Web Services, Service-Oriented Computing, and Service-Oriented Architecture: Separating Hype from Reality

Article *in* Journal of Database Management · July 2008

DOI: 10.4018/jdm.2008070103 · Source: DBLP

CITATIONS
101

READS
2,763

2 authors:

John Erickson
University of Nebraska at Omaha
42 PUBLICATIONS **980** CITATIONS

SEE PROFILE

Keng Siau
City University of Hong Kong
556 PUBLICATIONS **12,944** CITATIONS

SEE PROFILE

IGI PUBLISHING **ITJ4248**

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.igi-global.com

This paper appears in the publication, Journal of Database Management, Volume 19, Issue 3
edited by **Keng Siau © 2008, IGI Global**

Web Services, Service-Oriented Computing, and Service-Oriented Architecture:
Separating Hype from Reality

John Erickson, University of Nebraska - Omaha, USA

Keng Siau, University of Nebraska - Lincoln, USA

ABSTRACT

Service-oriented architecture (SOA), Web services, and service-oriented computing (SOC) have become the buzz words of the day for many in the business world. It seems that virtually every company has implemented, is in the midst of implementing, or is seriously considering SOA projects, Web services projects, or service-oriented computing. A problem many organizations face when entering the SOA world is that there are nearly as many definitions of SOA as there are organizations adopting it. Further complicating the issue is an unclear picture of the value added from adopting the SOA or Web services paradigm. This article attempts to shed some light on the definition of SOA and the difficulties of assessing the value of SOA or Web services via return on investment (ROI) or nontraditional approaches, examines the scant body of evidence empirical that exists on the topic of SOA, and highlights potential research directions in the area.

Keywords: *service-oriented architecture; service-oriented computing; Web services*

INTRODUCTION

Service-oriented architecture (SOA); Web services; mash-ups; Ajax; Web 2.0; some of their underlying middleware realization schemas such as SOAP (simple object access protocol), UDDI (universal description, discovery, and integration), XML (extensible markup language), and CORBA (common object request broker architecture); and many other ideas or approaches to cutting-edge information system architectures have become the buzzwords of the day for many in the business world and also in the IT and IS communities. It is quite difficult, perhaps nearly impossible, to pick up any relatively current practitioner publication without encountering an article focusing on at least one of the above topics. A recent library database search using keywords *service-oriented architecture*, *Web services*, and *SOA* resulted in 800-plus returns. Further investigation

revealed that roughly 25 of those 800 articles were sourced in research journals while the other (still roughly 800) articles were all from more practitioner-oriented sources.

When it comes to adopting and implementing SOA, it appears that businesses are doing it at astounding rates. Of course, what they are actually doing, even though they may say that their efforts represent a move toward service-oriented architecture, may not match anyone else's definition of SOA but their own. Furthermore, how can SOA be defined, and how can we define the benefits of moving toward such architectures? It seems that there is little agreement among practitioners and researchers alike as to a standard definition of SOA.

Worse still, a growing number of practitioners are now beginning to question the business return of some of the approaches. For example, Dorman (2007), Havenstein (2006), Ricadela (2006), and Trembly (2007) indicate that there is doubt emerging as to the real value of SOA to adopting businesses and organizations. Perhaps the question of return on investment (ROI) should not be that surprising since it sometimes seems that each organization has its own definition of what SOA really is.

This article attempts to reach for a clearer understanding of what SOA really is, and proposes some possible areas of research into SOA that could help clear up some of the definitional confusion, which could in turn help lead to better understanding of ROI as it relates to SOA. First is the introduction. Second, the article provides existing definitions of SOA, Web services, and some of the related and underlying technologies and protocols. The next section combines the various definitions of SOA into a more coherent form, while the section after that proposes ideas about what SOA should be. The fifth section discusses research possibilities and provides recommendations for future research efforts. Next, we look at ways of measuring and justifying SOA and SOC (service-oriented computing) success. Finally, we conclude the article.

BACKGROUND AND HISTORY OF SERVICE-ORIENTED ARCHITECTURE

A minimum of nine formal definitions of SOA exist as of this writing, from sources such as the Organization for the Advancement of Structured Information Standards (OASIS), the Open Group, XML.com, Javaworld.com, Object Management Group (OMG), the World Wide Web Consortium (W3C), Webopedia, TechEncyclopedia, WhatIs.com, and Webopedia.org. In addition, many other definitions put forth by numerous industry experts, such as those from IBM, further cloud the issue, and worse yet, other formal definitions might also exist. In other words, the concept of service-oriented architecture appears in many ways to be a virtually content-free description of an IT-based architecture. It is not our intent here to add yet another definition to this already crowded arena of definitions, but to try to cull the common, base meanings from the various distinct definitions.

Prior to about 2003, the term service-oriented architecture was not in general use for the most part, according to Wikipedia ("SOA," 2007). However, since that time, SOA has exploded nearly everywhere in the business and technology world. SOA appears to derive or develop in many cases from more basic Web services. These services can include enabling technologies such as SOAP, CORBA, EJB (Enterprise Java Beans), DCOM (distributed component object model), and even SIP (session-initiated protocol) among many others; services may also include other middleware created with XML (Lee, Siau, & Hong, 2003; Siau & Tian, 2004; Sulkin, 2007; Walker, 2007).

Service-Oriented Architecture Definitions

The Open Group (2007) defines SOA as "an architectural style that supports service orientation." The definition goes on to also include descriptions of architectural style, service orientation, service, and salient features of SOA. OASIS defines SOA as "a paradigm for

organizing and utilizing distributed capabilities that may be under the control of different ownership domains." The OASIS definition includes what they call a "reference model" in which the details of the definition are expanded and formalized. The Object Management Group (2007) defines SOA as "an architectural style for a community of providers and consumers of services to achieve mutual value." OMG adds that SOA allows technical independence among the community members, specifies the standards that the (community) members must agree to adhere to, provides business and process value to the (community) members, and "allows for a variety of technologies to facilitate (community) interactions" (OMG, 2007).

W3C (2007) defines SOA as "a form of distributed systems architecture that is typically characterized by…a logical view, a message orientation, a description orientation, granularity and platform neutrality." W3C adds details describing what it means by logical view, message and description orientations, granularity, and platform neutrality. XML.com (2007) defines SOA as follows:

SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software agents on behalf of their owners.

The Javaworld.com SOA definition, composed by Raghu Kodali (2005), is as follows: "Service-oriented architecture (SOA) is an evolution of distributed computing based on the request/reply design paradigm for synchronous and asynchronous applications." Kodali also goes on to describe four characteristics of SOA. First, the interfaces composed in XML, using WSDL (Web services description language), are used for self-description. Second, XML schema called XSD should be used for messaging. Third, a UDDI-based registry maintains a list of the services provided. Finally, each service must maintain a level of quality defined for it via a QoS (quality of service) security requirement.

Finally, IBM proposes that SOA "describes a style of architecture that treats software components as a set of services" (UNL-IBM System in Global Innovation Hub, 2007). Furthermore, it insists that business needs should "drive definition" of the services, and that the value proposition be centered on the reusability and flexibility of the defined services.

SERVICE-ORIENTED ARCHITECTURE

We begin the SOA discussion with an overview of SOA provided by Krafzig, Banke, and Slama (2005). They proposed a three-level hierarchical perspective on SOA in which Level 1 includes the application front end, the service, the service repository, and the service bus (SB). Accordingly, only the service child has children, consisting of the contract, implementation, and interface. Finally, the last level of the proposed hierarchy is composed of business logic and data, children of implementation. The next subsections will discuss the general ideas of the elements included in the hierarchy proposed by Krafzig et al. described previously. This is not to recommend adoption of the hierarchy and description as the final description of SOA, but rather as a framework for discussing the meaning of SOA for the remainder of this article.

Application Front End

This part of SOA comprises a source-code interface, and in SOA terminology, it is referred to as the application programming interface (API). In accordance with most commonly accepted design principles, the underlying service requests, brokerage (negotiation), and provision should be transparent to the end user.

Service Repository

The service repository could be thought of as the library of services offered by a particular SOA. This would likely consist of an internal system that describes the services, and provides the means in the user interface to call a particular service. UDDI could be seen as a realization of the service repository idea. UDDI is a global

registry that allows businesses to list themselves on the Internet. UDDI is platform independent and XML based. The point of UDDI is for businesses to list the Web or SOA-type services that they provide so that other companies searching for such services can more easily locate and arrange to use them.

Service Bus

The SB, more commonly referred to as the enterprise service bus (ESB), provides a transportation pathway between the data and the end-user application interface. Using an ESB does not necessarily mean SOA is being implemented, but ESB or some sort of SB use is almost always part of an SOA deployment. According to Hicks (n.d.), Oracle's idea of an ESB includes multiple protocols that "separate integration concerns from applications and logic." What this means is that ESBs have now become commercialized, and can be licensed for use much like other UDDI-based services. So, companies searching for ESB solutions as part of an SOA effort now have multiple choices and do not necessarily have to re-create the wheel by building their own ESB.

Common Services

It seems apparent from many of the SOA definitions that many of the technologies included in an SOA definition, and by default SOA implementations, are established and conventional protocols. To better understand the services provided in many SOA definitions, a brief explanation of some of the more commonly used underlying technologies is provided. A particular service may or may not be explicitly Web based, but in the end it matters little since the services provided by the architecture should be transparently designed, implemented, and provided. The general consensus from most involved in Web services is that the services are meant to be modular. This means that no single document encompasses all of them, and furthermore, that the specifications are multiple and (more or less) dynamic. This results in a small number of core specifications. Those core services can be enhanced or supported by other services as "the circumstances and choice of technology dictate" ("Web Service," 2007).

XML allows users to define and specify the tags used to capture and exchange data, typically between distinct and usually incompatible systems from different companies or organizations. This means that XML is a good example of middleware; it also means that XML enables Web services. XML was one of the initial drivers that provided the ability to conduct e-business for many businesses in the Internet era. XML cannot really be considered a service, but as the language used to write many of the Web services or service stack protocols.

SOAP, like all protocols, consists of a set list of instructions detailing the action(s) to be taken in a given circumstance. SOAP is designed to call, access, and execute objects. The original SOAP was typically for communications between computers, and usually involved XML-based messages. SOAP and its underlying XML programming comprised one of the first Web service communication stacks. One of the original Web services that SOAP provided was called remote procedure call (RPC), which allowed a remote computer to call a procedure from another computer or network. More recently, SOAP has taken on a somewhat modified meaning so that the acronym now means service-oriented architecture protocol. In both cases, what SOAP does is to use existing communications protocols to provide its services. The more common early SOAP contracts included XML applications written for HTTP (hypertext transfer protocol), HTTPS (HTTP over secure socket layer), and SMTP (simple mail transfer protocol), among others. It should be apparent from these that many early SOAP implementations involved e-commerce or e-business applications, which means that the concern at the time when many applications were first developed was to move sales and other data collected in Web portals to back-end data stores.

CORBA is an OMG-developed standard that allows different software components that are usually written in different languages and

installed on different computers to work together (Zhao & Siau, 2007). CORBA was developed in the early 1990s, and while not overtly an SOA at the time, it actually performs many of the functions in an SOA, using an IIOP- (Internet inter-orb protocol) based service stack.

EJB is a component typically situated on the server that "encapsulates the business logic of an application" ("EJB," 2007). EJB enables the creation of modular enterprise (and other) applications. The intent of EJB is to facilitate the creation of middleware that acts as a go-between tying front-end applications to back-end applications or data sources.

SIP is a signaling protocol designed for use in telecommunications at the application layer. It has generally become one of the primary protocols used in VoIP (voice over Internet protocol), H.323, and other communications standards. SIP can be seen as a primary provider of Web services for Internet-based voice communications such as VoIP (Sulkin, 2007).

Contract (Services)

Components of a service contract typically include primary and secondary elements. The primary elements consist of the header, functional requirements, and nonfunctional requirements. Subelements for the header consist of the name, version, owner, RACI, and type. Under functional requirements are functional requirement descriptions, service operations, and invocation. Nonfunctional requirements include security constraints, QoS, transactional requirements (the service part of a larger transaction), service-level agreement, and process ("SOA," 2007). The contract generally includes metadata about itself, who owns it, and how it is brokered, bound, and executed.

Interface

At this level of service provision, the interface referred to is a segment of code that connects the service with the data and/or business logic (process). The interface describes how data will be moved into and out of the data source by the service, and must be designed to comply with the physical (data, data structures, etc.) and process (business logic) requirements of the existing and/or legacy system.

Implementation

The implementation specifies the contract and interface to be used for each service requested, and contains the direct pathway into the data and business logic.

Architecture

The service component of SOA has been discussed, though admittedly at a high level. However, the architecture component has not yet been addressed and it will be helpful to speak briefly about the architecture segment of SOA. Architecture in general refers to the art (or science) behind the design and building of structures. Alternatively, an architecture may refer to a method or style of a building or a computer system. So, if SOA is taken literally as a description of its function, it could be taken to mean a structured way of organizing or arranging the services in a business or organization.

SOA FRAMEWORK

It is apparent from the existing definitions and models that service-oriented architecture is commonly seen as an architecture or way of assembling, building, or composing the information technology infrastructure of a business or organization. As such, SOA is not a technology in itself; rather, it is a way of structuring or arranging other technologies to accomplish a number of other tasks. This naturally leads to the problem of a multiplicity of definitions of SOA since many relatively similar structural arrangements of services are possible. Many of the definitions also indicate that the arrangement and relationships between modules should be loosely coupled rather than tightly coupled. This allows for customization of services based on need, and on-demand rather than some predetermined structure, but the downside is that it also leads toward a plethora of definitions and approaches to SOA implementation.

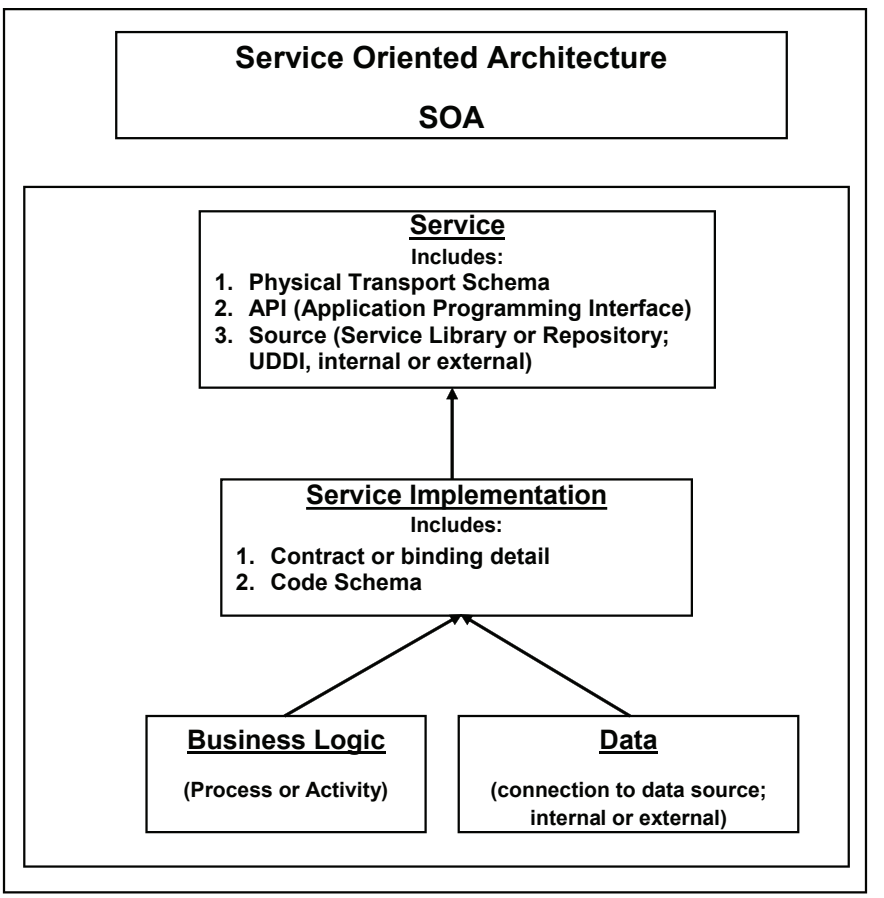
Some of the common features that seem sensible to include in a formal definition of

SOA would relate to a common framework, such as that specified by Krafzig et al. (2005) or one of the other standards bodies. In other words, a framework would include metadata describing the various important features of SOA, how those features can be arranged, and the libraries or location of services that allow adopting organizations to arrange bindings or contracts between themselves and the service provider, independent of whether the service provider is internal or external. We propose the framework depicted in Figure 1 as a starting point for visualizing SOA.

Several of the standards bodies have taken a stance in creating or calling for a metamodel, at least in some form. Among them are the Open Group, OASIS, OMG, W3C, and to a lesser extent industry-related bodies such as Javaworld.com, XML.com, IBM, and Oracle.

UDDI has become a very well-known structured repository for services and service components, which speaks to the universality of the library or centralized database of services. However, more standardization efforts will be necessary to enhance the interoperability of UDDI.

Figure 1. SOA framework



It also appears, especially with the industry definitions of SOA, that the contracts, bindings, interfaces, service buses, and other implementation-related portions of SOA are important elements to be considered when attempting to give an overall definition of SOA. This unfortunately could easily represent a stumbling block in garnering consensus on a definition of SOA since each of these companies has invested significant time, human, and other likely resources toward development of their specific pieces of the SOA pie. Each company has invested heavily and thus will likely be less willing to risk that investment and any potential return and customer lock-in in order to simply agree on standards. We observed a similar occurrence of this type of behavior in the recently ended format war in the high-definition DVD market. Similarly, if the standards bodies have political or industry leanings, agreement on a common SOA definition and standards could be difficult to achieve.

Another more recent development comes from Shah and Kalin (2007). They proposed that organizations adopting SOA follow a specific path based on an analysis of business challenges, including SOA business drivers and IT barriers. This led them to speculate that a specific adoption model be used to guide the SOA implementation process. They indicated that an ad hoc SOA model is better where the benefits of new services are specific to each individual service, where the technologies may be inconsistently applied (different implementations for the same service in different projects), where services cannot be reused, and where the increases in technological complexity translate into decreased system response times. Shah and Kalin ended with a call for a strategy- or program-based SOA adoption model that is situational.

We propose that a common definition of SOA is possible and necessary, and call for negotiations among interested bodies with the aim of reaching a common definition of SOA. We realize that in practice it might prove difficult or even nearly impossible to expect such a consensus to be arrived at, but a common definition and structure of SOA would go a long way toward dealing with some of the confusion, misinformation, and hype regarding the entire subject. Difficult though it might be to expect this, a realization that SOAP, CORBA, RPC, and XML among many other technological tools have reached a point of relative agreement amongst users if not ubiquity, at least related to their underlying standards, should provide some evidence that agreements can be reached. Next, we will examine SOA from the research perspective.

POSSIBILITIES FOR RESEARCH

Research into SOA is extremely limited at this point in time. What studies exist can be classified into several distinct categories. The first includes exploratory or recommendation-type efforts that propose various means to approach SOA implementation. These investigations may or may not include proprietary industry software, but most of these research efforts propose the use of patterns or blueprints and a metamodel of SOA as a means to understanding the SOA perspective. Second, in this category are research proposals that examine company-specific technologies or tools (i.e., IBM proposing the use of Rational Software, including the Rational Unified Process) in relation to SOA design and implementation. Neither of the first two types of SOA research generally involve ideas on how to measure SOA in terms of success or failure, or even suggest metrics. Finally, the third type of research articles focus on empirical research.

SOA Development or Deployment Patterns and Blueprints, and the Meta-Approach

Stal (2006) took a roughly similar approach to what we are attempting to do in this article; he advocated using architectural patterns and blueprints (software engineering patterns) as a means to enable or foster efficient deployment of SOA. He supported loose coupling of services in a registry or library to the extent that he thought

that removing the services' dependency on the registry's or provider's distinct location would benefit the deployment of SOA. Stal maintained that this would eliminate, or at least minimize, a layer in the SOA framework. He also proposed a more tightly defined and controlled integration of middleware using XML or similar tools. Basically, Stal suggested a metamodel and pattern approach to defining SOA, but did not suggest what the research might accomplish or how the research into SOA would be framed. Kim and Lim (2007) also proposed a distinct means to implementing SOA, using in this instance, business process management, in addition to a variant of the SOA framework specifically dealing with the telecommunications industry. Similar to Stal, Kim and Lim did not propose empirical research into SOA, but rather focused on implementation and standards in a specific industry.

Shan and Hua (2006) proposed an SOA approach for the Internet banking industry. They also compiled a list of patterns that have been proven successful for other online service industries. However, the models they used and ended up with are very detailed regarding how SOA should be implemented for first online companies in general, and then Internet banking specifically. This again does not propose or frame specific research but rather suggests an implementation approach and a structure for SOA.

The ESB is explained in detail, but from a general perspective rather than a company-specific approach in Schmidt, Hutchison, Lambros, and Phippen's (2005) expository. The article is informative regarding ESB implementation and design patterns, but it is not research oriented.

Crawford, Bate, Cherbakov, Holley, and Tsocanos (2005) proposed a different way to structure SOA, what they called on-demand SOA. They essentially proposed an even looser coupling of services and their connecting elements than in other perspectives of SOA. They argued that this would allow much more flexibility to the adopting organizations and the end users.

Company-Specific and Commercial Tool-Based SOA Deployment

Brown, Delbaere, Eeles, Johnston, and Weaver (2005) presented an industry-oriented perspective on the SOA puzzle. They suggested an approach to service orientation using the proprietary IBM Rational platform. Their recommendations follow similar paths as some previous research, but are also filtered through the IBM Rational lens. The article is primarily illustrative in nature, suggesting how to best implement SOA using IBM Rational tools. In a similar vein, Ferguson and Stockton (2005) also detailed IBM's programming model and product architecture.

De Pauw, Lei, Pring, and Villard (2005) described the benefits of Web Services Navigator, a proprietary tool created to provide a better visualization of SOA and Web services in a loosely coupled architecture. The tool can help with design-pattern, business-logic, and business-process analysis, and thus help with SOA architecture design and implementation.

Jones (2005) suggested that SOA, service, and Web service standards were "on the way" and provided a list of existing tools, such as UML (Unified Modeling Language) and/or the rational unified process that could aid the SOA (or service) design process. However, he also advocated the push toward formal definitions of such SOA basics as services, to the end of providing a more coherent and cohesive structure that he thought would enhance the ability of developers and adopters to understand and deploy SOA.

Research-Based Perspectives on SOA

Chen, Zhou, and Zhang (2006) proposed an ontologically based perspective on SOA, Web services, and knowledge management. They attempted, with some success, to integrate two separate research streams into one. They presented a solution to show that semantic- and syntactic-based knowledge representations could both be depicted with a comprehensive

ontology that also described Web service composition. While their framework represents a step toward automated (Web) service composition, more research is still needed.

Borkar, Carey, Mangtani, McKinney, Pate, and Thatte (2006) suggested a way of handling XML-based data in an SOA or service environment. Their idea involved the use of data both able to be queried and unable to be queried, and would necessarily also involve XML-formatted data. This represents empirical research into a part of SOA, namely, the underlying services, and is at least a step in the right direction, although it does not enter the realm of research into the efficacy or ROI of SOA.

Duke, Davies, and Richardson (2005) recommended and provided details on using the Semantic Web to organize an organization's approach to SOA and Web service orientation. They suggested that combining the Semantic Web and SOA into what they called Semantic SOA would provide benefits to adopting organizations. Then they further proposed an ontological model of the Semantic SOA, attempting essentially to create a meta-metamodel of SOA using their experience with the telecommunications industry as a case example. This is one of the few high-level articles that can also be seen as empirical research.

Zhang (2004) explored the connection between Web services and business process management, and described the modular nature of the service (and Web service) perspective. He detailed the software industry's approach to Web services and provided evidence that standards development would quickly mature, beginning in 2005. He maintained that once standards were agreed upon, a connection to business process management would be easier to sell to businesses. Zhang also developed a prototype e-procurement system that composed external services to operate.

Malloy, Kraft, Hallstrom, and Voas (2006) developed an extension to WSDL. They insisted that Web services' specifications were "typically informal and not well-defined," and proposed what they called an intermediate step between requiring more formal and rigorous service specifications and the informal nature of the existing service specifications. They accomplished this balance by extending WSDL to include support for application arguments that would help automate and expand the ability of services to operate in multiple environments. They provided an example of how their WSDL extension could allow a single service to function successfully in different applications using multiple zip code formats (five vs. nine digits, and hyphens vs. no hyphens).

Verheecke, Vanderperren, and Jonckers (2006) proposed and developed a middleware level that they called the Web services management layer (WSML). They saw the primary advantage of their approach in that it provided a reusable framework. They further believed that the use of their framework would enable "dynamic integration, selection, composition, and client-side management of Web Services in client applications" (p. 49). They were aware that their approach could cause some problems in a distributed system since implementation of it resulted in a centralized architecture.

Hutchinson, Henzel, and Thwaits (2006) described a case in which an SOA-based system was deployed for a library extension collaboration project. Much of the case details the SOA approach itself, and explains the experiences of the project developers and implementers. They noted that while the SOA architecture could be expected to reduce the operational maintenance costs overall, the way the system was specified and delivered in this particular case might require more work from IT to keep some services, such as flash players, up to date. While the authors did not specifically mention it in the article, perhaps a more loosely coupled architecture might alleviate some of those operational maintenance costs.

Li, Huang, Yen, and Cheng (2007) proposed a methodology to migrate the functionality of legacy systems to a Web services or SOA architecture. They used a case study to investigate the efficacy of their proposed methodology, finding that while it was possible to make such a migration from legacy systems to SOA (or Web services), the changes that it required

from the organization were considerable, and some process reengineering would likely be necessary.

MEASURING SOA AND SOC SUCCESS

Another tricky issue in SOA and SOC implementation is the measurement or evaluation of success. Traditionally, software (or system) successes and failures have been estimated by the usual suspects: traditional measures such as ROI, net present value (NPV), breakeven, internal rate of return (IRR), or other similar financially based approaches. Similarly, software itself has usually been measured in terms of errors or productivity via numeric methodologies such as lines of code, COCOMO (constructive cost model), and similar estimation techniques. These approaches are all based firmly on the idea that if we can assign some number to a system, then we can compare them across projects, systems, or organizations. The problem is analogous to the question often asked regarding enterprise resource planning (ERP) systems: If all of the Fortune 100 companies implement the same piece of software, such as SAP, then what allows one organization to differentiate itself from another if they have standardized on SAP's best processes and best practices? One way to answer that question is to examine other measures of success such as competitive advantages (Siau, 2003), competitive necessity, flexibility, agility (Erickson, Lyytinen, & Siau, 2005), nimbleness, responsiveness, and other relevant intangibles. We would even propose that the best way to evaluate SOA or SOC implementation is not ROI. Intangible but critical factors such as competitive necessity, agility, on-demand abilities, and responsiveness should be the decisive factors.

Nah, Islam, and Tan (2007) proposed a framework and critical success factors for estimating the success of ERP implementations. They empirically assessed a variety of implementation success factors including top-management support, project team competence, and interdepartmental cooperation, among many others. While the study answered a number of important questions regarding ERP implementations, the issue of assessing intangibles in terms of success factors remains a problem, not only for ERP-type implementations but also for other system types as well, especially for SOA since the SOA approach can be seen as an alternative in many ways to ERP.

Langdon (2007) noted that while many economic-based studies indicate that IT projects add value at the macrolevel, little has been done to assess how value is added at the more micro or individual project level. Specifically, Langdon proposed and evaluated a research model that included (IS) integration and flexibility as capabilities that could lead to IT business value. Of course, flexibility and integration are only two components of a larger IT capabilities structure, but the study indicates that the first steps have been taken to study intangibles in the context of an IT systems development project.

Two intangibles in the IT success-factor context are the oft-cited agility or nimbleness of a company or organization. An entire genre of systems development has emerged based on the principle of agility. However, there is little empirical evidence supporting the value added from such development approaches (Erickson et al., 2005). Since a growing number of SOA installations are constructed as ad hoc, which is in a basic sense agile, we propose that in environments where agility and nimbleness are important, so in turn are SOA and SOC important.

CONCLUSION

From the literature, it appears that only a few efforts can be said to be empirical research. A majority of the research efforts involved created tools or language extensions that would increase the interoperability of services, while other research proposed standards modifications. Many of the remaining articles published proposed new tools or the use of existing proprietary tools, described an approach to SOA from specific perspectives, or proposed model or metamodel changes. A limited number of case studies detailing SOA, Web services, or service deployments or implementation efforts provide

experience reports on how best to implement such systems.

As far as we can determine, virtually no research has been formally done regarding the benefits and drawbacks of SOA or Web services. Two problems with this are likely to revolve around the nebulous nature of SOA and Web services in terms of the widely varying definition and the emerging standards issue. An effort to identify SOA and Web services metrics would help to get research into this area started.

Another area of interest involving SOA and Web services adoption is the cultural and structural impacts on the organization or business. A number of articles note the importance of those elements, but little has been accomplished in terms of research specifically connecting SOA or Web services with cultural and structural changes in organizations.

A variety of standards bodies are working separately toward formal definitions including metamodels, and a number of SOA vendors, among them some of the very large and established software industry players, have emerged. While the effort toward standardization is direly needed and commendable, a more collaborative approach would, in our opinion, benefit the industry and implementing companies and organizations as well. The seeming result of the rather haphazard approach to SOA appears to indicate that an increasing number of implementing organizations are finding it difficult to assess the cost benefit of the entire services approach. Research efforts at this point appear to be in a similar state of disarray. Until a more coherent picture of SOA emerges, its image is likely to remain slightly out of focus, and research in the area is likely to remain somewhat unfocused as a result.

REFERENCES

Borkar, V., Carey, M., Mangtani, N., McKinney, D., Patel, R., & Thatte, S. (2006). XML data services. *International Journal of Web Services Research, 3*(1), 85-95.

Brown, A., Delbaere, M., Eeles, P., Johnston, S., & Weaver, R. (2005). Realizing service oriented solutions with the IBM Rational Software Development Platform. *IBM Systems Journal, 44*(4), 727-752.

Chen, Y., Zhou, L., & Zhang, D. (2006). Ontology-supported Web service composition: An approach to service-oriented knowledge management in corporate financial services. *Journal of Database Management, 17*(1), 67-84.

Crawford, C., Bate, G., Cherbakov, L., Holley, K., & Tsocanos, C. (2005). Toward an on demand service architecture. *IBM Systems Journal, 44*(1), 81-107.

De Pauw, Lei, M., Pring, E., & Villard, L. (2005). Web services navigator: Visualizing the execution of Web services. *IBM Systems Journal, 44*(4), 821-845.

Dorman, A. (2007). FrankenSOA. *Network Computing, 18*(12), 41-51.

Duke, A., Davies, J., & Richardson, M. (2005). Enabling a scalable service oriented architecture with Semantic Web services. *BT Technology Journal, 23*(3), 191-201.

EJB. (2007). *Wikipedia.* Retrieved October 12, 2007, from http://en.wikipedia.org/wiki/Ejb

Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: The state of research. *Journal of Database Management, 16*(4), 80-89.

Ferguson, D., & Stockton, M. (2005). Service oriented architecture: Programming model and product architecture. *IBM Systems Journal, 44*(4), 753-780.

Havenstein, H. (2006). Measuring SOA performance is a complex art. *Computer World, 40*(2), 6.

Hicks, B. (n.d.). *Oracle Enterprise Service Bus: The foundation for service oriented architecture.* Retrieved October 18, 2007, from http://www.oracle.com/global/ap/openworld/ppt_download/middleware_oracle%20enterprise%20service%20bus%20foundation_250.pdf

Hutchinson, B., Henzel, J., & Thwaits, A. (2006). Using Web services to promote library-extension collaboration. *Library Hi Tech, 24*(1), 126-141.

Jones, S. (2005). Toward an acceptable definition of service. *IEEE Software, 22*(3), 87-93.

Kim, J., & Lim, K. (2007). An approach to service oriented architecture using Web service and BPM in

the Telcom OSS domain. *Internet Research, 17*(1), 99-107.

Krafzig, D., Banke, K., & Slama, D. (2005). *SOA elements.* Prentice Hall. Retrieved October 2, 2007, from http://en.wikipedia.org/wiki/Image:SOA_Elements.png

Langdon, C. (2007). Designing information systems to create business value: A theoretical conceptualization of the role of flexibility and integration. *Journal of Database Management, 17*(3), 1-18.

Lee, J., Siau, K., & Hong, S. (2003). Enterprise integration with ERP and EAI. *Communications of the ACM, 46*(2), 54-60.

Li, S., Huang, S., Yen, D., & Chang, C. (2007). Migrating legacy information systems to Web services architecture. *Journal of Database Management, 18*(4), 1-25.

Malloy, B., Kraft, N., Hallstrom, J., & Voas, J. (2006). Improving the predictable assembly of service oriented architectures. *IEEE Software, 23*(2), 12-15.

Nah, F., Islam, Z., & Tan, M. (2007). Empirical assessment of factors influencing success of enterprise resource planning implementations. *Journal of Database Management, 18*(4), 26-50.

Object Management Group (OMG). (2007). Retrieved September 25, 2007, from http://colab.cim3.net/cgi-bin/wiki.pl?OMGSoaGlossary#nid34QI

Open Group. (2007). Retrieved September 25, 2007, from http://opengroup.org/projects/soa/doc.tpl?gdid=10632

Organization for the Advancement of Structured Information Standards (OASIS). (2006). Retrieved September 25, 2007, from http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

Ricadela, A. (2006, September 4). The dark side of SOA. *Information Week*, pp. 54-58.

Schmidt, M., Hutchison, B., Lambros, P., & Phippen, R. (2005). Enterprise service bus: Making service oriented architecture real. *IBM Systems Journal, 44*(4), 781-797.

Shah, A., & Kalin, P. (2007, July 6). SOA adoption models: Ad-hoc versus program-based. *SOA Magazine*.

Shan, T., & Hua, W. (2006). Service oriented solution framework for Internet banking. *Internet Journal of Web Services Research, 3*(1), 29-48.

Siau, K. (2003). Interorganizational systems and competitive advantages: Lessons from history. *Journal of Computer Information Systems, 44*(1), 33-39.

Siau, K., & Tian, Y. (2004). Supply chains integration: Architecture and enabling technologies. *Journal of Computer Information Systems, 44*(3), 67-72.

SOA. (2007). *Wikipedia.* Retrieved September 25, 2007, from http://en.wikipedia.org/wiki/Service-oriented_architecture#SOA_definitions

Stal, M. (2006). Using architectural patterns and blueprints for service oriented architecture. *IEEE Software, 23*(2), 54-61.

Sulkin, A. (2007). SOA and enterprise voice communications. *Business Communications Review, 37*(8), 32-34.

Trembly, A. (2007). SOA: Savior or snake oil? *National Underwriter Life & Health, 111*(27), 50.

UNL-IBM System in Global Innovation Hub. (2007). *Making SOA relevant for business.* Retrieved October 9, 2007, from http://cba.unl.edu/outreach/unl-ibm/documents/SOA_Relevant_Business.pdf

Verheecke, B., Vanderperren, W., & Jonckers, V. (2006). Unraveling crosscutting concerns in Web services middleware. *IEEE Software, 23*(1), 42-50.

Walker, L. (2007). IBM business transformation enabled by service-oriented architecture. *IBM Systems Journal, 46*(4), 651-667.

Web service. (2007). *Wikipedia.* Retrieved October 18, 2007, from http://en.wikipedia.org/wiki/Web_service

World Wide Web Consortium (W3C). (2007). Retrieved September 25, 2007, from http://colab.cim3.net/cgi-bin/wiki.pl?WwwCSoaGlossary#nid34R0

XML.com. (2007). Retrieved September 25, 2007, from http://www.xml.com/pub/a/ws/2003/09/30/soa.html

Zhang, D. (2004). Web services composition for process management in e-business. *Journal of Computer Information Systems, 45*(2), 83-91.

Zhao, L., & Siau, K. (2007). Information mediation using metamodels: An approach using XML and common warehouse metamodel. *Journal of Database Management, 18*(3), 69-82.

John Erickson is an assistant professor in the College of Business Administration at the University of Nebraska at Omaha. His research interests include UML, software complexity and Systems Analysis and design issues. He has published in journals such as the CACM, JDM, and in conferences such as AMICIS, ICIS WITS, EMMSAD, and CAiSE. He has also co-authored several book chapters.

Keng Siau is the E. J. Faulkner professor of management information systems (MIS) at the University of Nebraska, Lincoln (UNL). He is the director of the UNL-IBM Global Innovation Hub, editor-in-chief of the Journal of Database Management, *and co-editor-in-chief of the* Advances in Database Research series. *He received his PhD degree from the University of British Columbia (UBC), where he majored in management information systems and minored in cognitive psychology. Dr. Siau has over 200 academic publications. His research has been funded by NSF, IBM, and other IT organizations. Professor Siau has received numerous research, teaching, and service awards. His latest award is the International Federation for Information Processing (IFIP) Outstanding Service Award in 2006.*