Enterprise SOA: What are the benefits and challenges?

George Feuerlicht

Faculty of Information Technology,
University of Technology, Sydney,
P.O. Box 123, Broadway, 2007, NSW, Australia
jiri@it.uts.edu.au

Abstract

There is now little doubt that service-oriented computing will play a major role in enterprise computing in the future. Equally, it is evident that the benefits of enterprise SOA (Service-Oriented Architecture) are oversold at present and that the expectations for the potential of SOA to address most enterprise computing issues are too high. This paper is a discussion of the business drivers that motivate organizations to adopt SOA, and the challenges that must be addressed when making the transition to SOA. We argue that in order to gain the full benefits of SOA organizations need to go well beyond adopting Web Services standards and technologies and implement support for the entire life-cycle of service-oriented applications.

Keywords

SOA, Web Services, benefits, challenges

1 Introduction

The present enterprise computing architecture is the result of more that three decades of ad-hoc evolution resulting in highly complex and heterogeneous environment that is characterized by the coexistence of several generations of architectural approaches. Successive generations of enterprise computing architectures and approaches have attempted to address computing requirements of user organizations with the most up-to-date hardware and software technology platforms, each time claiming substantial benefits over previous approaches, but seldom delivering on such promises. Starting with centralized, mainframe solutions in the 1970s, progressing to client/server systems in 1980s, and Internet computing applications in 1990s organizations have attempted to support their increasingly complex business processes and at the same time to control the cost of IT solutions. Centralized mainframe computing is associated with poor scalability and limited flexibility, and generally regarded as not being able to keep up with user demands for applications. The main attraction of client/server systems was improved user interface and scalability achieved by distributing the processing load across multiple server and client platforms, with some configurations supporting thousands of concurrent users. However, this lead to proliferation of client (mainly PC) workstations and various types of servers (e.g. database servers, mail servers, application servers, etc), resulting in an environment where the ongoing maintenance and administration costs constitute a major component of the Total Cost of Ownership (TCO). The emergence of three-tier client/server systems, while enhancing scalability has introduced another layer of complexity with additional application server middleware. This situation was further exacerbated by the advent of Internet computing with massive increase in the size of the user population and corresponding increase in the number, size and complexity of the server platforms. Unlike centralized mainframe systems that are characterized by limited capacity to accommodate users and applications, distributed systems in general tend to provide excess processing capacity and suffer from resource underutilization. Modern enterprise computing environments typically involve independently developed and deployed applications systems (e.g. ERP applications) each configured to accommodate the maximum processing load and

storage. As individual application systems are unable to share resources effectively, most server and storage resources remain significantly underutilized and in some cases idle at least some of time. In retrospect, distribution of computing resources associated with client/server and Internet computing models, while achieving good scalability and many other benefits, also produced a situation where most resources are poorly utilized, difficult to administer and expensive to maintain.

1.1 Need for a new architectural approach

As a result of globalizations and increasingly more competitive business environment, organizations are looking for ways to minimize IT costs and at the same time offer more sophisticated products and services. This is forcing organizations to focus on their core business and increasingly rely on outsourcing and business partner relationships in order to deliver comprehensive products and services to their customers. This effort is hindered by the current business structures and supporting information systems that are typically acquired and organized according to specific business functions they support into separate information processing *silos* (e.g. Customer Relationship Management, Human Resources, Order Management, etc.). This leads to duplication of functionality and data, and results in high integration costs. Importantly, such environments are excessively complex and inflexible making it difficult for organizations to respond to changing business requirements in a timely manner. It is widely recognized that a new architectural approach is required to operate effectively in today's business environment. In addition to the requirement for more flexible and effective *internal* enterprise IT architecture, enterprises need to be able to interact with other organizations by forming supply chains, or by outsourcing individual business functions to external service providers without incurring excessive interaction costs. Traditional business-to-business integration methods such as EDI (Electronic Data Interchange) and similar techniques are costly and highly inflexible. Recent technological advances, in particular Web Services and related technologies are beginning to have a significant impact on the interaction costs associated with externally sourced services. Reduction in interaction costs allows outsourcing of finer grained business functions, so that a business processes can be configured from a combination of outsourced business functions each potentially supplied by a different service provider. However, to take advantage of these opportunities, organizations need to implement suitable business structures supported by a closely aligned IT architecture that facilitates the composition of complex business processes using a combination of internally and externally sourced services.

Service Oriented Architecture is widely regarded as having the potential to address the above issues. Most analysts predict that SOA will become the architecture of choice for enterprise computing and its adoption will lead to increased efficiency, the ability to rapidly respond to changing business environments, and significantly improved return on investment. According to Gartner Research SOA will be the prevailing software engineering practice by 2008 [Gartner, 2003]. In a recent forecast IDC estimates that worldwide spending on SOA-based external services will reach $8.6 billion in 2006, and almost $34 billion by 2010 [IDC.2006]. Given such forecasts it is important to ask the following questions:

1) What exactly is SOA?

2) Are these benefits really achievable?

3) What is required to make a transition to SOA?

In this paper we first describe the characteristics of the Service Oriented Architecture (section 2) and then discuss the benefits and challenges of SOA (section 3). Finally, we discuss the requirements for successful implementation of enterprise SOA applications (section 4).

2 What is SOA?

There is not much agreement about what exactly constitutes a Service Oriented Architecture. There are many different definitions of SOA, for example WC3 [WC3, 2004] defines SOA as a form of distributed systems architecture that is typically characterized by the following properties:

Service abstraction: *logical* view of programs, databases, business processes, etc., defined in terms of what it does, typically carrying out a business-level operation.

Message orientation: The service is formally defined in terms of the messages exchanged between provider agents and requester agents. The internal structure of an agent, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away.

Description orientation: A service is described by machine-processable meta data. The description supports the public nature of the SOA: only those details that are exposed to the public and important for the use of the service should be included in the description.

Granularity: Services tend to use a small number of operations with relatively large and complex messages.

Network orientation: Services tend to be oriented toward use over a network, though this is not an absolute requirement.

Platform neutral: Messages are sent in a platform-neutral, standardized format delivered through the interfaces. XML is the most obvious format that meets this constraint.

Forrester Research [Forrester, 2005] defines SOA as a style of design, deployment, and management of both applications and software infrastructure in which:

Applications are organized into business units of work (business services) that are (typically) network accessible.

Service interface definitions are first-class development artifacts, receiving the same degree of design attention as databases and applications.

Quality of service (QoS) characteristics (security, transactions, performance, style of service interaction, etc.) are explicitly identified and specified for each service.

Software infrastructure takes active responsibility for managing service access, execution, and QoS.

Services and their metadata are cataloged in a repository and are discoverable by development tools and management tools.

Protocols and structures within the architecture are predominantly, but not exclusively, based on industry standards (such as the emerging stack of standards around SOAP).

SOA definitions involve the concept of a service as a basic building block of distributed applications and the concept of an Enterprise Service Bus (ESB) - a software infrastructure that enables SOA by acting as a middleware layer that facilitates the interactions between services. Services (similar to components) have well-defined interfaces and independent implementations; the separation of the interface and implementation results in encapsulation of services creating potential for reuse. Services are loosely coupled, i.e. independent of the underlying technology platform and communicate via coarse-grained, asynchronous messaging.

Clearly, if SOA is to deliver on its promise to revolutionize enterprise computing it must be fundamentally different from existing approaches. Many of the SOA concepts are not new and are integral to earlier generations of distributed computing technologies such as CORBA and J2EE. For example, Enterprise Service Bus is reminiscent of CORBA ORB (Object Request Broker), WSDL is

similar to CORBA IDL (Interface Definition Language), and UDDI to CORBA Naming service. While the concepts are similar, CORBA has never delivered on its promise of a "global information appliance" because of lack of agreement among its various proponents (members of OMG – Object Management Group), slow evolution of the CORBA standard, complexity of the specification, and a host of other technical and implementation issues. Notwithstanding the lack of agreed definition, SOA has the benefit of twenty years of distributed systems R&D, including the successful implementation of J2EE and .Net component architectures.

SOA is best described as a set of architectural concepts that together form a blueprint for enterprise computing architecture. Service-oriented computing and SOA represent the next step in the evolution of distributed computing, rather than a *revolution*, and the benefits of implementing SOA are likely to be delivered incrementally. In most enterprise environments SOA will be adopted gradually and will coexist with more traditional IT architectures. SOA extends (rather than replaces) existing component architectures designed to facilitate the development of tightly coupled enterprise applications, and addresses the requirements of highly distributed, loosely-coupled inter-enterprise applications. While component architectures such as J2EE focus on supporting the development and deployment of enterprise applications by providing distribution transparency (across address spaces), SOA creates explicit service boundaries taking into account network latency and failures. The emphasis in SOA is not on the programming model; instead it focuses on the exchange of business information [Maron, 2005].

3 Benefits and Challenges of SOA

According to analysts, SOA benefits include reduction of integration costs, improved business agility and flexibility, improved asset reuse, and most importantly improved return on investment. From another perspective, transition to SOA is essential to allow participation in the new environment of closely integrated partner systems (i.e. to facilitate supply chain integration and extended enterprises), and outsourced IT services (i.e. to facilitate integration between service providers and consumers). SOA is also necessary in order to allow enterprise to take advantage of the emerging world of Utility Computing [Feuerlicht and Vořišek, 2004], [Feuerlicht and Vořišek, 2006].

The shift towards SOA has been facilitated by the unprecedented level of standardization based around Web Services core standards: XML, SOAP, WSDL, and UDDI, and the various WS-.* extensions that address service security, reliability, transactions, coordination and management. The universal acceptance of the core Web Services standards by the industry produced a situation where for the first time it is technically possible for applications to interact across diverse computing environments without incurring massive integration costs. The SOA standardization stack extends from low level protocols (HTTP and SOAP) right up to the level of business process execution. Graphical tools enable designers to develop BPEL language (Business Process Execution Language) process specifications that are orchestrations of Web Services and can be automatically translated into executable multi-party business processes.

3.1 What are the SOA Challenges?

However, it should be emphasized that agreement on technical standards (i.e. XML, SOAP, WSDL, etc.) alone does not ensure that the business benefits of SOA will be fully realized. Equally, an implementation of the entire Web Services standards stack does not constitute a Service-Oriented architecture. The adoption of technology standards is a necessary prerequisite for achieving low cost integration between disparate technology platforms, but other issues that include service analysis and design and agreement on data structures and semantics across the organization (or entire industry) are equally important.

The transition from the traditional vertically integrated business and IT structures towards SOA presents a number of important challenges. These challenges range from purely technical issues such

as performance of the SOAP protocol and the maturity Web Services standards to business issues that include considerations of skills availability and SOA infrastructure costs. While the core Web Services standards (i.e. XML, SOAP, WSDL, and UDDI) are relatively mature and stable, many of the additional standards that address important issues such security and reliability (e.g. WS-Coordination, WS-Atomic Transaction, WSDM, WS-Reliability, etc.) are still under development. The ever increasing number of Web Services standards and the complexities and politics of the standardization process make it difficult for vendors to maintain conformance and at the same time to deliver stable technology platforms.

In order to achieve the benefits of reuse, extendability and responsiveness to new business requirements, services must be specified at the correct level of abstraction and granularity. It is not sufficient to insist that services should be coarse-grained and have well-defined interfaces; what is required is a comprehensive design methodology that guides the process of service design and produces reusable services that can be used as building blocks for business-level composite services. Such methods are still the subject of extensive research and have not been tested on large-scale development projects. Perhaps the greatest challenge to integration projects in business to business environments is the disparate data semantics used internally by individual partner organizations. The use of coarse-grained XML messages, as recommended by SOA experts does little to alleviate this problem. What is required is an industry-wide agreement on data structures and semantics, and this is not easy to achieve in practice.

4 Developing Service-Oriented Applications

In addition to a mature and stable technology platform based on Web Services standards, successful implementation of enterprise SOA requires support for the entire service development life-cycle, including service modeling, service design, and implementation of Web Services applications. In this section we briefly discuss these issues.

4.1 Service Modeling and Design

The transition towards service-oriented computing necessitates re-evaluation of design methodologies that are used in the construction of enterprise applications. Service modeling methodology has to support business-level conceptual modeling of services and their relationship to business processes, capturing both functional and non-functional requirements. Business Process Modeling (BPM) and Object-Oriented Analysis and Design (OOAD) are well established techniques; however as experience from SOA implementation projects suggest the methods only partially support the architectural patterns currently emerging under the SOA [Zimmerman, 2004]. Service modeling methodologies are still evolving and opinions differ about the precise nature of the relationship between services and business processes on one hand, and services and components on the other hand. Enterprise service models need to be able to capture internally and externally provided services and model different types of outsourcing strategies such as Business Process Outsourcing, application services sourced from ASP providers (Application Service Providers), etc. Both core and supporting business processes should be modelled as services, specifying service interfaces using SLA's (Service Level Agreements). The model should allow the evaluation of critical success factors associated with alternative outsourcing strategies [Feuerlicht and Vorisek, 2004].

Service design is concerned with transforming the service model produced during analysis into a set of design specifications and artifacts such as service interface specifications and composition/choreography workflows that satisfy specific application requirements. Service design needs to determine what constitutes a service and its operations, and make decisions about the level of service aggregation. Correct design is crucial as using coarse-granularity services inhibits reuse, and fine-grained services cause higher network overheads and more complex interactions dialogues. Proponents of the message-oriented (document-centric) approach generally recommend the use of

coarse-grained service interfaces that allow a single request to implement a complete high-level business function (e.g. airline flight booking) improving performance and avoiding issues related to high latency and poor reliability of the existing Internet infrastructure. However, using services with coarse-grained interfaces externalizes complex data structures creating interdependencies, and impacting on service reusability. Furthermore, coarse-grained service operations are less flexible and more difficult to evolve as a number of business functions are typically combined and implemented as a single operation. Frequently, such services exhibit overlapping functionality (i.e. lack of orthogonality) and require that a number of operations are modified when requirements change, making application maintenance difficult in practice.

The service oriented design is an active research area with a number of different approaches being proposed. The approaches can be broadly classified into methodologies based on object-oriented design [Ambler, 2001], [Ambler, 2002], [Levi, 2002], [Luo, 2005], methods for transformation of industry domain specifications [Masud, 2003], and business process transformation approaches [Papazoglou, 2002], [Leymann, 2001]. For example, Papazoglou and Yang describe a design methodology that gives a set of service design guidelines based on the principles of minimizing coupling and maximizing cohesion to ensure that the resulting services are self-contained, modular, extendable and reusable. The methodology produces definition of WSDL Web Service interfaces and WSFL service flow models, and also includes non-functional service design guidelines that relate to service provisioning strategies and service policy management models. Other methods rely on data engineering techniques for the design of service interfaces and message structures [Feuerlicht, 2005], [Feuerlicht, 2005a]. At present, while there is some agreement about the basic design principles there are no widely accepted design methodologies that can guide designers of Web Services applications.

4.2 Service Implementation

The above section (section 4.1) focused on methodologies for modeling and design of services. In addition to comprehensive modeling and design methodologies, developers of service oriented applications need mature and stable development and deployment platforms. Transition to SOA cannot take place in the absence of application development environments that provide comprehensive support for the various SOA life-cycle stages. The Service Component Architecture (SCA) is an industry effort by BEA, IBM and Oracle to provide a model for the assembly of business solutions from collections of individual services [BEA, IBM, and Oracle, 2005]. SCA is a set of specifications that describe a model for building SOA applications using open standards (i.e. Web Services). The SCA model supports the development of business application code in the form of components that implement business logic and externalize service interfaces. Components consume functions offered by other components via service interfaces, called service references. SCA defines two implementation phases: implementation of components which provide and consume services, and assembly of components into business applications, by connecting services via references. The emphasis is on increasing the level of abstraction and focusing on the business problem, minimizing the direct use of low-level APIs, and access methods used to invoke services. The SCA Assembly Model supports both tightly coupled and loosely coupled service-oriented systems. Services can be implemented using conventional programming language (e.g. Java or C++) , XML-based languages (e.g. BPEL and XSLT), or query languages (SQL and Xquery). SCA supports different interaction styles, including asynchronous message-oriented styles, and synchronous RPC style. Given the industry momentum behind SCA it is likely that the vendors will incorporate support for SCA based development into their products.

5 Conclusions

A key benefit of SOA is that it enables a close alignment of IT architecture with the business requirements of an organization and at the same time facilitates high-levels of business process

automation. While SOA implementation can deliver short-term, incremental benefits in integration projects by wrapping existing application components as services, the main benefit of SOA is that it enables organizations to participate in the emerging world of service-oriented computing. Enterprise SOA is essential for effective implementation of applications spanning the boundaries of organizations and the utilization of application services provided by external service providers.

It is important to remember that enterprise applications are implemented using technologies supplied by IT vendors, not on the basis of a set of architectural concepts favored by analysts. The benefits of SOA can be only realized if solid technological solutions are available to support the development and deployment of service-oriented applications. Another determinant for successful adoption of this new approach is level of the skills and knowledge of IT professionals involved with SOA projects. People involved with introducing SOA into organizations need to be able to make informed decisions about technology selection, the timing of adoption of various Web Services standards and solutions, and successfully manage large-scale SOA projects.

Literature

[Ambler, 2001] Ambler, S. W. 2001, Web Services Programming Tips and Tricks: Introduction to UML Sequence Diagrams [Online]. Available: http://www-106.ibm.com/developerworks/webservices/library/ws-tip-uml3/ [Accessed 19 May 2005].

[Ambler, 2002] Ambler, S.W. Deriving Web Services from UML models, Part 1: Establishing the process, 2002, Available on: http://www-106.ibm.com/developerworks/webservices/library/ws-uml1/

[BEA, IBM, and Oracle, 2005] Service Component Architecture: Building Systems using Service Oriented Architecture, A Joint Whitepaper by BEA, IBM, and Oracle, November 2005, Available online: http://xml.coverpages.org/ni2005-12-07-a.html

[Feuerlicht, 2005] Feuerlicht G., Application of Data Engineering Techniques to Design of Messages for Service-Oriented, Proceedings of the First International Workshop on Design of Service-Oriented Applications (WDSOA'05), Amsterdam, The Netherlands, December 12, 2005, IBM Research Report, RC23819 (W0512-29), December 6, 2005

[Feuerlicht and Vořišek, 2004] Feuerlicht, G., Vořišek, J.: Utility Computing: ASP by another name, or a new trend?, Proceedings of the 12th International Conference Systems Integration 2004, June 14-15, 2004, Prague, 2004, pages 269-279, ISBN 80-245-0701-3

[Feuerlicht , 2005a] Feuerlicht, G., Design of Service Interfaces for e-Business Applications using Data Normalization Techniques, Journal of Information Systems and e-Business Management, Springer-Verlag GmbH, 26 July 2005, pages 1-14, ISS:1617-98

[Feuerlicht and Vořišek, 2006] Feuerlicht G. and Vořišek J., Software-as-a-Service Model for Enterprise Computing, Encyclopedia of e-Technologies and Applications, Edited by Mehdi Khosrow-Pour, (16 pages), Invited paper to be published by Idea Group in March 2006, ISBN 1-59140-799-0

[Forrester, 2005] Randy Heffner, Your Strategic SOA Platform Vision, Forrester Research, March 29, 2005

[Gartner, 2003] Yefim V. Natis, Service-Oriented Architecture Scenario, ID Number: AV-19-6751. 16 April 2003

[IDC, 2006] Worldwide SOA-Based Services 2006-2010 Forecast: Demand for Services Continues as Adoption Expands Across Industries and Geographies (IDC #35072)

[Levi, 2002] Levi, K. and A. Arsanjani (2002) A goal-driven approach to enterprise component identification and specification. Communications of the ACM. Vol. 45:(10). (2002) 45 - 52

[Leymann, 2001] Leymann, F. Web Services Flow Language (WSFL 1.0), 2001.http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.p

[Luo, 2005] Luo, M. et al. 2005, Service-Oriented Business Transformation in the Retail Industry Part 1: Apply SOA to Integrate Package Solutions and Legacy Systems [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-retail1/ [Accessed 15 April 2005].

[Maron, 2005] SOA and the Future of Application Development, Bill Eidson, Jonathan Maron, Greg Pavlik, Rajesh Raheja, Proceedings of the First International Workshop on Design of Service-Oriented Applications (WDSOA'05), Amsterdam, The Netherlands, December 12, 2005, IBM Research Report, RC23819 (W0512-29), December 6, 2005

[Masud, 2003] Masud, S. 2003, Use RosettaNet-Based Web Services, Part 2: BPEL4WS and RosettaNet [Online]. Available: http://www-106.ibm.com/developerworks/webservices/library/ws-rose2/ [Accessed 19 May 2005].

[Papazoglou, 2002] Papazoglou, M. P. & Yang, J. 2002, 'Design Methodology for Web Services and Business Processes', in Proceedings of the 3rd VLDB-TES Workshop, vol. 2444, eds A. Buchmann, F. Casati, L. Fiege, M.-C. Hsu, et al., Springer, Hong Kong, pp. 54-64.

[W3C, 2004] Web Services Architecture, W3C Working Group Note, 11 February 2004, Available Online: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#service_oriented_architecture

[Zimmerman, 2004] Zimmerman, O. Elements of Service-Oriented Analysis and Design: An interdisciplinary modeling approach for SOA projects, 2004 Available Online: http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/,