Oriental Journal of Computer Science & Technology Vol. **3**(1), 37-43 (2010)

Issues of service oriented architecture - A survey

HIMANSHU AGARWAL, ASHISH SETH and ASHIM RAJ SINGLA

Punjabi University, Patiala (India).
Ideal Institute of Technology, Ghaziabad (India).
Indian Institute of Foreign Trade, Delhi (India).

(Received: May 05, 2010; Accepted: June 20, 2010)

ABSTRACT

SOA is a Buzzword today and much is said about it, the actual goal of SOA is to help align IT capabilities with business goals .Another important goal of SOA is to provide an agile technical infrastructure that can be quickly and easily reconfigured as business requirement change. Until the emergence of SOA based IT systems, business and government organizations were faced off with difficult trade off between the expanses of custom solution and the convenience of packaged applications. In this paper we have argued that how service based information systems are different from component based systems. Further we have identified the line of division between two approaches and pointed out the issues of SOA adoption in an organization.

Keywords: Services, components, modelling, security, CBSE.

INTRODUCTION

In today's competitive scenario where business demand changes very frequently, the expectation from technology is raised to level where we are expecting the business processes are developed in such a manner that they can adapt the frequent changes without affecting the overall organization business architecture. Thus the need to assume business processes as a smart services that can be loosely coupled and form the basis of target architecture. Thus need of service oriented architecture arises.

A Service Oriented Architecture (SOA) is a style of design that guides an organization during all aspect of creating and using business services (including conception, modelling, design, development, deployment, management, versioning, and retirement). Though SOA gives you the ability to more easily integrate IT systems, provide multi channel access to your systems, to automate business process, and is the need for current business; it is still in nascent stage and is facing number of issues that must be properly taken care of in order to adopt it completely.

SOA provides a design framework for realizing rapid and low-cost system development and improving total system quality. SOA uses the Web services standards and technologies and is rapidly becoming a standard approach for enterprise information systems. SOA is an architectural software concept whose core working is based on services, a functionality that can perform any specific task and facilitates to support business requirements. In an SOA environment, resources are made available to other participants within the network as independent services that are accessible across the network in a standardized way.

Overall, a business centric, SOA approach delivers a number of benefits, which includes the following: reduced time to market, improved business alignment for growth, reduced costs, reduced business risk. Each Service Oriented Architecture plays one or more of three roles:

- It is a web service responsible for deciding the type of information exposed. A service provider has to make trade-offs between availability & security.
- Service broker or service register is responsible for making information available

to a requestor. A service broker has to decide the amount of information transfer.

- The service requestor or Web service client requests for a service and binds to the service provider in order to call upon one of its Web services.

Component based software engineering (CBSE) Vs Service Oriented Architecture (SOA)

Component based software development approach is based on the idea to develop applications using components, components is analogous to function or procedure in a procedure oriented development, component is more abstract form and is capable to perform specific functionality. Component based software systems are developed by selecting appropriate off-the-shelf components and then to assemble them with a well-defined software architecture.

CBSE is a process that aims to design and construct software systems using reusable software components. CBSE emerged from the failure of object-oriented development to support reuse effectively. Components may constructed with the explicit goal to allow them to be generalized and reused

CBSE is about creating a software package in such a manner as to be able to easily reuse its constituent components in other similar or dissimilar applications. It includes writing high level code that glues together pieces of pre-built functionalities or software building blocks called components. Component is one of the parts of the system that make up a system. It may be hardware, software or firmware and may be sub divided into other components [1].

A component is a software object, meant to interact with other components, encapsulating certain functionality or a set of functionalities. A component has a clearly defined interface and conforms to a prescribed behavior common to all components within an architecture [2]. Component reusability should strive to reflect stable domain abstractions, hide state representations, independent (low coupling), and propagate exceptions via the component interface.

Component-based software development approach is based on the idea to develop software systems by selecting appropriate off-the-shelf components and then to assemble them with a well-defined software architecture. The term component-based software development (CBD) can be referred to as the process for building a system using components [3]. This approach is based on the idea that software systems can be developed by selecting appropriate off-the-shelf components and then assembling them with well-defined software architecture [4, 5]. This new software development approach is very different from the traditional approach in which software systems can only be implemented from scratch. These commercial off-the shelf (COTS) components can be developed by different developers using different languages and different platforms. This can be shown in Fig. 1, where COTS components can be checked out from a component repository, and assembled into a target software system.

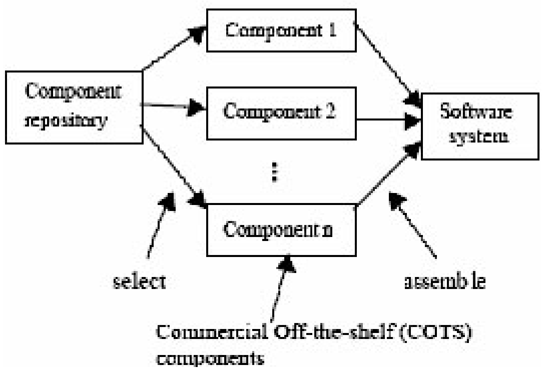
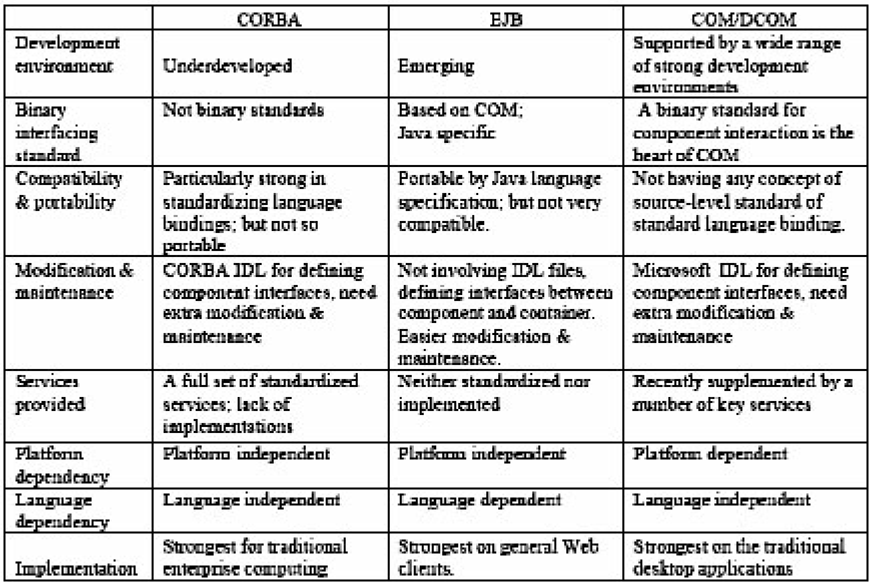


Fig. 1: Component-Based Software Development

The following table (fig 2) provides the comparison among current component technologies and can be found in [4]-[10]. Here is simply a summarization of their different features.

The basic SOA is not architecture only about services; it is a relationship of three kinds of participants: the service provider, the service discovery agency, and the service requestor (client)(see figure 3)



	CORBA	EJB	COM/DCOM
Development environment	Underdeveloped	Emerging	Supported by a wide range of strong development environments
Binary interfacing standard	Not binary standards	Based on COM; Java specific	A binary standard for component interaction is the heart of COM
Compatibility & portability	Particularly strong in standardizing language bindings; but not so portable	Portable by Java language specification; but not very compatible.	Not having any concept of source-level standard of standard language binding.
Modification & maintenance	CORBA IDL for defining component interfaces, need extra modification & maintenance	Not involving IDL files, defining interfaces between component and container. Easier modification & maintenance.	Microsoft IDL for defining component interfaces, need extra modification & maintenance
Services provided	A full set of standardized services; lack of implementations	Neither standardized nor implemented	Recently supplemented by a number of key services
Platform dependency	Platform independent	Platform independent	Platform dependent
Language dependency	Language independent	Language dependent	Language independent
Implementation	Strongest for traditional enterprise computing	Strongest on general Web clients.	Strongest on the traditional desktop applications

Fig. 2: Comparison of current component technologies

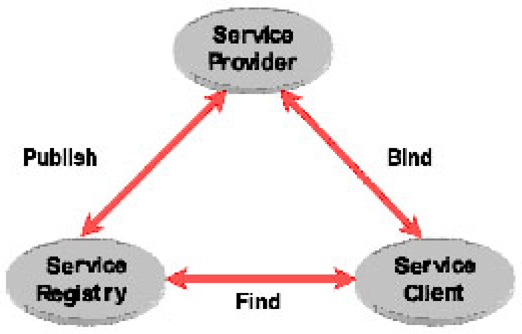


Fig 3. Basic SOA Architecture

Component based architectures and Service-oriented architectures seem to have the same goal: To provide a foundation for loosely joined and highly interoperable software architecture, enabling efficient, error-free software development. Nearly all evolution in recent years had this intention: To develop a type of architecture, that allows loose coupling and high reusability of its components. These attributes should allow more efficient, faster, error-free software production. In more abstract terms, one evolutionary step enhanced the previous step and helped to get closer to these objectives.

There is no clear dividing line between Service Oriented Architecture and Component Based Architecture. In principle SOA is the enhancement of Components: The individual services are single components, which can be linked to gain new business logic, new services or a new component. But the difference between SOA und components seems to consist of two major points:

- Services have to be publicly accessible. Models for consumption will probably be developed though not necessarily cost free. But through registries (UDDI) it should be possible to find services like other business partners in the yellow pages.
- Services have to be largely independent from implementation specific attributes. For users and customers it is irrelevant, if the service is released with Java, .NET or Perl. The shared communication is XML based, and as long as no other protocol exits, the protocol will probably be SOAP.

SOA Modelling Issues

Major modelling activities will concentrate on service discovery, service composition, (as described in basic SOA model)and service granularity, defining Service Level Agreements (SLAs).Zimmermann et al. (2004) specify quality attributes for SOA that cover reusable (well-crafted services), loosely coupled, cohesive abstractions, stateless, meaningful to business and standardized to comply with enterprise architecture patterns and underlying technologies..

According to Ravichandran et al. (2007) IT architectural design features for SOA must include reusable components, modular, autonomous, i.e. capable of interaction and adaptability without human intervention, interoperable, and re-configured flexibly in run time through service matching and dynamic binding.

Ren & Lyytinen (2008) classify three design features for service-based information systems as **system** design features**, service** design features and **business** design features**.** system design features deals with issues like platform-independence, loose coupling, re-usability and interoperability, service design features handles issues like encapsulation, autonomy, deceivability and designed for contracting and business design features takes care of issues related to business semantics, comply with business process and suitable for enterprise integration . They further distinguish between reusability, agility and scalability as quality attributes for IT architectures.

Lin et al. (2009) emphasize workflow monitoring and management, provenance management and data quality management as core building blocks for SOAs. Provenance module will cover Querying, Exception handling, RDF-to-Relational data mapping, OWL (Web Ontology Language)-to-Relational Schema mapping and Relational Provenance repository. Whereas the data quality module will cover XML-to-Relational data mapping and the workflow management module will cover workflow scheduling, removing redundancy, orchestration and breakdown into discrete, autonomous task activities.

According to Mike Rogers As a result of SOA, businesses can achieve better alignment through process optimisation and differentiated capability. Yet, in order to realise this compelling new promise of high performance, organisations will need to examine their current IT strategy with a perspective that reflects the new realities of SOA

SOA Development Issues

According to Zimmermann et al. (2004) service-oriented information systems analysis and design (they refer to as SOAD) have roots in three major existing disciplines; *Object-Oriented Analysis and Design (OOAD)*, *Business Process Modeling (BPM)* techniques, and *Enterprise Architecture (EA) frameworks*,. They suggest a hybrid approach that collates suitable elements from OOAD, EA, and BPM to come up with a three layers SOAD approach to include component, software service & business service layers. In the **business service layer** the approach suggests the use of *BPM techniques*, such as workflow diagrams, as well as *UML Sequence and Interaction diagrams* to model the interaction between the different components across the *enterprise service bus*. In the **software service layer** the approach suggests the encapsulation and *granularity of services*. In this regard, integration of existing legacy applications can be decomposed into stateless services, where *reusable business processes* and rules are abstracted into autonomous services managed by a *business choreography model* represented by *BPEL specifications*.

Phased approach is suggested by Bell (2008) for service-oriented modeling. He suggested three phases for SOA environment which includes service abstraction, service analysis and design activities. In the **abstraction phase** *service discovery* and *conceptualization* (high-level abstractions of business logic and re-usable processes) will be carried out, in the **analysis phase** *service descriptions* will be carried out along with *business integration*, *enterprise architecture* and *meta-data specifications*, and in the **design phase** *component* and *architecture logical and physical designs* will be outlined.

Bitzer & Schumann (2009) emphasize appropriate interactions between Functional and IT departments in order to overcome the Business/IT gap in modeling service-based information systems.. They suggest development process starts with a **business analysis** and *service conceptualizations* by the Functional department. Then both departments collaborate in designing the **SOA** by producing the corresponding *BPM models* and *BPEL specifications* supported by BPEL editors. This is followed by *orchestration* of the different services which will be carried out by the IT department in order to form the *service choreography* supported by BPEL editors. Finally execution and governance of services within the required SOA will be undertaken by the IT department.

Niemann et al. (2008) suggest a generic governance model for SOA to govern development, provisioning and operation of service-based information systems. In their model they specify four phases; planning, design realization and operation. The **planning** phase will cover SOA preliminary specifications along with organizational governance issues such as staffing, competences, streamlining cross department processes, migration of legacy systems & processes, enterprise-wide consolidation, as well as policy and metrics planning. The **design** phase will address detailed business and technical requirements, SOA topology and detailed service specifications. The **realization** phase will target implementation issues, such as realizing the service registry and semantics, SLA implementations, continuous service tests and reviews. and lastly the **operation** phase will cover the major governance activities that will include business service registry management, business service evolution management, architecture evolution and management, SLA management, etc

Issues in Understanding

Businesses can achieve better alignment through process optimization and differentiated capability. Whenever any investment is made on new infrastructure and new technology it is always very important to understand its scope and relevancy that how it suits to your business needs. Not only this but it is must to understand the right way to do it and what you can and can't do. SOA and web services are powerful, but it doesn't mean that they can do everything As identified by Grace A.Lewis et al [16] following are the misconceptions

- SOA provides the complete architecture for a system
- Legacy systems can be easily integrated into an SOA environment
- SOA is all about standards and standards are all that is needed
- SOA is all about technology
- The use of standards guarantees inter operability among services in an SOA environment
- It is easy to develop applications based on services
- It is easy to develop services anybody can use
- It is easy to compose services dynamically at runtime
- Services can only be business services
- SOA can be implemented quickly
- Testing applications that use services is no different than testing any other application

Understanding the SOA in respect to above points is very important, it requires a fresh approach, clear vision and a multi dimensional view to understand the SOA scope.

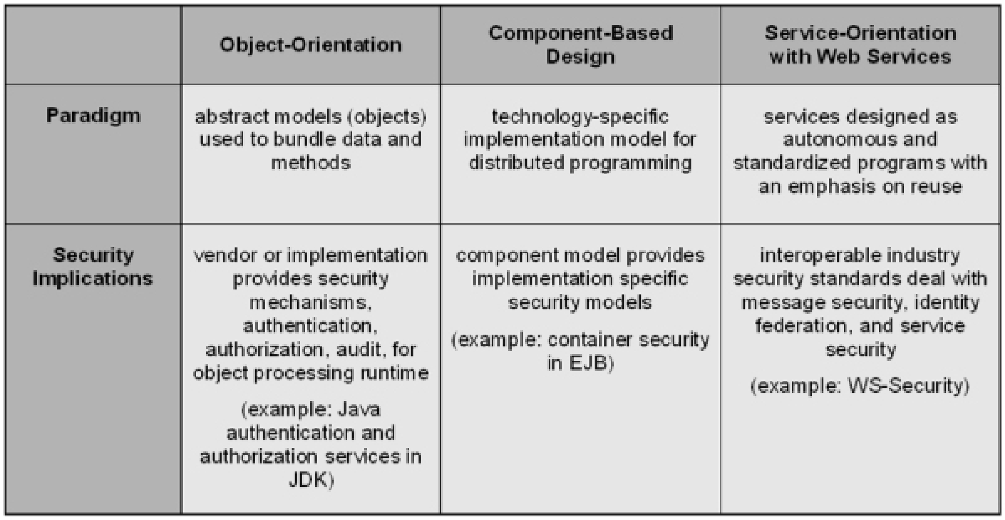
Security Issues

Security is a not a goal in and of itself. It is a business enabler. The great Robert Garigue [11] said that security is like brakes on a car. Because we have brakes we can drive faster. SOA security architects, this is your mantra. Find ways that you can deliver security services to your organization while enabling your business to grow.

Each progression in distributed computing - from object-orientation to component-based design and now to service-orientation - has introduced unique security considerations. Objects and components use similar binary runtimes, but when building services as Web services, we can no longer rely on binary controls for security (fig 4).

The primary security functions required by most systems are: authentication, authorization auditing, and assurance A basic SOA architecture contains components service requestor, service provider and service repository., the communication exists between service requestor to provider in the form of message passing and is the scope for intruder to enter, therefore it is within the message exchange where the authentication, authorization, audit, and assurance services add true value.

Gunnar Peterson shows how security services are needed to mediate communication



	Object-Orientation	Component-Based Design	Service-Orientation with Web Services
Paradigm	abstract models (objects) used to bundle data and methods	technology-specific implementation model for distributed programming	services designed as autonomous and standardized programs with an emphasis on reuse
Security Implications	vendor or implementation provides security mechanisms, authentication, authorization, audit, for object processing runtime (example: Java authentication and authorization services in JDK)	component model provides implementation specific security models (example: container security in EJB)	interoperable industry security standards deal with message security, identity federation, and service security (example: WS-Security)

Fig 4: Different distributed programming paradigms introduce different security considerations.

between a subject and its objects - or, in the SOA world, between the service provider and its requesters (or consumers).

Gunnar Peterson has also suggested following points that might be helpful to incorporate security into SOA:

1. Plan out a security architecture that represent the system from an end to end perspective, and focuses on your assets
2. For each service requester and service provider - and anything in the middle like proxies - understand the current state of access control by analyzing authentication, authorization, and auditing
3. Determine what policy enforcement and policy decision points exist today and which can be strengthened in the future

Issues in Implication for education

As SOA is at its nascent stage, the expertization and clear understanding of the concept is not found among the peoples. Thus it is one of the major academic challenges facing departments is to train existing faculty to teach the various technologies that support the SOA infrastructure. As ERP is already understood and successfully implemented in industries. Many universities have tie-up with industry and major vendors such as SAP / Oracle to standardize their curriculum on ERP backbone .The universities try to cover the conceptual part of ERP within its traditional teaching duration and practical exposure will be given at collaborating industry partner like one university may choose to instruct Business Process Modeling using the ARIS Toolset while another may choose to illustrate the technology using the Net Beans IDE's Business Process Execution Language (BPEL) support.

Also as new technology is emerging day by day it is really difficult to cover all in any particular course curriculum. As it is found that engineering and professional graduates has already wide contents in their course duration. Adding new subjects into their syllabus left the question as to which subject is eliminated form the course in order to balance the student load

CONCLUSIONS

To create an IT environment that maximises the benefits of an SOA approach, certain issues discussed above must be taken care of. It is necessary to be aware of your business and understand well before trying to automate them. SOA can align IT with business, but that simply boost up problems. SOA should consider hardware availability as well as software. If SOA grows to a good extent, then there should be a separate SOA governance system. Further more exploration in the identified issues and finding suitable solutions for them will be the future scope of work.

REFERENCES

1. Sajan Mathew, “Software Engineering”, Edition 2nd S.Chand.
2. CBSE Network, “Component based software engineering workshop”,Budapest April 3-4
3. www.scitation.aip.org/getabs.
4. Xia Cai, Michael R. Lyu, Kam-Fai Wong Roy Ko ,”Component-Based Software Engineering Technologies Development Frameworks and Quality Assurance Schemes”, The Chinese University of Hong Kong Hong Kong Productivity Council.
5. G. Pour, “Component-Based Software Development Approach: New Opportunities and Challenges”, Proceedings Technology of Object- Oriented Languages, TOOLS 26.,pp. 375-383,1998.
6. A.W.Brown, K.C.Wallnau, “The Current State of CBSE”, IEEE Software,Vol. **15** 5, pp. 37-46., Sept.-Oct. 1998.
7. G. Pour, “Enterprise JavaBeans, JavaBeans & XML Expanding the Possibilities for Web-Based Enterprise Application Development”, Proceedings Technology of Object-Oriented Languages and Systems, TOOLS 31, pp.282-291. 1999.
8. G.Pour, M. Griss, J. Favaro, “Making the Transition to Component-Based Enterprise Software Development: Overcoming the Obstacles –Patterns for Success”, Proceedings of Technology of Object-Oriented Languages and systems, pp.419 – 419, 1999.
9. G. Pour, “Software Component Technologies JavaBeans and ActiveX”, Proceedings of Technology of Object-Oriented Languages and systems, pp. 398 – 398, 1999.
10. C. Szyperski, “Component Software: Beyond Object- Oriented Programming”, Addison-Wesley, New York, 1998.
11. “Thinking about Robert Garigue”, 1 Raindrop Blog,
12. Security in SOA - It’s the Car, Not the Garage, by Gunnar Peterson Published: February 9, 2008 (SOA Magazine Issue XV: February 2008, Copyright © 2008)
13. Douglas W. Frye Enterprise Integration, Inc., Alexandria, Virginia, USA, and Thomas R. Gulledge Enterprise Integration, Inc., Alexandria, Virginia, USA and George Mason University, Fairfax, Virginia, USA Industrial Management & Data Systems Vol. 107 No. 6, 2007 pp. 749-761 q Emerald Group Publishing Limited 0263-5577
14. Creating and maintaining coherency in loosely coupled systems
15. Written by Lori MacVittie | Technical Marketing Manager, Application Services
16. Grace A. Lewis, Edwin Morris, Soumya Simanta, Lutz Wrage, “Common Misconceptions about Service-Oriented Architecture” Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS’07), IEEE, 2007