

Service Oriented Architecture (SOA)

A New Paradigm to Implement Dynamic E-business Solutions

Joseph Bih

Tyler Junior College

P.O. Box 9020, Tyler, Texas 75711

E-mail: jjmbih@ieee.org, adj-jbih@tjc.edu

Abstract:

Software development has undergone various stages of paradigms – the very last one is called Object-Oriented (O-O) paradigm. Typical of this paradigm has been culminated by several O-O programming languages such as C++, Java and VB. With the advent of Internetworking and evolvement of web technologies, business needs, especially isolated service requests/components, put a new demand on new software architecture. Existing models are examined in the light of their constraints and limitations to meet these constant changing business requirements. Service-oriented Architecture (SOA), as the next generation software architecture and through the utilization of the web service, XML and other related technologies provides viable working solution to implement dynamic e-business. This paper discusses SOA model and how it fits into implementing dynamic e-business solutions.

1. Introduction

Over the last four decades, software architectures have attempted to deal with increasing levels of software complexity. As the level of complexity continues to evolve, traditional architectures do not seem to be capable of dealing with the current problems. While traditional needs of IT organizations persist, the need to both respond quickly to new requirements of the business and continually reduce the IT cost, and the ability to absorb and integrate new business partners and new customer sets become more in demand. The industry has gone through multiple computing architectures designed to allow fully distributed processing, programming languages designed to run on any platform, greatly reducing implementation schedules, and a myriad of connectivity products designed to allow better and faster integration of applications. Service Oriented Architecture (SOA) is being advocated in the industry as the next evolutionary step in software architecture to help IT organizations meet their ever more complex set of challenges. [2]

The existence of Web services technologies has stimulated the discussion of Services Oriented Architecture (SOA), which has been advocated for more than a decade now, ever since CORBA extended the promise of integrating applications on disparate heterogeneous platforms. Problems of integrating those applications arise, often because of so many different (and non-CORBA-compliant) object models. Architects and engineers alike became so bogged down in solving technology problems, constantly in search for a more robust architecture that would allow simple, fast, and secure/seamless integration of systems and applications was lost. Meanwhile, the distributing computing model opens the way of cross-platform and cross-programming language interoperability. SOAP is a great distribution computing solution because it achieves interoperability through open standards at the specification level as well as the implementation level.

Meanwhile, basic business needs such as lowering costs, reducing cycle times, integration across the enterprise, B2B and B2C integration, greater ROI, creating an adaptive and responsive business model demands better solutions. "Point solutions" won't work as desired solutions for the lack of a consistent architectural framework within which applications can be rapidly developed, integrated, and reused. Thus an architectural framework must be developed to allow the assembly of components and services for the rapid, and even dynamic, delivery of solutions; an architectural view unconstrained by technology.

1.1. Definition of Service-Oriented Architecture (SOA)

A service-oriented architecture [3] is essentially a collection of services, among which the communication can involve either simple data passing or it could involve two or more services coordinating some activity, requiring means of connecting services to each other. The first service-oriented architecture in the past was with the use DCOM or Object Request Brokers (ORBs) based on the CORBA specification. [4]

To understand service-oriented architecture must begin with a clear understanding of the term service. [4] A service is a function that is well defined, self-contained, and does not depend on the context or state of other services. The technology of Web services is the most likely connection technology of service-oriented architectures. Web services essentially use XML to create a robust connection.

The following figure 1 [4] illustrates a basic service-oriented architecture. It shows a service consumer at the right sending a service request message to a service provider at the left. The service provider returns a response message to the service consumer. The

request and subsequent response connections are defined in some way that is understandable to both the service consumer and service provider. How those connections are defined is explained in Web Services explained [5]. A service provider can also be a service consumer.

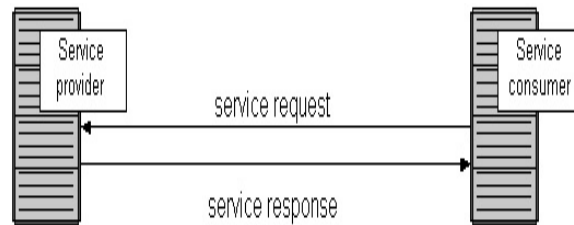


Fig. 1 Basic Service-oriented Architecture

As a distributed software model, an SOA is usually comprised of three primary parties: Producer (of services), Consumer (of services), Directory (of services), as shown in figure 2 [6]. Web Services are considered an example of Service Oriented Architecture. Service Networks take on the properties of an SOA.

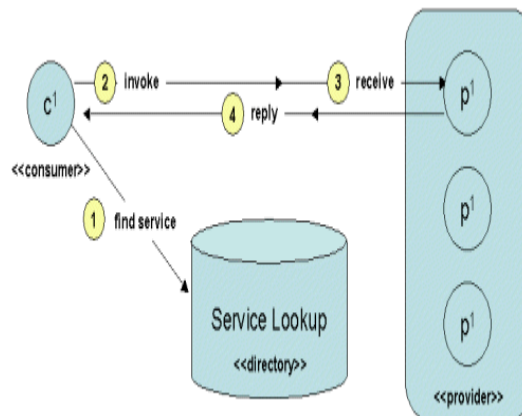


Fig. 2 Components of Basic Service-oriented Architecture

1.2. Service-Oriented Architecture (SOA) Model

It all started with HP's e-speak [1] that appeared with a marketing campaign built around a proprietary implementation of SOA. However, e-speak failed to make much of a market impact partly because of its proprietary requirements. Nevertheless, the potential concept of SOA was found to have merit by companies like IBM and Microsoft who recognized that for SOA to succeed where other distributed computing

concepts had failed, it must be implemented on open standards. Thus, the recent cooperation between these companies on recommended standards like UDDI and WSDL.

1.2.1 SOA Model

According to IBM, SOA is comprised of three participants and three fundamental operations, regardless of its implementation, (see Figure 3[2, 7]).

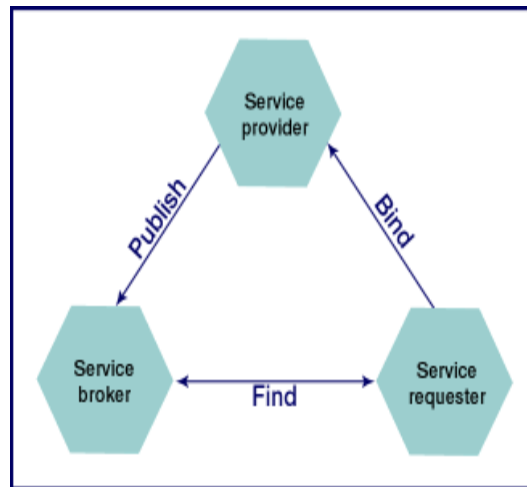


Fig. 3 The SOA model

REPRINTED WITH PERMISSION FROM DAN GISOLFI ©2001

A *service provider* is a Network node that provides a service interface for a software asset that manages a specific set of tasks. A service provider node can represent the services of a business entity or it can simply represent the service interface for a reusable subsystem.

A *service requestor* is a Network node that discovers and invokes other software services to provide a business solution. Service requestor nodes will often represent a business application component that performs remote procedure calls to a distributed object, the service provider. In some cases, the provider node may reside locally within an intranet or in other cases it could reside remotely over the Internet. The conceptual nature of SOA leaves the networking, transport protocol, and security details to the specific implementation.

The *service broker* is a Network node that acts as a repository, yellow pages, or clearing house for software interfaces that are published by service providers. A business entity or an independent operator can represent a service broker.

These three SOA participants interact through three basic operations: *publish*, *find*, and *bind*. Service providers *publish* services to a service broker. Service requesters *find* required services using a service broker and *bind* to them. The interactive process among these 3 agents calls/centers on the *service* components (rather than objects which characterizes Object paradigm).

1.2.2. Business Roles

Because of the role-based nature, SOA strives to meet services and business needs much more effectively. In the service-oriented architecture (SOA) of Web services, three distinct actors - the *Provider*, the *Requestor*, and the *Broker* interact to help an organization make a choice among five possible business roles. [7]

Service Requestor: For a business to identify with this SOA role, it must find some commonality between their business activity and the actions of a requestor. There are two clear business activities that would allow a business to benefit from implementing the role of a service requestor - Content Aggregation and Service Aggregation. Content Aggregation is an activity where a business entity interacts with a variety of content providers to process or reproduce such content in the desired presentation format of its customers (such as Internet portal or information service provider). Service Aggregation is an activity where a business entity interacts with a variety of service providers to re-brand, host, or offer a composite of services to its customers (such as a mobile portal and the alike of OnStar) (www.onstar.com).

Service Provider: For a business to identify with this SOA role, it must view itself as performing some degree of an electronic service. Whether that service is defined as the processing of data or the act of carrying out a specific task, the business entity must believe it is performing work for others as an occupation or a business.

Registry: If a business entity finds itself collecting and cataloging data about other businesses and then selling that data to others, it may identify well with a registry, a form of SOA Broker. Usually, a registry would collect data such as business name, description, and contact information. In UDDI terms, this SOA role is often referred to as the *White Pages*.

Broker: Building on the concept of a registry, business entities may also be able to identify with the notion of a broker, which in UDDI terms is often referred to as *Yellow Pages*. Brokers usually extend the value proposition of a registry by offering intelligent search capability and business classification or taxonomy data.

Aggregator/Gateway: Any business entity that provides Broker capabilities plus the ability to describe actual policy, business

processes and binding descriptions would be able to identify itself as *Green Pages*.

2. SOA, Web Services and XML Technologies

2.1. Why SOA?

The practice of software development has gone through several different programming models, causing/shifting architecture paradigm. Each shift was made in part to deal with greater levels of software complexity and to enable the integration of applications through parts, components, or services. More recently, Java technology contributed platform-neutral programming, and XML contributed self-describing, and thus platform-neutral, data. Web services have removed another barrier by allowing the interconnection of applications in an object-model-neutral way. Employing a simple XML-based messaging scheme, Java applications can invoke DCOM-based, CORBA-compliant, or even VB applications.

CICS or IMS transactions on a mainframe in Hong Kong can be invoked by a COM-based application driven by Lotus Script running on a Domino server in London. The invoking application is largely separated from where the transaction runs, what language it is written in, or what route the message may take along the way. Just that simple: a service is requested, and an answer is provided. SOA creates an architecture paradigm shift from Object-oriented to Service-oriented – calling directly the service components rather than objects.

Web services are more likely to be adopted as the more genuine standard to deliver effective, reliable, scalable, and extensible machine-to-machine interaction than any predecessors, as a result of the timely convergence of several necessary technological and cultural prerequisites. A ubiquitous, open-standard, low cost network infrastructure, and technologies that make for a distributed environment much more conducive to the adoption of Web services than both CORBA and DCE faced; a degree of acceptance and technological maturity to operate within a network-centric universe that requires interoperability in order to achieve critical business objectives, such as distributed collaboration; consensus that low-cost interoperability is best achieved through open Internet-based standards and related technologies. [2]

The maturity of network-based technologies (such as TCP/IP), together with tool sets (IDE's, UML, etc.), platforms (such as J2EE platforms and .NET Framework), and related methodologies (such as OO, services, etc.), provide the infrastructure needed to facilitate loosely-coupled and interoperable machine-to-machine interactions,

promising a much brighter future than what CORBA users could have expected.

Further more, Grid computing is not just the application of massive numbers of MIPS to affect a computing solution but also provide a framework whereby massive numbers of services can be dynamically located, relocated, balanced, and managed so that needed applications are always guaranteed to be securely available, regardless of the load placed on the system.

The availability and effective use of these new resources and capabilities will require the restructuring of many existing applications. SOA comes at a time, when pressing need for conversion of architecture is needed, to utilize the opportunities and adapted itself in the new era best suited for building dynamic e-business solutions.

2.2. SOA vs. Web Services

The advent of Web services has produced such a fundamental change that the implementation of a true service-oriented architecture becomes a reality. This makes it possible to examine the application architecture while addressing the basic business problems. From business perspective, it is a matter of developing an application architecture and framework within which business problems can be defined, and solutions can be implemented in coherently and consistently.

First, though, it must be noted that there is a difference between *Web services* and *service-oriented architecture*. Web services is a collection of technologies, including XML, SOAP, WSDL, and UDDI, which make it possible to build programming solutions for specific messaging and application integration problems. These technologies are on the way to mature, and eventually be replaced with better, more efficient or more robust ones. They are, at the very least, the basis that SOAs can finally be implemented.

SOA, however, is an architecture. [1] It is more than any particular set of technologies, such as Web services; it transcends them, and, in a perfect world, is totally independent of them. Within a business environment, a pure architectural definition of SOA might be something like "an application architecture within which all functions are defined as independent services with well-defined invocable interfaces which can be called in defined sequences to form business processes". All functions are defined as services - purely business functions, business transactions composed of lower-level functions, and system service functions. These services are independent, operating as "black boxes"; external components neither know nor care how they perform their function merely that they return the expected result.

In general, the interfaces are invocable; that is, at an architectural level, regardless of local (within the system) or remote (external to the immediate system) level, what interconnect scheme or protocol is used to effect the invocation, or what infrastructure components are required to make the connection. The service can be within the same application or in a different address space within an asymmetric multiprocessor, on a completely different system within the corporate Intranet, or within an application in a partner's system used in a B2B configuration.

As the key element, interface focuses on the calling application. It defines the required parameters and the nature of the result; thus, it defines the nature of the service, not the technology implementation. The system is responsible for effectively managing the invocation of the service, not the calling application. This allows two critical characteristics to be realized: the truly independent services, and manageable services. Management of these services includes: [2]

- Security -- authorization of the request, encryption and decryption as required, validation, etc.
- Deployment -- allowing the service to be redeployed (moved) around the network for performance, redundancy for availability, or other reasons
- Logging -- for auditing, metering, etc.
- Dynamic rerouting -- for fail over or load balancing
- Maintenance -- management of new versions of the service

2.3. Integrating XML Technologies into SOA

XML technologies provide the foundation for web services. The surrounding XML technologies are summarized as follows: [8]

XML: The Extensible Markup Language 1.0 standard is a text-based markup language specification from the World Wide Web Consortium (W3C). Unlike HTML, which uses tags for describing presentation and data, XML is strictly for the definition of portable structured data. It can be used as a language for defining data descriptive languages, such as markup grammars or vocabularies and interchange formats and messaging protocols.

SOAP: Simple Object Access Protocol is an XML-based lightweight protocol for the exchange of information in a decentralized, distributed environment. SOAP defines a messaging protocol between requestor and provider objects, such that the requesting objects can perform a remote method invocation on the providing objects in an object-oriented programming fashion. The SOAP specification was co-authored by Microsoft, IBM, Lotus, UserLand, and DevelopMentor. The specification subsequently spawned the creation of the W3C XML

Protocol Workgroup, which is comprised of over 30 participating companies. SOAP forms the basis for distributed object communication in most vendor implementations of SOA. Although SOA does not define a messaging protocol, SOAP has been referred to as the *Services-Oriented Architecture Protocol* due to its common use in SOA implementations. The beauty of SOAP is that it is completely vendor-neutral, allowing for independent implementations relative to platform, operating system, object model, and programming language. Additionally, transport and language bindings as well as data-encoding preferences are all implementation dependent.

WSDL: The Web Services Description Language is an XML vocabulary that provides a standard way of describing service IDLs. WSDL is the resulting artifact of a convergence of activity between NASSL (IBM) and SDL (Microsoft). It provides a simple way for service providers to describe the format of requests and response messages for remote method invocations (RMI). WSDL addresses this topic of service IDLs independent of the underlying protocol and encoding requirements. In general, WSDL provides an abstract language for defining the published operations of a service with their respective parameters and data types. The language also addresses the definition of the location and binding details of the service.

UDDI: The Universal Description, Discovery, and Integration specification provides a common set of SOAP APIs that enable the implementation of a service broker. The UDDI specification was outlined by IBM, Microsoft, and Ariba to help facilitate the creation, description, discovery, and integration of Web based services. The motivation behind UDDI.org, a partnership and cooperation between more than 70 industry and business leaders, is to define a standard for B2B interoperability.

XML's strength is to describe and manipulate complex data. As one successful implementation shows, The JUNOScript API encodes data for both requests and replies in the W3C's Extensible Markup Language (XML) [9]. XML's simple encoding allows the representation of arbitrary hierarchies of data using ASCII text in HTML-like tags. The order and structure of the elements can be defined using either Document Type Definitions (DTDs) or XML Schemas [10].

XML technologies also present many other advantages [11, 12]: Better integration of management data from disparate sources; more flexible link between managed object and management application; tighter interoperability among management applications from different vendors; easy and powerful rendering and transformation of management information; automatic and centralized validation for management data. These advantages and other maturing

technologies, compatible with SOA architecture provide a working solution to implement dynamic e-business solutions.

3. Implementing Dynamic e-business Solutions

3.1. Challenges, Business Requirements in Dynamic e-business

Today, growth through merger and acquisition has become common. IT organizations, applications, and infrastructures must be integrated and absorbed in the dynamic process of growth and evolution. In an environment of such constantly changing nature, any "point solutions" merely exacerbate the problem rather than provide a better solution. Systems must be developed heterogeneous environment to accommodate an endless variety of hardware, operating systems, middleware, languages, and data stores. The need to deal increasingly side effects resulted from redundant and non-reusable programming calls for new software architecture to be developed that meets business as well as technical requirements.

First, to leverage existing assets. Existing systems often contain within them great value to the enterprise. Strategically, the objective is to build a new architecture that will yield all the value hoped for, but tactically, the existing systems must be integrated such that, over time, they can be componentized or replaced in manageable, incremental projects.

Second, to support all required types or "styles" of integration. [2]

- User Interaction -- being able to provide a single, interactive user experience
- Application Connectivity -- communications layer that underlies all of the architecture
- Process Integration -- choreographs applications and services
- Information Integration -- federates and moves the enterprise data
- Build to Integrate -- builds and deploys new applications and services.
- Allow for incremental implementations and migration of assets - this will enable one of the most critical aspects of developing the architecture: the ability to produce incremental ROI. Countless integration projects have failed due to their complexity, cost, and unworkable implementation schedules.
- Include a development environment that will be built around a standard component framework, promote better reuse of modules and systems; allow legacy assets to be migrated to the framework and the timely implementation of new technologies.

- Allow implementation of new computing models; specifically, new portal-based client models, Grid computing, and on-demand computing.

3.2. Distributing Computing Protocols – CORBA, DCOM and Web Services Technology

While CORBA and DCOM models have been successfully implemented on many platforms in the past, their limitations are exposed more obviously since the solutions/applications built on these protocols depend on a single vendor. These protocols are server-to-server and depend on a closely administered environment, therefore lack of interoperability. The advent of Internet and challenge of business requirements – integration of applications outside the enterprise create a need as well possibility for implementation solutions to be more vendor, platform and language neutral. The Web Services Technology stack is comprised of HTTP, XML, SOAP, WSDL, UDDI, and WSFL. HTTP, an RPC-like protocol is simple, widely deployed, and firewall-friendly. As an elegant protocol, HTTP is payload agnostic transport that offers most of the connection management features also found in CORBA and DCOM. The lack a mechanism, however, for representing parameter values in the RPC messages makes XML language comes in to represent a platform-neutral tagged-data. XML allows data to be serialized into a message format that is easily decoded on any platform. Yet, (unlike DR and CDR), XML is simple to use, offers a flexible easy-to-extend data format, and is supported on virtually every computing platform. SOAP is not tied to a specific transport protocol; HTTP has become the early favorite among SOAP adopters. Using HTTP, a SOAP envelope uses XML as an encoding scheme for request and response parameters. A SOAP message is basically an HTTP request and response that complies with the SOAP encoding rules.

3.3. Dynamic e-business Model

The dynamic e-business strategy is founded on a core set of emerging technologies, resulted from the collective efforts of a variety of companies and industry organizations. But critical to the successful implementation of e-business largely depends on open Internet standards.

According to IBM's Web services, the feature of next generation of e-business focuses on the integration and infrastructure complexities of B2B by leveraging the benefits of Internet standards and common infrastructure to produce optimal efficiencies for intra- and inter-enterprise computing.

published and accessible; program-to-program messaging must be compliant with open Internet standards; applications to be built on seamless transition through combining core business processes with outsourced software components/resources.

Others such as software resources are important too. An increase in the availability of granular software resources should improve the flexibility and personalization of business processes; Reusable outsourced software resources should provide cost and/or productivity efficiencies to service consumers; Further more, common architectures and open Internet standards are needed to support SOA architecture; XML-based management tools/policies are already in place to facilitate the process.

Today, however, most of the systems and applications running in the business world are made up of tightly coupled applications and subsystems. The drawback with this is that a change or disruption to any one subsystem can causes breakage in a variety of dependent applications, leading to the high cost of system maintenance and the limitations around the number of trading partners one can manage.

Service-Oriented Architecture (SOA [1]) as discussed above, in its very initial conception, addresses the core issue and challenge of implementing dynamic e-business strategy, thus provides an *architecture* for implementing dynamic e-business solutions. SOA, in its true nature as an architecture paradigm, overcomes these problems.

4. Conclusion

Service-oriented architecture allows designing software systems that provide services to other applications through published and discoverable interfaces, and where the services can be invoked over a network. To implement a service-oriented architecture using Web services technologies is to create a new way of building applications within a more powerful and flexible programming model. Development and ownership costs as well as implementation risks are reduced. SOA is both an architecture and a programming model, a new way of thinking about building software.

SOA defines the next generation software architecture through embracing the challenges brought by emerging business and technology needs. SOA is targeted at providing more efficient business solutions through a new software architecture paradigm. Through the utilization of variety of technologies, SOA addresses and responds the needs of dynamic evolvement of business and web service, therefore, provides effective implementation architecture for dynamic e-business.

5. References:

- [1] Karl Gottschalk and the IBM Web Services Architecture team, "Web Services architecture Overview - The next stage of evolution for e-business"
<http://www106.ibm.com/developerworks/webservices/library/w-ovr/>
- [2] K. Channabasavaiah, K. Holley, E. M. Tuggle, Jr., "Migrating to a service-oriented architecture"
- [3] http://www.service-architecture.com/web-services/articles/serviceoriented_architecture_soa_definition.html
- [4] www.service-architecture.com
- [5] http://www.servicearchitecture.com/web-services/articles/web_services_explained.html
- [6] <http://www.serviceoriented.org/>
- [7] "Web services architect, Part 2: Models for dynamic e-business". <http://www106.ibm.com/developerworks/webservices/library/ws-arc2.html>
- [8] <http://www.xml.com/pub/a/ws/2000/11/01/protocols/quickref.html>
- [9] <http://www.w3c.org/TR/REC-xml>
- [10] <http://www.w3.org/TR/xmlschema-0/>
- [11] Network Management Research Group, <http://www.ibr.cs.tu-bs.de/projects/nmrg/>.
- [12] OASIS Management Protocol TC, <http://www.oasis-open.org/>.
- [13] Dan Gisofi, "An introduction to dynamic e-business", <http://www106.ibm.com/developerworks/webservices/library/ws-arc1/resources#resources>. April 1, 2001
- [14] Alan K. Karp, "E-Speak Explained", Communications of the ACM, July 2003/Vol.46.No.7.

6. Author's Biography:

Since completing his graduate studies at the University of North Texas, Joseph Bih has been teaching undergraduate courses at Jarvis College and Tyler Junior College for more than 5 years. He is actively involved in research, with many publications through professional journals and technical conferences. His work resulted in grants that helped upgrading computer lab facilities and software, directly benefited the students. He has MCSE, MCSD, CCNA, SJWCD and OCP other IT certifications and is actively involved in the industry. His research interests focus on secure networks, information systems management and new technology applications. Joseph Bih is an IEEE Senior Member. He can be reached at jjmbih@ieee.org.

SOURCE: Ubiquity -- Volume 7, Issue 30
(August 8, 2006 - August 14, 2006)