benefits.

Real Time Information

One of the key requirements of organizations today is the need for the most up-to-date information for the runnings of their business processes. Systems that provide real time data must go beyond periodic retrieval of data from different places but must also present the data to consumers in a less ambiguous way.

Factors that influence real time include:

Business process awareness: SOA helps define service as abstractions based on business awareness. It can present data in the most relevant format to a consumer

Data availability: SOA can pull data from disparate sources, be they databases or files. A well-architected SOA exhibits data from multiple sources in the same uniform way.

Work flow and Transaction Processing: By exposing, abstracting and combining business processes SOA applications can orchestrate simple and composite workflows built from multiple components and also incorporate transaction processing to provide accurate data with minimal human intervention

Analysis and Reporting: While not directly lending themselves for analysis or reporting, SOA applications can be used to aggregate and report the data they processes in an optimal fashion.

Advantages to Developer

Implementing SOA offers several advantages to developers. The biggest of these is that SOA allows developers to focus on business logic and implement functionality using a platform and a tool of their expertise without getting bogged down with logistics issues required by the systems or the data. As services have a well-defined interface that is independent of implementation, developers are free to use any programming language they feel comfortable with. Using multiple vendor products can help developers take advantage of unique features provided by each of them in building the system.

The loosely coupled aspects of SOAs make developers more efficient. If a certain component needs to be upgraded, the developer can make build a newer version of the component with the required enhancements and bring it online with minimal disruption to the overall operation of the system. This makes the organization as a whole more responsive to the changes in its environment.

Another side effect is that having a well-defined contract enforces good programming and design practices. Developers can save time and reduce the risk of adding bugs by using pre-built reusable services as building blocks. They can create flexible loosely coupled systems that can adapt to changing business conditions. Using deployed, documented, tested and internally certified components saves, time and effort to reinvent the wheel. Time saved can be spent on other efforts of the business.

A combined industry effort called SOA Blueprints, helps organizations build applications using SOA. SOA Blueprints creates a common vocabulary to discuss SOA in an architecturally unambiguous way allowing implementations to be built from a variety of tools and technologies.

SOA Security

The security of corporate IT systems today is of more paramount importance than ever before. They must safeguard their sensitive data and also protect their systems from damage. These threats can come from both, within and outside the corporation.

Threats to corporate systems include attacks from malicious users who may try to steal or access confidential information, or bring down a corporate system. Corporations must also guard against espionage, virus and worm attacks through their own intranet and the internet.

For most corporations today, the primary security gatekeeper was the firewall. A user trying to access information from a company must first authenticate herself through a firewall. However, firewalls have several limitations:

It is very restrictive with its all-or nothing policy. A user who authenticates successfully can access any resource within the corporate network. This means than resources need to be separately protected from unauthorized users even some who have authenticated with the firewall successfully. In short, it is hard to enforce a corporate security policy through a firewall.

With SOA systems the entity that has to access a resource or data i.e. the consumer, need not be a human and thus such systems need a different authentication system.

Thirdly, with most SOA systems being implemented through web services, the primary protocol use to send and receive messages is SOAP. By design, SOAP message is designed to move across firewalls, as they invariably travel on HTTP and HTTPS. Hackers could thus send malicious messages and cause damage to corporate systems or steal data on them. There is thus a need for more sophisticated security systems In this chapter, we will look at some security issues that are inherent in SOA systems and some of the technologies available to address these needs.

Security Issues

The following are some of the security issues that must be addressed in a SOA environment.

Authentication and Authorization

From an SOA system context, authentication is the process by which the system verifies the identity of an entity (user or application) that wants to access and use the system. As mentioned earlier, older systems used VPNs and firewalls to authorize users. Typically users authenticated themselves with the system by providing a username and a password. This mechanism that is clearly inadequate for SOA systems.

Authorization is the process by which system verifies if the entity wishing the use of a system resource has the appropriate permissions to use it. To authorize use of resources within a large company, the company must have a sophisticated authorization mechanism in place.

As we saw above, a VPN or firewall is not adequate for SOA authentication. VPNs and Firewalls cannot verify the identity of the consumer who could be a human or an application.

Further, authorization mechanisms provided for HTTP web applications are also not adequate for SOA authorization. Corporations need a mechanism to assign policies to consumers of SOA services. The loosely coupled nature of web services mandates the necessity of a corporate security policy that is both strict and flexible. Security becomes even more of a problem when SOA builders seek to reuse applications or build composite applications.

Confidentiality and Integrity

SOA systems may send and receive messages within and across corporate boundaries. SOA confidentiality is vitally important and entails that messages that are sent to, from and within the SOA system remain private between the sender and the receiver of the message and that no one is snooping on the messages sent. Malicious hackers have often stolen or eaves dropped on confidential data sent with and across company borders. SOA systems thus need a message scheme that ensures the privacy of both senders and receivers.

Integrity of SOA systems entails that messages send and received within and across SOA systems have not been tampered with while on transit. Malicious hackers could also pose as the receiver or sender of a message and send false messages to a receiver or accept unauthorized messages from a sender. Thus there is a need for a reliable encryption mechanism that ensures that messages send from SOA systems can be verified for their authenticity and that the sender can verify the authenticity of the receiver of its message

Privacy and Integrity are two more aspects of SOA security. Privacy is the ability to conduct business without eaves dropping and integrity is the ability to have confidence that messages are not tampered with during transit. Hackers often listen in on messages and modify messages during transit for reasons of mischief or crime.

In addition to using HTTPS to encrypt the messages sent and received by SOAs, corporations may use other security mechanism like signatures, keys, certificates to ensure privacy and integrity.

Denial of Service Attacks

Corporate systems today are also prone to Flooding or Denial of Service (DoS) attacks. In both cases, a corporate system receives a tirade of messages and then crashes due to the overload. In some cases, this may happen inadvertently as more users and applications are accessing the website. In other cases, the intent of the sender is to cause harm. The sender could continuously send messages to deliberately bring it down.

When corporate systems are brought down due to flooding or DoS attacks, their inability to operate could mean diminished ability to conduct business which in turn could mean lesser revenue for the company.

Corporations must build security to guard against DoS to their SOA systems by monitoring the network traffic for abnormal loads. Their security systems must provide the ability of administrators to take precautionary or emergency measures in a timely manner. They must also ensure that their systems have enough failover and round robin ability to ensure that their systems can handle a large load of incoming messages.

Auditing and Reporting

Auditing and Reporting are key to running of a reliable SOA system. An SOA system must have an audit log to give administrators a feel for the performance of the system and security. Logs help in diagnosing problems and also provide a clear sense of accountability in case a problem emerges and helps ensure that the problem and its threat are eliminated from the system.

A good SOA security system must be able to diagnose and report on problems or concerns and alert the responsible parties to take diagnostic, precautionary or remedial measures.

Security Policy and Provisioning

As mentioned earlier, to allow access to resources, SOA systems must have a mechanism to authenticate and authorize users and applications that interact with them. This involves identifying the user or application and then establishing if the user or application could conduct the required operation (accessing or modifying a resource) on the system.

An ideal security system must provide the ability to centrally manage the SOA system within a corporation, establish policies for users and applications and allow the ability to delegate administration of sub systems of the SOA to a certain department or part of an organization.

Ron Schmelzer and Jason Bloomberg suggest that [18], "*enterprises must institute policies that apply to their entire enterprise network (including participants invited from outside) and administer that security in a tiered, hierarchical fashion with a centralized root administrator. Departments or other organizational groups may then have their own administrators, but those administrators must in turn be administered by a more senior admin at a higher level within the enterprise.*"

Federation and Identity Management

Identities of entities, be they users or applications have to be managed across interdepartmental and inter-organizational boundaries, ie: across domains. This means establishment of security authentication and authorization polices, managing user profiles across multiple user stores which are often heterogeneous and the ability to handle multiple networking protocols and compatibility across heterogeneous security environments. A security platform for an SOA system must be able to tackle all these issues and also support SOA standards. In addition to supporting SAML , WS Security and other newer technologies, a security platform must also support existing standards like ODBC, JDBC, LDAP and coordinate security functions among them

Governance

The governance of SOA systems requires the need for integrity in their data towards internal standards, external regulations and corporate stakeholders. In a sense in encompasses almost all the issues outlined above and more: security policy, auditing and reporting measures.

A well governed SOA must be able to provide data that can withstand the scrutiny of an audit .The next chapter had been dedicated to SOA governance.

An Overview of SOA Security Solutions

We saw that conventional security mechanisms used for web application security are clearly adequate for SOA. SOA systems implemented through web services use SOAP which my design can traverse firewalls through HTTP and HTTPS. SOAP transmission is also vulnerable to buffer overflows and thus is susceptible to denial of service attacks. Further the loosely coupled nature of SOA enables not only user-system interaction but interaction between distributed applications located in different systems, which could be housed in different corporations. It thus becomes a challenge to enforce security policies

[18] http://expertanswercenter.techtarget.com/eac/expertAnswer/0,295208,sid63_gci1022062,00.html

which must encompass not only users but applications and traversing inter and intra corporate boundaries. As we saw in the earlier chapters, there are multiple ways to implement SOA and thus security solutions for SOA must be able to handle a variety of messaging mechanisms and network protocols

If security policies are too tight, they could defeat the loose coupling advantages of SOA systems. Leniency on the other hand, can result in compromising security of a firm. While a single silver bullet is impossible for SOA, both corporate management and IT must coordinate efforts in enforcing a balanced corporate security policy and be aware of the available technologies to built such a security infrastructure.

The complex security requirements of SOA and specifically Web Services are driving a new set of security standards as follows:

Web Services Federation Language: This is a mechanism that allows subject identities with policies shared across web services. Web services can 'broker' trust among different security zones using WS Federation Language.

Security Assertion Markup Language(SAML): SAML from Oasis allows exchange of authentication and authorization information between disparate systems and protocols. It is has an XML based format thorough which a subject can send security information through assertions. A subject could be a human or an application. Assertions are information with regard to authentication and authorization policies and decisions made by the subject. SAML is more advantages than HTTP cookies, since SAML tokens, unlike cookies, can work across DNS domains and enforce Single-Sign-on in heterogeneous systems. They also allow for federation of subject identities across multiple business environments.

WS Security Language: Earlier in this chapter, we saw how the privacy ntegrity and confidentiality of messages in SOAs can be compromised through snooping or the addition of bogus messages. WS-Security from Oasis offers an XML based signature mechanism, encryption specifications for messages with message digests and XML signatures to ensure the authenticity, privacy and integrity of SOAP messages sent and received by web services

WS Trust Language: The WS Trust language extends the security language to define the format for issuing of security tokens that contain subject credentials, etc. within domains that have trust relationships

WS Security Policy is a framework that allows web services to declare their constraints and requirements. These constraints and assertions are declared through assertions

WS Security Conversation Language: This is a layer of security built over WS-Security and WS-Policy. It is aimed at authentication and the sharing of security contexts. It helps enable secure communication.

XML Encryption: This is a W3C extension which can be used to encrypt XML data. The result of the encryption is an XML element that contains the data encrypted between XML tags defined in the encryption specification.

XML Signature: This is a combined specification from W3C and IEFT. It defines the XML rules and syntax for digital signatures in XML. It helps ensure message and subject authentication and helps verify the integrity of XML messages exchanged between systems.

Extensible Access Control Markup Language (XACML) presented by OASIS is another XML specification to define authorization and access policies to specify subjects, rights and conditions.

These standards are in varying levels of completion and as a result, security providers have implemented proprietary solutions that are independent or in varying levels of conclusion. There are also proprietary solutions for authentication and authorization of users from multiple directories, user provisioning, single sign on, etc.

Eric Koch's blog on IT toolbox[19], mentions that SOA security requirements are often complex and that the standards described above are still immature to handle them. According to Koch, the requirements are further complicated with the need for security of legacy systems. He proposes that there is a firm argument case to consider commercial security packages as the infrastructure for identity management within the SOA.

Companies like Oblix and Netegrity(now part of Computer Associates) provide several commercial packages to perform functions like single sign on between LDAP and SQL databases, supporting SAML, WS Security, etc and user profiling, centralized security policy administration and synchronization of data across directories. These packages are built to minimize system degradation while performing security functions

Summary

Planning and preparing for SOA security threats goes far beyond awareness of possible threats. Architects and analysts must consistently prepare for possible ways that the security of the enterprise could be compromised. In a chapter on risks, I outline some oversights that organizations can make with respect to security. Managers must be aware of these pitfalls and use all the expertise and creativity in the organization to avoid them.

[19] http://blogs.ittoolbox.com/eai/business/archives/soa-security-architecture-11431

SOA Governance

SOAs can integrate multiple departments within a corporation, multiple partnering companies which have heterogeneous applications, platforms, operating systems and data stores to coordinate and synchronize activities and share data.

In addition to leveraging existing corporate assets, SOAs must be flexible, scalable robust and responsive to the business environment, reconfigurable and secure. For all of this to happen and to ensure maintenance of SOA efficiency their management and execution must be monitored and governed.

SOA governance thus deals with continuous monitoring, decision making and the enforcement of policies and procedures to SOA systems. It also helps assign accountability for the operation of a certain facet of the SOA to a certain individual or group.

According to an article by Phillip Windley in InfoWorld[20], the big payoff of business agility and other benefits from SOA have more to do with policies and procedures than the quality of code.

Jason Bloomberg in an article on ZapThink[21] rightly asserts that SOA goes beyond the he governance of SOA and is the governance of IT. For companies that rely on IT for their operation SOA governance is the governance of their corporate policies. The process of governance must be considered from the early stages of the design process of the SOA.

Rules and Policies

During the design process, all parties involved in creating, using or benefiting from the SOA must be brought together to initially establish rules and policies. Rules and policies must be then documented and enforced. Over the lifetime of the SOA the rules and polices must be carefully assed for their utility value and modified or eliminated as necessary. Governance and policy establishment is an ongoing process.

During the design process, the exercise of establishing policy helps create crisp interfaces and delineation of process boundaries across subsystems that make the SOA.

A white paper[22] on SOA governance on weblayers.com gives a good example of SOA polices for a medical organization:

[20] http://www.infoworld.com/article/06/01/19/73698_04FEsoagov_1.html

[22] http://www.weblayers.com/gcn/whitepapers/Introduction_to_SOA_Governance.pdf

"Policies might start at the business level with regulatory compliance issues:
• Patient name and contact information may not be transmitted as clear text
• Each message must carry information to uniquely identify the message source, destination, timestamp, and transaction ID, to meet mandatory archiving requirements

This is could be followed by specific policies for information security like:

• Messages must contain an authorization token
• Password element lengths must be at least 6 characters long and contain both numbers and letters
• Every operation message must be uniquely identified and digitally signed
There might be lower-level technical policies that ensure architectural strength:
• Do not use RPC Encoded web service operations
• Do not use Solicit-Response style of operations
• Do not use XML 'anyAttribute' wildcards

If any one of these three sets of policies is not followed, the impact on company operations and bottom line can be lethal."

Governance and architecture go together in the effectiveness of an SOA. Architects who build systems for the organization are more aware of the constraints within the system. Operators and users of the system are more aware of how to use the system and thus systems work predictably and the risk of malfunction due to operator error is also eliminated. Testers also are more aware of how to test and debug the SOA. This also facilitates the process of extending the system to incorporate new functionality.

SOA governance not only impacts the SOA system and its usage, but enforces companywide adherence to policies forcing paradigm shift in behavior of employees. As employees work together and jointly create corporate policies, procedures and their rational are better understood by all parties.

Tolerance

In another article titled 'A degree of tolerance for SOA'[23] Windley states that governance comes with its own issue. One need is for modularized IT systems with unprecedented agility but it is impossible to foresee every variation or every possible future scenario.

The answer to this, Windley suggests is to have tolerance - carefully laid out rules and formal change management. In some instances, the creativity of web developers must not be stifled by an architectural committee. In their book, Design rules[24], Carliss Baldwin and Kim Clark warn that governance that is overly strict on developers can drive value out of SOA.

[23] http://www.infoworld.com/article/06/01/19/73741_04FEsoagovloose_1.html
[24] http://www.amazon.com/exec/obidos/ASIN/0262024667/infoworldcom-20

As mentioned, the design phase of an SOA must bring together all parties of an organization who are in some way associated with it. It is also important that meetings for design are properly moderated and a compromise is reached with any prickly policy issue.

Tracking, Reviewing and Improving Polices

Governance involves constant tracking, reviewing and improving of existing conditions.

These include reviews of:

Policies: Policies must be carefully reviewed. How are they enforced? Are they being followed? Are they in line with the strategic objectives of the company? If new policies are to be enforced, how will they impact the respective system module and rest of the system? Are there exceptions to a certain policy?

Functionality: Proper governance mandates ensuring that services implement functionality in line with their core competencies. Are the interfaces in line with the functionality they deliver?

Security: Are the services offered by the SOA secure? Can the security of the system be compromised? Are security policies and lines of controls adequate to avoid security breaches? Are there process flows that could have security holes into the system?

Dependencies within the System: How will a change made to a unit or subsystem of the SOA impact other parts of the system? How will it affect business operations of the domain or organization?

Consolidation of Functionality: When functionality within a service or service set has to be consolidated, the consolidation must be invisible to the end consumer who interacts with the service interface. It must be ensured that new or existing policies are in place when the consolidated services are operational.

Consolidation of Processes: When processes need to be consolidated within a service or service set, it must be ensured that backward compatibility is preserved if the individual processes are still used. Such implementations must thus conform to enterprise policies with respect to design, development, implementation, testing, deployment and maintenance.

Domain vs. Departmental Governance

Conventional IT management divided up management of IT systems by the departments within an organization. SOA governance is divided among organizational groups based on a business process called domains

Individuals within a domain group are responsible for maintaining services that deliver functionality based on the business process. They are responsible for the maintenance and monitoring of the services design of the service interfaces and formatting of results from computations performed by their services. Those intending to use the services of a domain must collaborate with members of the domain group and create a written agreement that elaborates on the use of the domain.

The maintenance of a domain requires the following functions

Domain Leaders or Leader who is responsible for deciding the key high level purpose of the domain, the functions it is responsible for and its future directions. This person or persons works closely with upper management and carefully monitiors the domain operation from the perspective of the entire company. This person or group must constantly seek to maximize the returns from their SOA subsystem.

Domain Business Experts who identify functions that the domain can perform through its services that will benefit the company as a whole. These individuals must work closely with IT personnel to decide on the feasibility of their business vision an in directing them on how the service ought to be implemented

Business and Development Liaisons: These are individuals who understand both, the business functions of the domain and organization and the technology that implements the domain services, at least at a high level. They are responsible for translating business processes into technical specifications. They work closely with architects and developers in designing the SOA.

Developers: These are the architects and programmers who implement the services. They must insure that all specifications have been taken into account in the design and the system works as it was designed.

Testers: These are people who rigorously test out the system for usability and performance manually and by using tools.

Administrators: These are system specialists who are responsible for creating and enforcing SOA policies. They continuously monitor the system for usage patterns; diagnose system problems and alert responsible parties as necessary.

Summary

Like security SOA Governance is a holistic process that must involve all parts of an organization. SOA Governance is one of the most challenging, but significant aspects of SOA adoption and implementation. Some of the complexity of SOA governance comes from the very nature of SOA itself as we will see later in the chapter on SOA risks.

SOA in Industry

Most large IT vendors are pushing SOA tools in the marketplace. In this section, we will look at several SOA initiatives from some of the major players. We also look at SOA adoption for different needs and the growth of SOA in industry.

IBM's Efforts to Push SOA

According to an article on Mortley fool,[25]

"Bridging software and services is ***IBM*** *(NYSE: IBM), who wants SOA to lift growth of its $16.8 billion software business from 4% to 6%. For services, IBM says it has done 1,800 assignments for 1,000 clients, and is training 90,000 consultants in SOA."*

IBM's SOA propositions include the following:

Free trials on SOA Development tools

Big Blue's efforts to push large corporations to adopt to its infrastructure and consulting services include free trails of its application server, Websphere and its development tools to enable SOA implementors build prototypes of applications.

As developers in firms test out and learn SOA tools, enterprises would be more likely to use IBM in building their SOA infrastructure.

Ready for IBM

IBM promotes a 'Ready for IBM validation' as a potential goal for its customers. The customers will receive this certification when they achieve certain IBM specified SOA standards and when a certain number of their employees have reached some educational milestones.

Z System Access

The Z system is IBM's mainframe computer. IBM promotes it as a single alternative to distributed computing as a means for enterprises to reduce costs. To help corporations make this system their central internet server, IBM offers tools and initiatives to offer the Z system's computing power through web services.
In addition to housing and running corporate web applications, the Z system supports integration of email, and media applications on DB2 Viper, IBM's database.

One Z system development tool generates COBOL code from languages like Java or PL1 to allow flexibility in language for mainframe web development. The Z server offers powerful tools to support web applications, transaction processing and order shipment delivery status. Further, corporations can secure their web transactions with IBM's

[25] http://www.fool.com/investing/value/2006/09/18/soa-the-new-software-bazaar.aspx

security solution, the Tivoli identity manager. All systems used within an enterprise can be housed in a single System Z mainframe computer using web services.

In October 1996, IBM and Oracle together announced an initiative to ensure compatibility Oracle enterprise tools to run on System Z.

Sun's SOA Efforts

Sun Microsystems is another of the big players who offer SOA infrastructure solutions. These include

Sun's SOA Readiness Stages

Sun's version of SOA readiness includes for steps, Discovery, Analysis, Working Sessions and Findings. In the discovery stage Sun's consulting services research a customer's current business processes. In the analysis stage, Sun consultants look for opportunities where their SOA infrastructure can add value or mitigate weaknesses in the customer's business processes.

The consulting services then work with the customer to explain how Sun can bring value to the customer through its SOA infrastructure. This is the working Sessions stage. Finally, Sun provides the customer with a detail report that includes a strategy for the customer to migrate to SOA.

Project Kitty

Along with business partners BEA and Oracle, Sun released its SOA initiative at the 2004 June JavaOne Conference. Titled Project Kitty, the initiative included a Java Enterprise System, Java SOA development tools and Sun's SOA readiness Stages described above. These tools enable building of open, robust and extensible SOAs using the Java programming language.

Oracle and SOA

Oracle is yet another industry giant pushing its own SOA initiatives. These include the following:

Oracle Fusion

In January 2006, Oracle released its Fusion Middleware Suite that it claims can work with both Oracle and non-Oracle SOA applications. Just as it did with its previous E-Business suite Oracle is banking on broad based offering on a wide range of offerings capable of running on a variety of different platforms.

Oracle SOA 2.0

One of the most ambitious SOA infrastructure initiatives to date is what Oracle calls the 'next generation version' of SOA. At the Java One Conference, in San Francisco, in May, 2006, Steve Harris, the head of the Fusion middleware group, described 2.0 as a combined SOA and event driven architecture[26]. The event driven part of 2.0 provides the ability for the system to external events and notifications

This move has not been well received by all. In response to the announcement of 2.0, Mark Little[27] said

"I expected more of Oracle on this one! Giving an architectural approach a version number is crazy: it makes no sense at all!"

Little goes on to say,

"Where does it say that SOA is all about clients and servers?"

Macehiter Ward-Dutton, a consulting firm had an online petition[28] to oppose Oracle in making such changes to the SOA model.

SAP

Germany based SAP, a competitor to Oracle and another titan in Enterprise applications, was the largest investor in SOA for 2006 according to Mortley Fool[29]. Shai Agassi, a senior executive for SAP joined the firm when SAP acquired Top Tier, an Israeli software portal.

Agassi heads SAP's software development and is the Tour De Force behind SAP's SOA initiative. SAP acquired Netweaver and as a result made its flagship product SAP integration engine in 2002. SAP is thus trying to brand itself as a SOA Business Integration infrastructure vendor for large organizations.

Unlike Oracle's Fusion which is more open and supports integration with components with non- Oracle applications, the SAP's SOA platform is more catered to SAP and SAP partner build applications.

[26] http://www.infoworld.com/article/06/05/17/78420_HNsoa20_1.html
[27] http://www.webservices.org/weblog/mark_little/soa_2_0_ignorance
[28] http://www.mwdadvisors.com/resources/stop-the-madness.php
[29] http://www.fool.com/investing/value/2006/09/18/soa-the-new-software-bazaar.aspx

Accenture

Accenture, is a global IT consulting firm that according to Motley Fool[30] invested$450M on a SOA research lab. The company provides a SOA readiness model which will be described in the next chapter.

Microsoft

Microsoft's attitude toward SOA over the last couple of years has been interesting to observe.

An article on internetnews.com in February 2005 talked about somc collaboration between Sun and Microsoft in pushing SOA[31]. It also spoke of Sun offering federated identity management on between Microsoft, Linux and Solaris systems. Was Sun planning to team with Microsoft against IBM?

Microsoft also teamed up with Oracle, IBM and Sun in October 2005 for a new initiative with the OASIS standards body to standardize web service protocols to make it easier for architects to build SOAs[32]

In October of 2006, Microsoft for the first time, broke its silence on SOA and renamed its Business Process and Integration Conference for BizTalk partners, on this week, to "SOA and Business Process Conference" to add itself to the SOA bandwagon.

Microsoft pushed its SOA approach under the title, 'the Real World Approach to SOA'[33] .An article by Joe Kendrick on ZDNet summarized Microsoft's philosophy. Unlike IBM or Sun, Microsoft proposed a strategy of solving one business problem at a time rather than a grandiose top-down approach for the entire enterprise.

An interview with John deVadoss[34], director of Architecture Strategy, Microsoft appears on Microsoft's website. According to deVadoss'

"The fundamental problem with the big-bang mega approaches to SOA is that they almost always end up being out-of-sync with the needs of the business."

Further, deVadoss goes on to say,

[30] http://www.fool.com/investing/value/2006/09/18/soa-the-new-software-bazaar.aspx

[31] http://www.internetnews.com/bus-news/article.php/3467551

[32] http://www.itbusinessedge.com/item/?ci=7875

[33] http://www.itjungle.com/two/two101106-story04.html

[34] http://www.microsoft.com/presspass/features/2006/oct06/10-04SOA.mspx

"Customers and partners realize that big-science projects don't create value for the business,"

So here is Microsoft's approach in a nutshell, according to de Vadoss,

"We are proving that it is the incremental, iterative, real-world approach that helps you correct your course and deliver value to the business in a meaningful way."

Microsoft's roadmap of current and ongoing investments[35] according to deVadoss, includes a new guidance on how to deliver an Enterprise Service Bus (ESB) functionality to simplify and accelerate the development of SOA. Further, customers could take advantage of IBM midrange and mainframe solutions using via BizTalk adapter for Host systems.

Further, Microsoft's Office Application Strategy made possible by the new platform capabilities in the 2007 Microsoft Office System enables customers to realize the 'Real world' value of SOA.

Current Industry SOA Adoption

An article[36] by Mark Brunelli, news editor, SearchOracle.com talks about industry adoption of SOA. Forrester Research interviewed 53 early adopters and concluded that most are proceeding with caution. According to the article, the most telling fact is that most are not sure whether to standardize on an application platform provider like SAP or Microsoft or a middleware provider like IBM or Oracle.

The article mentions that Forrester analyst Wang believes that it may not be a question of deciding which vendor's approach is best, but a question of figuring out how many SOA systems can be supported by the respective . Large companies may need to support several SOA systems. Wang sees a lot more interoperability in the future.

The article also mentions that according to Anne Thomas Manes, a vice president and research director in the Burton Group, companies adopting to SOA expecting a project completion in 15 months are seeing its benefits in 6 months. Anne cautions that these benefits are from integration from web services and that this is not the same as SOA.

As mentioned in the article,

"SOA, Manes explained, is about identifying replication, eliminating duplicate implementations of functionality and ultimately removing application silos so that all systems share what they need to share at the core level. "

[35] http://www.microsoft.com/presspass/features/2006/oct06/10-04SOA.mspx
[36] http://searchoracle.techtarget.com/originalContent/0,289142,sid41_gci1209435,00.html

According to a Aberdeen Group benchmark report[37] in June 2006, from than 120 IT and business professional surveyed, 90% of the companies are adopting or have adopted to SOA. However, this has come at a price high integration costs and customization challenges.

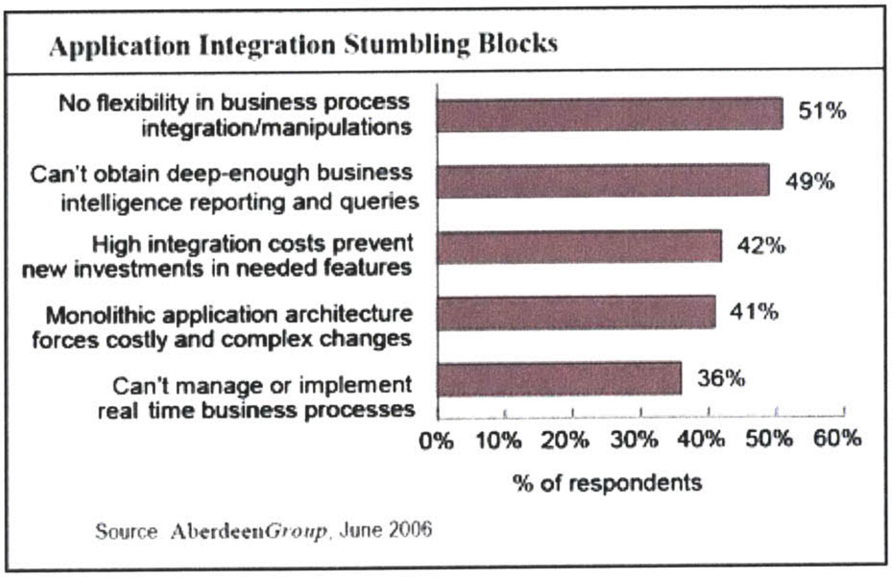
Kastner, vice the author of the report mentions that SOA is seen as a real technology step forward with large companies leading the way.

According to the report, enterprises are taking three approaches to SOA adoption.

These include, a 'Lite' version adopted primarily by small companies which uses simple web services, built with open source tools and adhering to industry standards. The second approach used by large companies is the 'Enterprise' version which is a more complex SOA and for critical applications.

Midsize companies adopt a 'ERP' version which primarily consists of SOA extensions to ERP software. The figure below from the Aberdeen Group[38], shows the usage issues of of different types of SOAs in application silos across the companies they surveyed.

Figure1: Application Silos



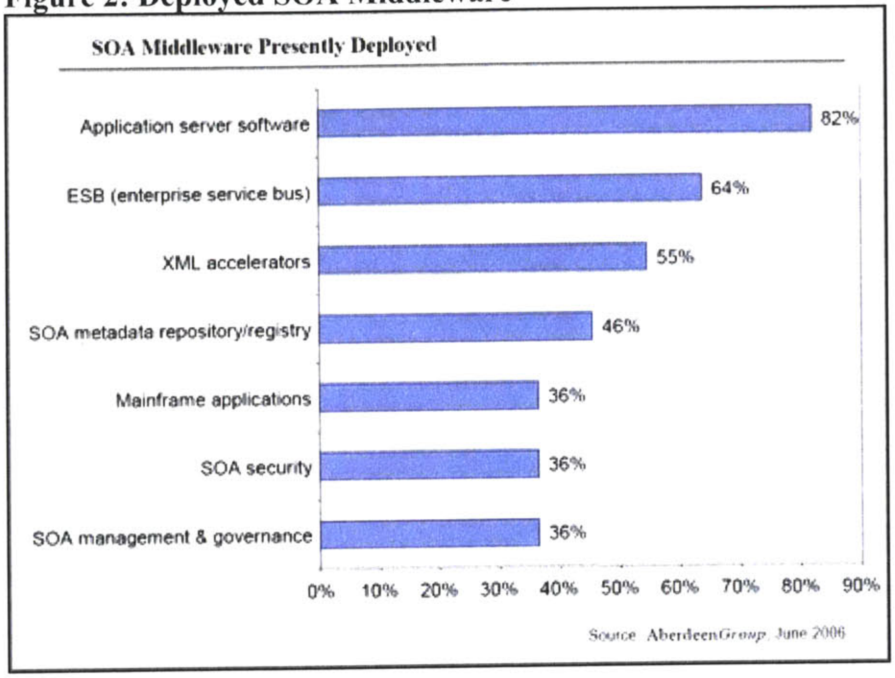
Source AberdeenGroup, June 2006

The figure below, also from the AberdeenGroup[39] shows the extent of SOAs deployed in middleware among the companies surveyed.

[37] http://www.dmreview.com/article_sub.cfm?articleId=1059636

[39] http://www.dmreview.com/article_sub.cfm?articleId=1059636

Figure 2: Deployed SOA Middleware



Source AberdeenGroup, June 2006

Examples of SOA used in Businesses

The following are some of the ways companies, educational institutions and other organizations today use SOAs to increase their effectiveness.

Consolidate Offerings

Many financial institutions and hospitals provide numerous online services using SOAs. These include application processing from legacy systems, consolidated offerings from several heterogeneous back end systems, etc.

Managing Data

Financial Institutions, government agencies like NASA, etc. use SOAs to facilitate access, online searching and other operations on large quantities of data. SOAs enable uniform presenting of different types and formats of data through a standard interface.

Online Transactions and Workflow Applications

SOAs today help organizations routinely process millions of daily transactions and also enable presenting workflow applications like online questionnaires and application forms. Customers are guided smoothly through clearly laid out screens through the questionnaire or application. SOAs can significantly reduce transaction management costs for organizations.

Customer Support and Management

Companies today use SOAs to serve customers. Many use SOAs to provide services from legacy systems. Others replace older systems with SOA frameworks from new state of the art systems. In addition to providing credit card management and online support, some SOAs also provide online customer warranty management.

Partner Services

Many companies use SOAs to present integrated offerings or offerings they licensed from their partners or 3rd parties using SOAs and thus leverage themselves to the customer.

Bringing Applications Online Sooner

With the ability to build SOAs quickly with powerful development tools, companies can easily build SOAs to build new applications and then use them for tasks like inventory management, order billing, etc and so on.

Improving Processes and Enhancing Agility

By allowing companies to form relationships with partners, agents and suppliers, companies invest in SOAs to become more efficient in their business process and can differentiate themselves from their competition in speed, quality and performance.

Summary

In this chapter we have looked at how SOA initiatives by major industry players and how organizations have implemented SOAs to create and add value to their business processes. As more organizations adopt to SOA we are bound to see more creative applications of SOA to contribute value to organizations. The next chapter goes into detail as to why organizations must adopt to SOA.

The Business Case for SOA

Earlier, we saw some of the technical benefits SOA offers architects and engineers. However, we only scratched the surface with respect to the benefits of SOA. SOA is a means of adding value to not just IT systems but to the entire enterprise. Leaders of all parts of the enterprise like Business Analysts, Finance, and Human Resources must collectively collaborate with the IT department to extract maximum value from an enterprise. Here I will present the business case as to why corporations must adopt to SOA. I will also offer ways in which SOA proponents can push for it within the organization. I will also prescribe models of SOA that organizations can choose to adopt based on their own business processes.

Why Adopt to SOA?

The following are some of the advantages gained from adopting to SOA.

Increased Output from Pre-existing Systems

SOAs can standardize the access and communication for both older systems like mainframes and newer servers. Companies that provide data from such systems to their customers can leverage themselves by building an SOA infrastructure around the older systems. By doing so, they can let more consumers access their systems. On the flip side, they can retrieve data from more systems and thus leverage themselves from more producers. SOAs can coordinate and facilitate communication among heterogeneous operating systems, platforms, applications and even infrastructures.

Flexible, Maintainable, Economical, Scalable Infrastructure

As mentioned in an article by Kishore Channabasavaiah, Kerrie Holley & Edward Tuggle[40]. SOAs can incorporate old and new systems and thus make the underlying infrastructure a commodity. SOAs can mitigate risk through the reuse of proven existing components and subsystems thus reducing costs and efforts. Reuse also eliminates redundancy of code and thus makes maintenance a breeze. For developers it reduces programming efforts. SOAs can be easily made scalable to deliver value from both existing assets and from new development or even through the incorporation of and external partner's asset. The use of SOAs also makes it easy to modify existing functionality to suit new business needs.

[40] http://www.looselycoupled.com/opinion/2004/chann-case-infr0927.html

Quicker Product and Service Delivery

When a SOA is established within a firm, with a set of well defined governance rules, adding new functionality becomes easier. Reuse also adds to quicker developmental time. As a result, firms can bring their offerings online sooner. Further, with SOAs deployment of services also becomes a lot faster.

Focus on Enhancing Business Process

Unlike prior technologies, SOAs focus on business processes and not on the programming model. Everyone in the organization is thus focused on the business processes and thus a company can enhance their offerings by examining their processes in the SOA, feedback from all corners inside them and from feedback from their customers.

Process Automation and Improved Data Access

Processes like inventory and supplier management can easily be automated with SOAs. As SOAs facilitate coordination of multiple systems, companies may have quicker access to data they need for their operations. As a result they can make better informed decisions sooner,

Enhanced Operations

SOAs force inter-organizational and interdepartmental cooperation, communication and coordination. Thus they force a behavior change in the organization as a whole. Such behavior change and better communication can increase operational efficiency, and reduce costs.

Companies can selectively exhibit their operational model to supply chain partners and thus help suppliers understand and improve their ability to serve them. Usage of web services or other services can grant companies an opportunity to expand into new market segments. They could thus increase the scope of existing partnerships and mergers or create entirely new ones. An SOA can also give the offerings of a company more visibility.

Furthermore, by observing the case histories of their customers and suppliers, firms can avoid dealing with corporations or individuals with questionable pasts. They could also track the authenticity of credit card payments and save themselves legal overhead. SOAs can also provide audit trails of online transactions.

More Choices of Technologies

The standardized nature of web services has brought more vendors in the market. Some vendors offer powerful tools to quickly build powerful web services while others offer entire infrastructures that could be customized to suit a firm's needs. As SOAs grow in popularity, the price of such tools and infrastructures are bound to drop due to competition, offering companies more choice at lower cost.

In addition to development tools web services can also offer options of leasing applications like CRMs or ERPs and using their services without investing money in buying them.

Adapting to Changing Market Conditions

By providing the ability to deploy services quickly, facilitated communication and quicker access to market data, SOAs can help companies adapt quickly to changing market conditions like sales opportunities, etc.

Market Segment Specific Benefits

SOAs can offer benefits to different market sectors based of the need of each sector. For example, in the *financial sector*, SOAs can service more users and grant them access to financial data from numerous sources. They can also help secure financial transactions and reduce costs of such services

In the *retail sector*, SOAs can provide access to more suppliers and consumers or enhance communication between different players in the supply chain. In warehouses they can help track inventory and status of shipments. Further, players on the supply chain will have timely access to supply and demand data. Suppliers can track a company's internal inventory quantities of material they supply online and can arrange for replenishments without additional communication

In the *healthcare sector*, SOAs can provide confidentiality of data, and help patients access more providers and make cost comparisons. Patients and doctors can have quicker access to data, test results and patient insurance information.

Integration of Disparate Systems

Within a company, SOAs can bring together data from different types of databases, spreadsheets, CRM and ERP applications. They can standardize access to heterogeneous systems and standardize presentation of data in different formats. They can also coordinate data among purchase, inventory and sales systems. All of this translates into improved efficiency and reduced cost for the company.

Pushing SOA Adoption in an Organization

So how does one push SOA Adoption in the enterprise?

The adoption of SOA by an organization needs a buy-in first from the IT department, upper management and last but not the least, a buy-in or at least a reluctant 'yes' from other stakeholders in the organization.

Usually, this effort is championed from someone in IT who has respect all over the organization. This person must sell the fact that an SOA is simply not a new IT system but an infrastructure that will require a paradigm shift in operations from all departments within the operations.

The champion must first push for a pilot project which has well defined goals in terms of output and cost savings. The question now becomes "How can organizations estimate their readiness for SOA adoption?

Accenture's SOA Readiness Model

Analogous to CMM's capability and maturity model that helps evaluate the maturity of a company's software process, **Accenture** has developed a four level SOA capability and progress Model[41].

Level 1: The Planning Stage
Companies at this stage have not adopted to SOA. This is the time senior management must be sold on the SOA vision. At this stage it is important understand the business processes and operations of the respective company. Then management with the help of internal or external expertise must carefully assess how SOA can leverage the company's business processes and assets.

Next, the organization can prepare for SOA transformation from both a business and technical perspective. From a technical standpoint, this may mean assessing the company's in-house expertise, technologies and investing in SOA tools. From an enterprise standpoint preparation involves bring together opinion shapers from different parts of the environment and look for a possible project that could quickly demonstrate the value of SOA.

[41] http://www.intelligententerprise.com/showArticle.jhtml?articleID=196601079

Level 2: The Early Deployment Stage

Companies at this level have a few SOA projects launched or deployed. They are just commencing the process of giving out their offerings as web services. They may also be in the process of consolidating and encapsulating their services as combined offerings.

Level 3: The Design of Services Architecture Stage

At this stage, firms are looking at SOAs from a business strategy perspective. They focus on offering their services as business processes and are now using sophisticated SOA tools for their development methodology. The focus has now shifted to looking at SOAs as an extension of the business rather than just an extension of IT.

Level 4: Industrialization of SOA

Companies in this stage have a mature SOA methodology. SOA is part of the company's DNA. They have or have the capability to have SOAs that operate across corporate boundaries, with security mechanism like federated identities. In addition to their own offerings, these companies may have services that internally or externally offer services from their partners. Their IT is robust and thus 'predictive' and as a result their operations are both efficient, timely and effective.

Though this model is not universally accepted, it offers a good framework for tracking SOA transitions and progress for companies. Like SEI, one would like to see an organization like Oasis grade companies on their progress of SOA adoption. This would also be an incentive for corporations to transition and enhance their SOA readiness.

The Champion's Story

Here are some suggestions for a SOA champion to sell SOA to an organization. As mentioned earlier, SOA can leverage all aspects of a firm's operations. A champion for SOA usually is an IT veteran with industry wide respect. However, to force SOA adoption, he or she might need a buy in not only from the CEO but also from CIOs CFOs and COOs.

Earlier in this chapter, we examined several reasons as to why a company must adopt to SOA. The champion can use many of these reasons to build a case to upper management to push SOA. It would be to the advantage of the champion if the CFO is an IT proponent and if the CIO has an understanding of how the existing IT services add to the company's operations

- The champion can use any of the reasons given earlier to higher management to create a case for SOA. However, he or she must tie the arguments for SOA with tangible benefits like reduced operational costs in actual estimates. However, the champion must also push for the intangibles like easy access to information, less reliance on IT personnel and real time information.

- Secondly the champion can push for a small pilot project where progress and benefits are visible and easy to understand.
- Another important part of the sales pitch is to tell the CEO or CFO that they will be the heroes who with SOA can expand operations for the firm, increase efficiency build organization cooperation or even create mergers and acquisitions which are strategic wins for the company.
- Further the champion can elaborate on governance and describe a 'dashboard' which an executive could use to control her domain.
- Finally the champion must sum up a strategy with a timeline on achieving various milestones of SOA progress using the SOA model described above.
- The champion must also elaborate on the new skills that need to be internally honed or externally bought to leverage the footing of the company's operations.

Vendor Considerations

With so many major players like SAP, IBM, Oracle and Microsoft offering SOA services, how does a firm pick a vendor? In the last chapter we looked at many SOA solutions offered by these companies. Should they go with a pure services company like IBM or should they select a company with product suites and services like Oracle or Microsoft?

It depends on several factors. Firstly, companies adopting to SOA must first look at their own businesses processes and their existing level of IT in-house to handle the business. Further they must look at their own IT systems, hardware and software. They must also decide on how quickly they plan to make a transition to SOA.

This exercise will give them an idea of how much they might have to invest in hardware, software and in hiring personnel. If a company already has some level of services or product or services or some service agreement with a major vendor what would it cost the company if the company decided to use this vendor's infrastructure? Are they happy with the level of service they currently receive?

Can the firm leverage its operations and business processes by going with another vendor? These are questions that must be answered through a careful cost-benefit analysis.

Cost Considerations

While SOAs can bring TCOs and yield large ROIs for corporations, their implementers will need factor in the following expenses.

Planning and Design Costs: SOA architecture requires planning, resources, hardware, and acquisition of software development tools and hiring or reallocation of personnel. All costs associated with these moves must into the SOA project budget.

Implementation Cost: Implementing, testing and deployment the system brings associated costs.

Support and Staff Training Costs: These are costs associated with the system, training staff on using the system, setting up and maintaining governance and security policies, etc.

Licensing costs: These may be costs associated in licensing software for development or security or simply using services from an SOA which is external to the company.

Other costs: These are other costs associated with running of the SOA.

Roles for SOA Infrastructure Providers

Up till now, we have looked at reasons why companies must adopt to SOA to optimize their business processes and some of the considerations that come up while they adopt to SOA. However, some IT companies can look for opportunities to help companies that would like to provide services without the overhead of an SOA infrastructure. Such IT companies could also offer the services of others in combination with the services they provide themselves to leverage their offerings. Further, they could offer lookup services to help consumers look up information on services or companies providing services. We will now look at some of the roles that companies can play in providing SOA services.

Content and Service Providers

Companies in Content Provider roles can either act as a content provider, consolidating data from many sources like news broadcasts from several worldwide agencies or financial data from worldwide stock exchanges, etc. Other organizations and companies could publish their own content as services.

As a service provider a company could either bring out functionality it implements as services or host, license and resell services provided by other companies and act as 'internet middlepersons' for other companies

The services in both cases, content and business services can be sold as a set of individual services or as a consolidated group of dedicated services.

Registry Management Providers

We saw earlier how web service providers can publicize their offerings as UDDI Registries. Registries also allow providers to classify their services based on taxonomy. Consumers of web services can locate and use the services of a company through a UDDI. Many large vendors set up UDDI registries, which are large database servers allowing providers to publish their services on them.

Companies could get into this business and thus help catalog names business and contact information for content providers and information about the services they provide. Sophisticated Registries called brokers can provide search tools for consumers to locate a service based on some criteria. They can also act as gateways in describing the usage policies of the services provided by a company

Summary

In this chapter, we have looked at the business case for SOA, some recommendations for pushing for SOA adoption in an organization and some of the business considerations that must be made while moving forward with SOA adoption. In the next chapter we will consider the major risks that come with SOA adoption and look for ways to mitigate these risks.

SOA Risk Analysis

While it may in the best interest of most organizations to adopt to SOA, the success of an SOA requires a continuous process of understanding the risks inherent in the system as well as the risks that arise from the continuous evolution of the system. We will look at the generic risks of SOA and then look at some of the risks that arise from the inherent nature of the SOA model. We will also look at the risks of certain types of SOA implementation.

Inherent SOA Risks

As we saw earlier, the very nature of SOA and Web services cause some of their weaknesses

- All web service communication mechanisms, SOAP, UDDI, WSDL, etc are XML based and thus can traverse firewalls and VPNs. None of them have built in security as part of their specification. Without any added security, exposed transactions are accessible to anyone who can access the internet. It is thus the responsibility of the implementer to use an appropriate security solution.

- Adding security like simple encryption can defeat the advantages gained by loose coupling. Implementations of SOAs must thus strike a balance between flexibility and security. Otherwise, encryption mechanisms can force tight coupling between producer and consumer, which means that changes on the producer side will warrant changes in consumer side.

- If security solutions are not made scalable, enhancements to security or to the system can be expensive and labor intensive.

- Foreseeing security threats and use cases for all scenerios in large SOAs with several loosely coupled components can be almost impossible and thus requires carefully proper risk contingency and usage planning. Systems must also be monitored for new use cases and security threats

Security Risks

We have looked at SOA security issues and technologies devoted to addressing those issues in SOAs. However, security contingency measures need an understanding beyond possible security threats and solutions. Enterprises must approach security from a much broader enterprise context. Here are some oversights that can happen when a company transitions to SOA.

Failing to understand Security Threats

Firms that are transitioned to SOA, or those who just have made the switch often fail to understand that they must confront security issues almost from scratch. As mentioned earlier, SOA because of its inherent nature brings to the forefront new security threats. Most corporations and their IT personnel are not aware of these risks.

Traditional methods of having a firewall could give corporations a false sense of security to corporations. As mentioned earlier, SOA messages are built to travel over HTTP or HTTPS, bypassing firewalls and thus allowing hackers to attack systems and access confidential information. It is the job of the implementers to make it clear to the organization that security is most important for the SOA and that everyone's input must be considered in planning contingency measures.

Furthermore corporations must use the right measures to monitor the effectiveness of security Just keeping track of reported security bugs is not enough to validate security of an SOA. Corporations must also focus on the usage patterns of their interfaces to check for security threats.

Underestimating importance of Security

Many firms fail to see the significance of security measures to their SOAs and often do not budget adequately for them. Some smaller firms that offer services simply do not have the money to put a security mechanism in place. As a result, their systems are vulnerable to Hackers and even inadvertent user access to sensitive information. Security issues must be brought forth and addressed right from the proof of concept and design phases.

Pushing the envelope

This is a malaise that exists at several levels in many corporations. Executives who may have a gut feeling about possibility of security threats often fail to question the effectiveness of their corporate security threats. They console themselves that IT has taken care of these issues. As a result important security considerations simply slip through the cracks.

Often IT folks whose expertise is primarily in software, do not want to confront security threats to switches, routers, bridges and other hardware equipment as it is outside their comfort zone. They must work together with hardware specialists and system administrators to monitor and evaluate security systems, both for hardware and software.

Sometimes a product team may simply abandon security issues to meet deadlines on their deliverables. This could leave security vulnerabilities which are simply time bombs waiting to be ignited.

Many firms that use third party tools and technologies to implement their security solutions rely on those vendors to handle security issues. This could prove dangerous as no SOA vendor or administrator can plan for all possible security threats. Security of SOAs is everyone's business.

Another incarnation of this passing of blame is to rely on security standards. As we have seen, many of these security standards are still in incomplete states of development and thus leave SOAs vulnerable to threats.

Failing to Monitor Effectiveness of Security Continuously

Many organizations fail to realize that Security Risk assessment and risk management is an ongoing process. While most security threats occur on hardware devices, hackers are constantly trying newer and cleverer ways to break into both software and hardware systems. An investigation of hardware security precautions is not enough for security.

Similarly merely investing in software security solutions is also inadequate. All people within the organization must brainstorm all possible usage patterns of their services, with ach service used singly and in combination with other services to check for possible vulnerabilities.

Also constant monitoring of usage patterns and access logs of SOA servers must be part of the SOA governance policy. All parties in any way associated with security must seek to understand security issues which may even be outside their sphere of influence.

Security evaluation must be done from as many diverse sources within and outside the system.

Inadvertent Security Risks

Many vendors offer tools that make it very east to build web services. This means that developers could build web services around legacy and older systems without adequate knowledge of their operations or their vulnerability to threats when exposed as web services. As a result new threats are created, compromising the security of the system. Whenever a new service or set of services are added as part as a corporate offering, security issues must be brought up and evaluated with them.

SOA Implementation Risks

Some of the risks associated with SOA implementations include:

Easy Basic Implementation Risks

There are many tools from Microsoft and other Java tools that can be used to create basic services, consumers of services and so on with very little code writing from the programmer. With such a tool, the programmer simply moves through a sequence of screens choosing one or more options as she goes along. The SOA Tool creates code based on the programmer's choices creates code and then at the end, creates a framework that the programmer can enhance. While these tools do provide a basic framework, they do not automatically make production level services for enterprises.

The ease of using such tools to create basic SOAs which are nothing more than prototypes causes developers to be complacent and as a result, the end service created may not be optimal in terms of performance, flexibility, reliability and security.

Standards Risks

As SOAs can be built as web services, RESTFUL services and so on, there is no single SOA standard and this can cause interoperability problems among SOAs. Though some aspects of web services exist, many of them are not comprehensive enough to guarantee security or interoperability among them.
Often, tight deadlines, feature creep and other factors can push developers to abandon adherence to them.

Vendor Choice Risks

Many large vendors offer SOA solutions to the enterprise. Companies must not base their choice of vendor simply based on a vendor's reputations. A company must carefully evaluate how well the vendor model can handle the company's business model and processes. Also consideration must be paid for level of support, stability and lifetime of the vendor company and SOA product. Companies must also asses the effectiveness of the contract they have with the vendor.

New Technology Risk

Companies transitioning to SOA must realize that they are creating a new layer of software and processes over their existing infrastructure. This means that they must deal with the new complexities of architecture, governance and security, management and so on. They must anticipate problems arising from deployment, lifecycle and change management and so on. As SOAs can interact with entities within and outside corporate boundaries they may have dependencies on external factors to run effectively.

Interaction and Coordination Risks

SOA brings together almost all departments in an organization. Many projects must be brought into a single architecture and this process involves coordination of technical and business groups. This coordination of several separate efforts to a single architecture is a monumental task that must be approached diligently and carefully.

Going overboard with Services

It is possible that some of a firm's business offerings may be optimally implemented via services. Service providers

must be comfortable in offering only a subset of their business functions as external services. Firms must implement services based on ROI, capability to add provisions for security breaches and in line with their governance policies.

Making Assumptions on Underlying implementation

Consumers must carefully adhere to service contracts and never make assumptions on the underlying implementation of a service. This could lead to interoperability problems. Further, the underlying implementation is subject to change at any point without any notice from the service provider. Developers must make the usage of the service clear and explicit in their contracts.

Too Fine Grained Services

In an attempt to enhance reuse, developers and architects may make even internal pieces of a system as a service. Services must be designed for the external customer must be kept lean, precise but powerful. Sometimes within an organization, developers may decide to treat every possible interface as a service. This may be overkill and also cause additional unnecessary maintenance overhead for the organization.

Focus on Programming Models instead of Business Processes

Services must abstract business processes and not programming models like object or RPC style models. Such an implementation could lead to tight coupling defeating the very purpose of having SOAs.

Summary

In this chapter we looked at some important risks considerations for Organizations that use or plan to adopt SOAs. It must be reiterated that careful risk analysis is the differentiating factor in the ultimate success or failure of an SOA.

The Future of SOA

Earlier, we saw that a study by the Aberdeen group has concluded that SOA adoption has now reached 90% in industry. In 2004, Gartner predicts that by 2007, most companies would have adopted SOA for both new applications and as an interface to legacy applications[42].

It is certain that companies must change their view of applications and develop new skill sets and tools to adopt an SOA strategy. Areas like security and the ability to handle large transactions still pose some of the biggest obstacles to industry wide option.

In an interview[43] with Forbes magazine, author and business consultant John Hagel describes web services as

".. a disruptive technology in that the early adoption is not disruptive at all--it simply offers the promise of doing what you already do cheaper, faster and more flexibly. The disruptive part comes after adoption because the technology now creates the option of restructuring businesses and moving to very different operating practices. This will take years if not decades to play out in full form, but we are already seeing early illustrations of the potential."

Hagel mentions in the same article that at least half of SOA implementations are happening outside the US. He also mentions that many national and state governments are adopting to SOA to lower the cost of information aggregation on websites and to enable the storage of information on diverse websites. The DoD, he says is also an early adopter and uses web services to increase flexibility to deploy military resources in response to targeted threats.

SOA adoption has taken on a multiple set of innovative and futuristic paths. Many of them have been through synthesis with newer and existing technologies. We will now look at some of the evolutions of SOA.

Enterprise Service Bus

An enterprise service bus is a layer of software abstraction over one or more systems that serves as a standards based event driven, messaging engine. It provides a framework which allows multiple systems to be integrated with it to form a distributed messaging system. ESBs are XML based, operating system and programming language agnostic. They are distributed and thus have no centralized rule engine.

[42] http://news.zdnet.com/2100-3513_22-5415942.html
[43] http://www.forbes.com/home/2002/12/09/1209hagelchat.html

Unlike traditional architectures, the ESB is build on functionality that can be componentized into smaller parts and deployed in a distributed fashion. Architects can use the EDB to send messages and alerts under certain situations without any programming.

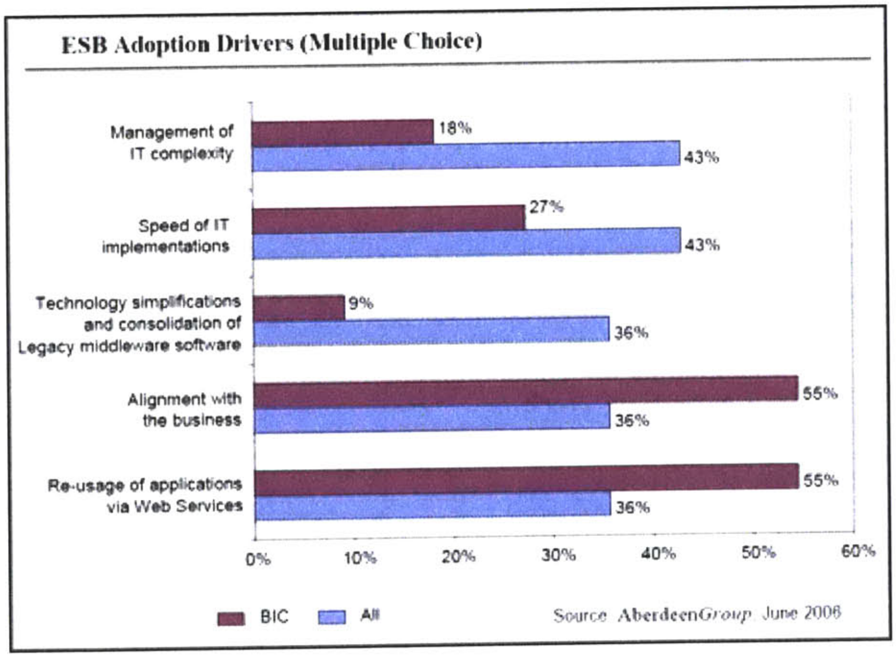
ESB do not include SOA functionality but an SOA can be implemented on its basic framework. It can support many transport mechanisms and when an SOA is implemented within it, the transport protocols of the SOA can be clearly separated from the services offered by the SOA. Many commercial ESB developers include SOA functionality to leverage their sales.

The key benefits of ESBs are that they allow faster and cheaper accommodation of existing systems, standards adherence, flexibility and scalability. Further, they allow for configuration rather than integration coding.

The ESB also has some disadvantages. One is that it requires a mandatory messaging model. This means that disparate systems must adhere to some messaging standards. Versioning problems on messaging models can actually make ESB components tightly coupled. Some ESB implementations may need more hardware.

ESBs rarely give back the ROIs on their early projects. For effective running of ESBs organizations actually need a pre existing well-run governance policy..

According to a report[44] by the Aberdeen Study Group, ESBs have been adopted by about 64% of organizations today. The graph below shows the various modes of usage of ESBs.



[44] http://www.dmreview.com/article_sub.cfm?articleId=1059636

Smart Clients

The last few decades have seen the multifold increase of power in desktop systems, thanks to powerful processors from companies like Intel and AMD. The combinations of processing power, the power of the web and the evolution of web services has evolved another emergent technology, Smart Clients. Smart Clients harness the power of the desktop and couple it with the power of the internet. Smart Clients can also be used with non-computer internet devices like PDAs and cell phones.

Harness Localized Processing Power

As mentioned above, Smart Clients can utilize the untapped power of the desktop or laptop or device and use internal hardware like memory to accomplish tasks. They can also interact with software within the device or other applications that are specific to the context of the task they perform, When the desktop, laptop or device is not connected to the internet, the smart client can do house keeping tasks and work on optimizing performance with caching of data or effective connection management.

Web Access

When the desktop is online, a smart client can request data or processing from a remote web service and also be capable pf deploying or updating applications automatically as necessary. They can be configured to deploy external applications based on certain permissions or security mechanisms.

In addition, Smart clients can also use SOAs to provide security mechanisms, and configuration of access policy for applications. Combined with SOAs, Smart Clients can enrich the user experience and increase the response time with caching and asynchronous data fetching.

Device Specific Customization

Smart clients can be made machine or device specific. This means that when they work, they can take advantages of the unique features of the device where they run.

SOAs provide the ability to draw a clear line between business logic and the front end. They provide a uniform interface to access data. Smart Clients that reside on a variety of devices can seamlessly connect to SOAs to access data or to perform computations.

Performance, Scalability and Robustness

The traditional client-server model broke up tasks so that all heavy processing was done at the server end. However, the smart client paradigm has shifted some of the heavy work to the client side and thus enables use of the untapped power of the desktop. Thus distributed applications can divide tasks in an optimal fashion between front and back ends, thereby improving their performance. In conjunction with SOAs, smart clients allow applications to be more robust, flexible and scalable.

Open Source and SOA

The open source movement has gained ground in corporations today. Linux has been an operating system of choice for many corporations in the server space and in the embedded market. JBoss and Tomcat application servers and Mysql databases are other open source success stories and 70% of all web servers used today are Apaches.

Open Source and SOA are compatible in many ways. For instance, one of the main goals of both is reuse although they approach reuse in different ways. Another SOA goal is to provide the end user with the best possible solution to address his or her need. The theory of 'many heads' works with open source SOA as it will involve a large programming community working to offer the end user with optimal solutions. True to the goals behind SOA, Open Source developers strive to make their code as flexible, robust and adaptable to the end user. Also open source software is almost always intended to be backward compatible with their previous releases. This aspect of open source will add value to SOAs as it is a means of maintaining interoperability.

Open Source has spread into the SOA space and we now do have open source registries and SOA development packages. Examples of open source SOA tools are JBoss and Logic Blaze for ESB and SOA related software. The JBoss bundle is called the JBoss Enterprise Middleware Suite (JEMS). The JEMS offering includes Hibernate, Apache Tomcat, JBoss, JBoss Portal, Messaging, Cache and an Application Server. The JBoss Eclipse development environment enables development and testing of SOA applications. So does open source SOA pose a threat to commercial software?

SOA systems in large organizations act as a framework to facilitate coordination, management, performance and interoperability and communication. Traditionally, providers of such solutions like CRM and ERP vendors have formed close alliances and relationships with their customers and by offering support services.

In his article, Is Open Source SOA Software Ready for Prime Time?, Eric Koch warns that forming such partnerships will be a challenge for SOA. He goes on to propose that if open source must threaten commercial software, it must give more value added benefits. He believes that if a company wants to make a profit out of open source SOA, it must adopt a model similar to one like Red Hat uses with Linux distribution.

Though Linux is free to acquire and install, it takes a lot of overhead to deploy, support, manage and maintain. Red Hat distributes Linux with customer support, training and other services like bug fixes, etc. Red Hat thus becomes the vendor of choice for companies using open source Linux. This is the model that Koch proposes for open source. A company can thus make its revenue by bundling open source SOA and offer Support services.

Firstly, Open Source SOA systems and development tools are still in their infancy in comparison to their commercial counter parts. In his article, Koch warns that it would presently be 30% more costly to install and maintain a open source solution than going with a commercially purchased solution. However, the ROI attained through an open source SOA would be 200% higher. In essence Koch maintains that there is added value for companies to use open source one they overcome the initial cost factor.

Grid Computing and Web Services

Grid computing is a computing paradigm where computers distributed across a network act as a single virtual entity to distribute tasks and enable parallel computing when possible. As a result, the single virtual computer can solve large complex computational problems by breaking them down into sub tasks and thus enable higher performance.

There is no centralized control in this model and computation is distributed. The computers in the grid must use standardized protocols and interfaces to enable interoperability among heterogeneous machines and optimize performance through efficient use of available resources. This is where web services become important to grid computing.

As early as February 2002, an article by Tom Sullivan of Infoworld[45] talks about Sun's efforts to combine grid computing and web services. The product, Technical Compute Portal, is composed of the iPlanet Portal, Sun's Grid Engine, and Sun ONE (Open Network Environment). Sun One is build on web services and allows resource access and communication though them. The processing power of the system comes from the grid dengine while iPlanet provides the standardized interface.

In an article on Techtarget.com, titled 'The future is grid computing and Web services as one[46]', Aneet Shah mentions that at the Finexpo City Technology Strategies conference in February 2004, JP Morgan and UBS had a session where they explained how they are using grid computing and Web services together. Shah goes on to say,

"They are reporting positive returns from early small-scale implementations of Web services and Grid computing, raising expectations for wider uptake in new application areas across the enterprise."

While organizations today continue to adopt both Grid computing and SOAs, Aneet suggests that the next logical step would be to harness the power and capability of grid computing as part of a service oriented architecture.
He goes on to say,

[45] http://www.infoworld.com/articles/hn/xml/02/02/15/020215hnsunonegrid.html
[46] http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci949511,00.html

"However there are still many challenges that need to be overcome to make this an industry-wide strategy. Some challenges that will need to be considered over the next few years are: improving and developing interoperability standards across the distributed resource management space while there are operational grids are being implemented and the general computing environment is evolving rapidly; complete migration to a Web service-orientated architecture is still a while away; and finding applications that can efficiently utilize both of them. "

Services used within the grid called Grid services and must adhere to guidelines specified in the *Open Grid Services Infrastructure (OGSI) specification(*www.ggf.org/ogsa-wg*).*

The specification extends the WSDL and WS-Security formats and provides standardized formats to name, locate, create and manage the lifecycles of grid services. Grid services like web services use SOAP for their communication.

According to Grimshaw, Grid Service conventions are not superficial in their function and address fundamental issues in distributed computing relating to how to name, create, discover, monitor and manage the lifetime of stateful services.

In his article 'Introducing Open Grid Services', Savas Parastatidis[47] mentions that in June 2003 that "the Global Grid Forum (GGF) adopted the Open Grid Services Infrastructure (OGSI) specification (www.ggf.org/ogsa-wg) as a GGF standard"

He goes on to say that OGSI is essential to the Open Grid Computing vision as it is the foundation on top of which the building blocks of future Grid applications will be placed. Those building blocks, the Grid services, are being defined by various GGF working groups, with the Open Grid Services Architecture (OGSA) working group orchestrating the entire process.

SOA and M2M

The field of Machine to Machine (M2M) also commonly referred to as the Pervasive Internet is involved in bringing online machines and peripherals of everyday use. These could include HVAC systems, Music or Movie disks, house keys, cardio monitors, bar codes on merchandise, etc.

It raises the possibility of connecting such devices to intranets and internet, to enable the monitoring of performance, receiving alerts on critical entities, and to control and manipulate them.

Manufacturers can them monitor the performance of their equipment, doctors and medical personnel can receive alerts from hospital devices situated in a different place.

[47] http://webservices.sys-con.com/read/39827.htm

Building administrators could control the temperatures of remotely located buildings, etc. Remote control of devices could be done via a computer or a personal device.

An article[48] titled 'Moving SOA onto the Plant Floor' by Tom Kevan mentions that it is difficult to evaluate the benefits of SOA in industrial production environments as software vendors have concentrated on applications for industries that have a high volume of data movement like Financial institutions. Vendors supplying software for plant-floor systems have been slow to adopt this technology.

The article goes on to say that temperature control system vendors like Rockwell and Siemens are all bringing such products into the market. These products will be SOA enabled to allow their monitoring and control over the net.

The article further says that the growing presence of software called Sensor middleware may make SOA's ROI more visible. Augusta Systems' SensorBridge allows rapid integration of sensor assets into existing data systems. Rockwell's FactoryTalk integrator allows manufacturers to connect to business systems. Sensicast's Sensinet Gateway combines wireless sensors with legacy systems.

This is yet another growing field that is still very much in its infancy. SOA may be the key to building intelligence in everyday devices and creating endless possibilities.

Summary

This chapter provided an overview of some growing fields that use SOA. The growing popularity of SOA in organizations will create more widespread use of SOA technologies. Many innovators will use SOA to create and deliver value in a multitude of new applications and in other industry segments.

[48] http://www.sensorsmag.com/sensors/article/articleDetail.jsp?id=393403

Conclusion

In this thesis, I have introduced Service Oriented Architectures, described its evolution, and then described the most popular technologies used to implement SOA. Further I have built both a technical and a business case for SOA adoption. I have also presented an overview of the current state of SOA today in the corporate world and presented how some key proponents of SOA have proposed their models for SOA adoption. Finally, I examined a few possible future paths in the evolution of these technologies.

It is worthwhile for any architect or business analyst to read the chapters on SOA risks, security and governance before embarking on SOA adoption. As mentioned earlier, the goal of this thesis is to introduce the reader to SOA and outline its advantages. It is not intended to be a how-to model.

Here are some key takeaways from my analysis.

- SOAs have evolved from some of the best aspects of existing technologies and from other common computing themes that have repeatedly reoccurred from time to time.

- Web Services are the most common types of SOAs implemented. Web services are XML based and include SOAP, WSDL, UDDI technologies. They offer a standardized format for providers to register their services and for consumers to lookup and use the services provided by providers.

- SOAs provide benefits both from a technical and a business standpoint.

- Most of the research on SOA reveals that companies adopting to SOA can realize measurable gains through relatively small investments in a short timeframe.

- SOA adoption and subsequent architectural decisions must be made by all key departments within a company. SOA is not for the IT department alone. Corporations choosing to adopt to SOA must educate all major department (Marketing, Sales, Accounting, Finance, Human Resources, IT) heads and opinion shapers on SOA, its potential benefits and risks.

- The chapter titled 'The Business case for SOA' explains the benefits, vendor and cost considerations that companies must evaluate while adopting to SOA. Some companies may choose to deliver SOA services. The same chapter also describes two roles a corporation can take delivering SOA. These include Content and Service providers and Registry management providers. Companies must carefully examine their own business model and choose one or more roles for themselves that fit this model.

- The chapter titled 'The Business case for SOA' also provides some recommendations for SOA champions and evangelists on how to sell SOA to upper management. SOA proponents can use some of the advantages of SOA mentioned here and blend it with the advantages to their own respective business model SOA brings in making their case.

- The same chapter also describes how corporations must choose a vendor based on their business needs. This is a vital decision and must be decided based on the company's business model, the robustness and flexibility of the vendor architecture (based on available reports/reputation of the vendor) the level and quantity of the service and support provided by the vendor. Can the architecture of the vendor be nimble and flexible enough for the changing nature of your company's business? The chapter 'SOA in Industry' describes the offerings of some of the large SOA infrastructure vendors. This chapter can be used this as a starting point for company decision makers in researching the infrastructure most appropriate for their respective line of business.

- The SOA architecture adopted by a corporation must be evaluated based on the company's current business model and its future directions of growth. Can the model be customized to adopt to future business directions like partnerships, mergers, acquisitions and other changes that can happen to the respective organization? Can it be adopted to include new suppliers or new services from existing suppliers? How quickly can the system access new data in a timely fashion? How can it leverage existing assets of the organization?

- While a corporation must plan on an overall SOA strategy applicable for entire organization, it is wise to start small with a pilot project where an ROI value can be estimated before hand and subsequently evaluated once the SOA is operational. This strategy would help key players make financial estimates on value created and earned by an SOA as well as the technical challenges, risks and security issues associated with SOA.

- As SOAs are implemented and enhanced, corporations must give due consideration on the functionality that they wish to provide as services. It may wise to keep certain functionality as it is for a number of business or security reasons.

- Functionality of services provided by a firm must be based on the core competencies of the firm, leveraging existing assets, reusable and redundant functionality and consolidation of resources.

- Before an SOA goes live, all heads in the organization must work together with IT to establish company wide governance policies and evaluate security threats to the system. A reliable reporting and auditing system must be in place to report on all relevant activities carried on in the context of the SOA and to alert the respective authority when critical situations arise.

- After an SOA is live, it must be constantly evaluated for risks and security threats. It must also be evaluated for compatibility with the latest SOA standards..

- Corporations implementing SOA must also set and update policies and procedures on how the systems must be used. This is an ongoing process and needs to be addressed each time there is a business or architectural change in the operations of a corporation. A live policies document in a well defined location ensures that everyone within a corporation is on the same page with respect to governance.

SOA is not a silver bullet for corporate business processes. However, a well run corporation with good business processes can leverage itself tenfold with a modest investment through a well planned and managed SOA implementation.

References

Books

J2EE Web Services, Richard Monson-Haefel, Addison Wesley

Introduction

http://www.xml.com/pub/a/ws/2003/03/04/security.html
http://www.xml.com/pub/a/ws/2003/09/30/soa.html
http://www.webmethods.com/Products/SOA/Why
http://h71028.www7.hp.com/enterprise/cache/329751-0-0-225-121.htm006C
http://en.wikipedia.org/wiki/Service-oriented_architecture
http://www.xml.com/pub/a/ws/2003/09/30/soa.html
http://www.looselycoupled.com/opinion/2004/chann-soa-infr0630.html
http://bobbreedlove.com/tech/soahistory.html
http://www.looselycoupled.com/opinion/2004/chann-soa-infr0630.html
http://www-128.ibm.com/developerworks/webservices/library/ws-improvesoa/
http://objectwebcon06.objectweb.org/xwiki/bin/download/Main/DetailedSession/A-Trenaman-SOA.pdf
http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf
http://en.wikipedia.org/wiki/OASIS_SOA_Reference_Model
http://www.ws-i.org/about/Default.aspx
http://www.fool.com/investing/value/2006/09/18/soa-the-new-software-bazaar.aspx

SOA History

http://en.wikipedia.org/wiki/Internet
http://en.wikipedia.org/wiki/Computer_networking
http://bobbreedlove.com/tech/soahistory.html
http://www.computerhistory.org/exhibits/internet_history/

HTTP

http://en.wikipedia.org/wiki/HTTP (HTTP)

RPC

http://en.wikipedia.org/wiki/Remote_procedure_call
http://tools.ietf.org/html/rfc707

Object Oriented Technology

http://en.wikipedia.org/wiki/Abstraction

http://en.wikipedia.org/wiki/Abstract_object
http://en.wikipedia.org/wiki/Type_polymorphism
http://en.wikipedia.org/wiki/Inheritance_%28computer_science%29
http://en.wikipedia.org/wiki/Polymorphism_%28computer_science%29
http://en.wikipedia.org/wiki/Object-oriented_programming

CORBA

http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=396&page=1
http://www.auditmypc.com/acronym/IIOP.asp (IIOP)

Component Technologies

http://www.informit.com/articles/article.asp?p=345781&rl=1
http://objectwebcon06.objectweb.org/xwiki/bin/download/Main/DetailedSession/A-Trenaman-SOA.pdf

Microsoft Technologies: (COM /DCOM)

http://www-128.ibm.com/developerworks/library/co-tmline/index.html (DCOM)
http://www.informit.com/articles/article.asp?p=345781&rl=1
http://msdn2.microsoft.com/en-us/library/6zzy7zky(VS.80).aspx
http://www.samspublishing.com/library/content.asp?b=Visual_C_PlusPlus&seqNum=221&rl=1

Microsoft.Net History

http://en.wikipedia.org/wiki/.NET_Framework

Java

http://ei.cs.vt.edu/~history/Youmans.Java.html

XML

http://www.nytimes.com/library/tech/00/06/biztech/technology/07mark.html

SOAP

http://webservices.xml.com/pub/a/ws/2001/04/04/soap.html

Loose Coupling

http://en.wikipedia.org/wiki/Loose_coupling
http://www.johnhagel.com/view20021009.shtml

Encapsulation

http://www.cs.cmu.edu/People/clamen/OODBMS/Manifesto/htManifesto/node5.html

Interoperability and Language Independence

http://www.webopedia.com/TERM/I/interoperability.html
http://Whatis.com
http://safari.informit.com/013046130X/ch02lev1sec4
http://en.wikipedia.org/wiki/Cross-platform

Reuse

http://en.wikipedia.org/wiki/Code_reuse

Deployment

http://en.wikipedia.org/wiki/Software_deployment

Standardization

http://www.enterprisenetworksandservers.com/monthly/art.php?2135

Evolution of SOA

http://objectwebcon06.objectweb.org/xwiki/bin/download/Main/DetailedSession/A-Trenaman-SOA.pdf

XML Origins

http://www.nytimes.com/library/tech/00/06/biztech/technology/07mark.html

Technologies of Service Oriented Architectures

http://en.wikipedia.org/wiki/Service-oriented_architecture

Web Services

http://en.wikipedia.org/wiki/Web_service
J2EE Web Services , Richard Monson-Haefel, Addison Wesley
http://en.wikipedia.org/wiki/Service-oriented_architecture

HTTP
http://www.jmarshall.com/easy/http/

SOAP

http://en.wikipedia.org/wiki/SOAP

WSDL

http://209.85.165.104/search?q=cache:CfT_v1u6vs4J:sern.ucalgary.ca/courses/CPSC/547/W2002/slides/uddi-wsdl.ppt+uddi+simple+example&hl=en&gl=us&ct=clnk&cd=9
http://www.w3.org/2001/03/14-annotated-WSDL-examples.html

REST

http://www.xml.com/pub/a/ws/2003/09/30/soa.html
http://www.zapthink.com/report.html?id=ZAPFLASH-2006712
http://expertanswercenter.techtarget.com/eac/knowledgebaseAnswer/0,295199,sid63_gci983966,00.html

Technical Benefits of SOA

http://www1.webmethods.com/PDF/The_Business_Case_for_SOA.pdf
http://www.ftponline.com/ea/magazine/winter2006/features/tbeack/
http://www.soa.com/index.php/section/solutions/interoperability/
http://java.sun.com/developer/technicalArticles/WebServices/soa/index.html

Reuse

http://www-128.ibm.com/developerworks/webservices/library/ws-reuse-soa.html?ca=dgr-lnxwReuseSOA

Encapsulation

http://msdn.microsoft.com/isv/solution/crm/soa/default.aspx

Interoperability

http://www-128.ibm.com/developerworks/webservices/library/ws-soa-seminterop.html

Abstraction

http://blogs.cio.com/node/179

Real Time Information

http://msdn.microsoft.com/isv/solution/crm/soa/default.aspx

Benefits to developers

http://www.j2eegeek.com/blog/2005/10/01/beaworld-2005-soa-keynote-panel-discussion/

Jini and SOA

http://dev2dev.bea.com/blog/clarkrichey/archive/2005/07/the_ultimate_so.html

SOA Security

http://www.aspnews.com/strategies/article.php/11296_3613041_3
http://expertanswercenter.techtarget.com/eac/expertAnswer/0,295208,sid63_gci1022062,00.html
http://www.pesg.com/soa/security.html
http://expertanswercenter.techtarget.com/eac/expertAnswer/0,295208,sid63_gci1022062,00.html
http://blogs.ittoolbox.com/eai/business/archives/soa-security-architecture-11431
http://www.cigital.com/papers/download/bsi12-soa.doc.pdf
http://msdn.microsoft.com/webservices/webservices/understanding/specs/default.aspx?pull=/library/en-us/dnglobspec/html/wssecurspecindex.asp

An Overview of security solutions

http://blogs.ittoolbox.com/eai/business/archives/soa-security-architecture-11431
http://www.xml.com/pub/a/ws/2003/03/04/security.html?page=2
http://xml.coverpages.org/saml.html
http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf
http://www-128.ibm.com/developerworks/library/specification/ws-secon/
http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf
http://www.w3.org/TR/xmldsig-core/
http://blogs.ittoolbox.com/eai/business/archives/soa-security-architecture-11431

SOA Governance

http://www-128.ibm.com/developerworks/webservices/library/ws-soa-govern/
http://www-306.ibm.com/software/solutions/soa/gov/
http://www.zapthink.com/report.html?id=ZAPFLASH-10272004
http://archive.infoworld.com/reports/04SRsoagov.html
http://www.infoworld.com/article/06/01/19/73698_04FEsoagov_1.html
http://www.weblayers.com/gcn/whitepapers/Introduction_to_SOA_Governance.pdf

SOA in Industry

http://www.fool.com/investing/value/2006/09/18/soa-the-new-software-bazaar.aspx
http://blogs.zdnet.com/service-oriented/?p=508
http://news.zdnet.com/2100-3513_22-5415942.html

IBM's SOA Initiative

http://www.infoworld.com/article/06/05/07/78113_HNibmsoamainframe_1.html
http://news.com.com/IBM+pulls+partners+into+SOA+push/2110-1011_3-5766540.html
http://www.crn.com/sections/breakingnews/breakingnews.jhtml?articleId=193104528&cid=CRNBreakingNews

Sun's SOA initiative

http://www.sun.com/smi/Press/sunflash/2004-06/sunflash.20040628.9.xml

Oracle SOA

http://www.oracle.com/technologies/soa/soa-suite.html
http://www.eweek.com/article2/0,1895,1910378,00.asp
http://www.infoworld.com/article/06/05/17/78420_HNsoa20_1.html
http://www.webservices.org/weblog/mark_little/soa_2_0_ignorance

Microsoft and SOA

http://blogs.zdnet.com/service-oriented/?p=423
http://www.microsoft.com/presspass/press/2006/oct06/10-04PragmaticSOAPR.mspx
http://www.ebpml.org/ms_-_soa.htm
http://blogs.zdnet.com/service-oriented/?p=725
http://www.internetnews.com/bus-news/article.php/3467551
http://www.infoworld.com/article/06/10/04/HNmsesbsoa_1.html
http://www.microsoft.com/presspass/features/2006/oct06/10-04SOA.mspx

The Business Case for SOA

http://www.looselycoupled.com/opinion/2004/chann-case-infr0927.html
http://www-128.ibm.com/developerworks/webservices/library/ws-arc2.html
http://www.investni.com/index/develop/ebusiness/ebusiness_web_services/ebusiness_web_services_what_are_the_benefits.htm
http://www.sun.com/products/soa/industry.jsp
http://www.webservicesarchitect.com/content/articles/clark02.asp
http://www.intelligententerprise.com/showArticle.jhtml?articleID=196601079
http://www.mega.com/index.asp/l/en/c/solution/p/it-architecture-governance/p2/service-oriented-architecture

http://www.codeproject.com/webservices/BPI_with_Web_Services.asp

Pushing SOA in the organization

http://searchoracle.techtarget.com/originalContent/0,289142,sid41_gci1209435,00.html
http://www.intelligententerprise.com/showArticle.jhtml?articleID=196601079
http://www1.webmethods.com/PDF/The_Business_Case_for_SOA.pdf

Pricing Models

http://www.webservicesarchitect.com/content/articles/clark02.asp

SOA Risk Analysis

http://pluralsight.com/blogs/tewald/archive/2005/03/04/6336.aspx
http://www.cigital.com/papers/download/bsi12-soa.doc.pdf

Risks from Loose coupling

http://www.aspnews.com/strategies/article.php/3613041

Risks of not using a business registry

http://www.ebizq.net/topics/soa/features/5652.html

Security Risks

http://www.cio.com/blog_view.html?CID=20038
http://www.cio.com/archive/110106/col_key.html

Implementation Risks

http://www.weblayers.com/gcn/whitepapers/Introduction_to_SOA_Governance.pdf

The Future of SOA

http://news.zdnet.com/2100-3513_22-5415942.html
http://www.forbes.com/home/2002/12/09/1209hagelchat.html

ESB

http://en.wikipedia.org/wiki/Enterprise_service_bus

Smart Clients

http://msdn.microsoft.com/isv/solution/crm/soa/default.aspx

Grid Computing

http://en.wikipedia.org/wiki/Grid_computing
http://www.infoworld.com/articles/hn/xml/02/02/15/020215hnsunonegrid.html
http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci949511,00.html
http://webservices.sys-con.com/read/39829.htm
http://webservices.sys-con.com/read/39827.htm

M2M

http://www.maya.com/web/what/papers/harbor_infofuture_sept2005.pdf
http://www.bpm-today.com/story.xhtml?story_id=0120007JU1TO

SOA and Open Source

http://blogs.ittoolbox.com/eai/business/archives/is-open-source-soa-software-ready-for-prime-time-11046
http://www.thbs.com/tech_docs/final_whitepapers/Role%20of%20Open%20Source-SOA.pdf