The Study of SOA Architecture

Ms. Megha Shinde[#1] Mr. Dharam Sherathia[#2] Ms. Devika Galia[#3] Mr. Abhijeet Pasi*[4]
[#]Student, Department of Information Technology
*Lecturer, Department of Computer Technology
Shah and Anchor Kutchhi Polytechnic, Mumbai, India
meghashinde135@gmail.com[#1]*,dharam0409@gmail.com*[#2]*, sweetyde10@gmail.com*[#3] *,abhijeetpasi2008@gmail.com* [*4]

Abstract: The early services used to develop a software created great deals of complexity in building new applications, integrating existing modules and at the same time keeping up the maintenance. This paper presents how the rise of Service Oriented Architecture (SOA) handles the complexity faced by Information Technology (IT). The today`s business organizations related to IT demands for something new, ease of handling, better understanding and responsiveness .The main aim in various business organizations is to achieve all these things in a systems in terms of justifying their projects with a return on the investments done on it. To achieve the changing needs in the business environment i.e. adding some new services to the organization bio-data is by changing the business process quickly as per demand. To make a system that includes all the effective features required for uplifting the businesses market value and fulfilling end users requirements and yet being cost effective is to use Service Oriented Architecture (SOA). Web Services plan an important part in implementing Service Oriented Architecture (SOA).

*****

I. INTRODUCTION

Service Oriented Architecture (SOA) is viewed as major concept of modern information technology. SOA is an architectural style or a design approach or organizing, utilizing and developing distributed system for building business applications as a set of loosely coupled interacting software components. SOA is not appropriate for all types of IT applications. For example, it is appropriate for real-time applications. While using SOA paradigm, business applications are looked as set of black box components in order to uplift the level of abstraction and thus, ease the reuse of components in various systems. Business goals such as easy and flexible integration, reduced costs, attractive and innovative services to customers and reaction to opportunities are satisfied by using SOA approach. SOA represents many ways as an evolution of client server architecture. SOA can be implemented using various web services.

II. ARCHITECTURAL CONSTRAINTS

There are no official standard SOA constraints, the only constraints you get are the ones that each vendor or supplier of SOA platforms and systems decides to give you. Naturally, each different solution in this space will differ in those constraints.

Since there are no standard constraints, it really depends on the design of solution/implementation and the requirements to be fulfilled.

Important pointers while implementing SOA without constraints are -

Error handling in case there is a thread exhaustion at the service provider end, the consumer should read time out and should not keep waiting for the response from the provider.

Proper Management of resources and garbage collection at the provider end extrapolating the expected load and designing the infrastructure according to the expected load.

Again for eliminating Constraints for messages & interfaces, keeping in mind the packet size of these messages, the form of the messages should not be too heavy and easily consumable and parseable. Caching is a very important aspect to be noted for these points since it can eliminate constraints for unique request handling.

1.1. Interfaces

The interface constitutes a contract defining the functionality of the service in a platform-independent manner. This implies that the invocation mechanism (protocols, descriptions, and discovery) must comply with widely accepted standards enabling a client to use the service from anywhere applying any OS or programming language. The service interface description publishes the service signature, e.g. its input, output, and error messages. The (expected) behavior is described by the behavior description and the QoS (Quality of Service) describes both functional and non-functional service quality attributes, e.g. performance, security attributes, reliability, etc. Services exhibit several other properties. They are stateless, this means that users can use them without knowing the current conditions of the service; the service maintains its own state. [7]

1.2. Messages

Message passing is a form of communication for inter-module interaction. Processes communicate with each other by sending and receiving messages, where each sent mechanism must match the corresponding receive mechanism. Services communicate with each other and with consumers using messages. The service interface defines the messages a service can process. To achieve

Platform-and language-independency, messages are typically constructed using XML documents that comply with the corresponding XML Schemas. In contrast to Remote Procedure Call (RPC) the mechanism is an asynchronous communication, directly supported by message passing. A schema limits the vocabulary and structure of messages. An extensible schema allows new versions of services to be introduced without modifying existing services. [7]

III. BASIC SOA

SOA constitutes a concept to provide services to clients through published interfaces and to coordinate interaction through the exchange of messages. Generally, the basic SOA describes the
Relationship between three kinds of participants: the service providers, the registry, and the service requestors. The service represents a logical separation of declaration and implementation, its implementation is hidden from the client and can be subject to changes which may not influence the client so long as the service interface stays unchanged. [7]

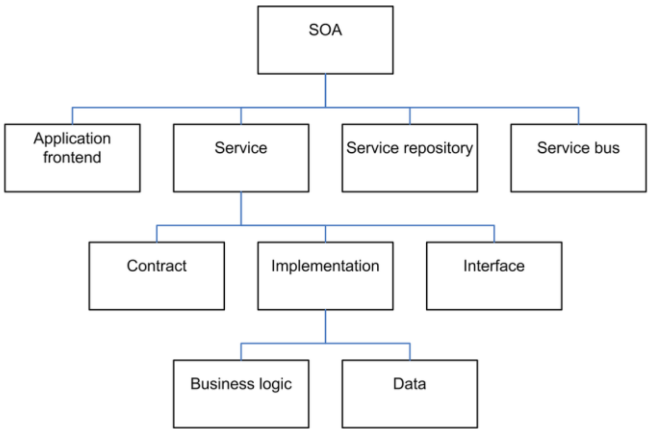

Figure 1. Basic SOA Architecture[12]

1.3. Advantages & Disadvantages of Basic SOA

Advantages include:-

- Improved Information Flow.
- Lower software development and management costs.
- Performance measurement.
- Security attack detection.
- Data confidentiality and integrity.
- Ability to develop new function combinations rapidly.
- Ability to optimize performance, functionality, and cost.
- Easier introduction of system upgrades.
- Ability to integrate existing assets.
- Improved reliability.
- Ability to scale operations to meet different demand levels.

Disadvantages include:-
SOA would not be suitable for applications with GUI functionalities. Those applications would become more complex if they use SOA which requires heavy data exchange. Also application requiring asynchronous communication can't make use of SOA. Also in case of standalone and short lived applications' implementations, SOA will become an added burden.

IV. EXTENDED SOA

The higher layer of the SOA pyramid provides support for service composition and management, and service orchestration, transaction, and security. In the composition layer several atomic services can be consolidated into one composite service.
Depending on their requirements clients apply atomic or composite services as applications and/or solutions. Service aggregators may utilize such composite services as components in further service compositions thus becoming service providers by publishing the service description they create.
A composer of several services must encompass functionalities such as:

- Coordination: establish and manage the control of data flow among the services.
- Monitoring: subscribe to events generated by component services.
- Conformance: ensure integrity of composite service by controlling conformance of component services.
- Quality of Service (QoS): bundle QoS of component services to derive the composite QoS,

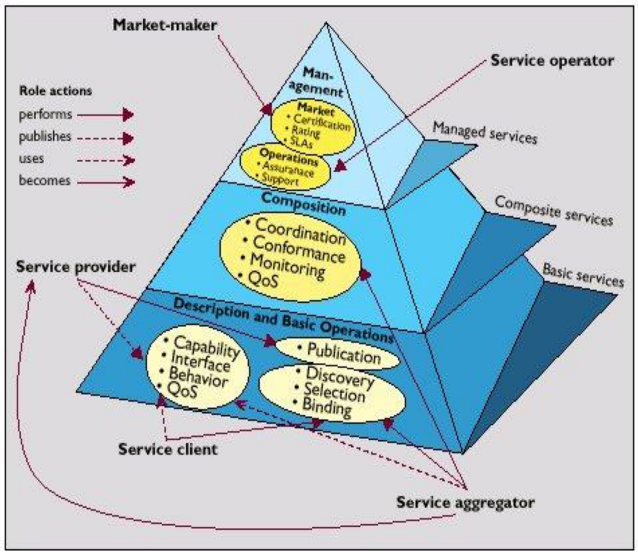

Figure 2. Extended SOA Architecture

1.4. Advantages & Disadvantages of Extended SOA

Advantages include [10]

- Service Reusability
  In SOA, an application is built by assembling small, self-contained, and loosely coupled pieces of functionality. Therefore, the services can be reused in multiple applications independent of their interactions with other services.
- Easy Maintainability

Since a service is an independent entity, it can be easily updated or maintained without having to worry about other services. Large, complex applications can thus be managed easily.

- Greater Reliability

SOA-based applications are more reliable since small, independent services are easier to test and debug as compared to massive chunks of code.

- Location Independence
  The services are usually published to a directory where consumers can look them up. This approach allows a service

to change its location at any time. However, the consumers are always able to locate their requested service through the directory look up.

- Improved Scalability and Availability

Multiple instances of a single service can run on different servers at the same time. This increases scalability and availability of the service.

- Improved Software Quality

Since services can be reused, there is no scope for redundant functionality. This helps reduce errors due to inconsistent data, and thereby improves the quality of code.

- Platform Independent

SOA facilitates the development of a complex product by integrating different products from different vendors independent of the platform and technology.

- Increase productivity

Developers can reuse existing legacy applications and build additional functionality without having to develop the entire thing from scratch. This increases the developers' productivity, and at the same time, substantially reduces the cost of developing an application.

Disadvantages include[10]

- Increased Overload

Every time a service interacts with another service, complete validation of every input parameter takes place. This increases the response time and machine load, and thereby reduces the overall performance.

- Complex Service Management

The service needs to ensure that messages have been delivered in a timely manner. But as services keep exchanging messages to perform tasks, the number of these messages can go into millions even for a single application. This poses a big challenge to manage such a huge population of services.

- High Investment Cost

Implementation of SOA requires a large upfront investment by means of technology, development, and human resource.

Not Recommended For:-

- Homogenous

Implementing SOA for applications that use the technologies of a single vendor will not be cost-effective. For example, if an application is built in Java, then it would be better to use methods of Java rather than using HTTP for inter-component communications.

- GUI – Based

SOA would not be suitable for applications with GUI functionality, e.g. a map manipulation application. Such applications require heavy data exchange, which in turn would increase the complexity of the application if SOA is used.

- Real –Time

SOA is not desirable to be used with strictly-enforced response times since the services communicate asynchronously.

- Stand Alone

It would be futile to invest in SOA for stand-alone non-distributed applications, which do not require request and response-based calls.

V. CHALLENGES IN ENTERPRISE SYSTEM

Following are the changes that the SOA tries to overcome [3]

- Isolating business logic
- Interoperability
- Software silos
- Redundancies

1.5. Isolating Business Logic

Large organization with large business needs and business setup's, have business application that have business logic and are much more tough or complex to build when compared to normal or general application. The Business logic can be set or defined by a non-IT person from the organization. They change the logic according to their own will without understanding the efforts put behind to develop the old business logic and the complexity a small change can create in an application. We can consider a organization's purchase system where a business rule can be set where the purchases or service cost exceeds a threshold value pre-determined has to go through approval of higher authority in the organization . The main problem in programming is to maintain the business logic while maintaining the computer logic, which at times can be a very difficult or tedious task. To avoid unnecessary changes the management need to be educated about the problems to isolate the business logic.

1.6. Interoperability

Interoperability is a requirement of an enterprise system; it is also a challenge at the same time. In a service-oriented architecture, "interoperability" refers to the ability of the service to be invoked by any potential client of the service There are several attributes of a SOA that make this possible: Even if it is, in the context of the business world, the company might acquire another company that uses a totally different IT system altogether. The result is that additional work has to be done to allow interoperability. This can yet introduce another problem associated with the reluctance to migrate or upgrade existing system. Having invested effort and resources to allow interoperability, migrating to a new system might pose challenges if it is not compatible with existing system. From a semantic point of view, service-oriented architectures by themselves do not offer any guarantees. Semantic interoperability depends on how the interfaces to a service are described and how the meaning of the information is shared with potential clients of the service. Several issues that need to be addressed include

- How to know exactly what a service actually offers
- The quality of the service it offers.

1.7. Software Silos

In the physical world, a silo is a robust structure meant to hold and contain things to prevent what is outside from getting in and what is inside from getting out. The problem is that many applications and IT systems end up becoming such silos. The term usually refers to systems that cannot communicate with other related third party systems. Information flow is usually vertical. As the business expands or business requirements change, the end result is that there are a cluster of siloed systems that cannot cooperate or work with one another.

1.8. Redundancies

A problem that is common to many large companies is that there are many similar yet slightly different applications that are used throughout the organization. Each department usually comes out with its own version of software components rather than coordinate with other departments to see if component reuse is possible[u]. The latter turns out to take too much effort as such it usually involves chores such as going through rounds of inter-departmental meetings to determine the common functionalities amongst the different systems and what or what not to be included in the system. While companies might have policies or guidelines for such scenarios, often when deadlines are tight, or due to budget issues, it is often more convenient to build the application the department needs rather than coordinate across divisions. The problem can surface again when one company acquires another and realize that they too have similar applications with the similar functionality. Another issue with such redundancies is the increased effort and complexity to maintain such applications. Any change in business policy will probably render these applications obsolete. All updates will then have to be propagated through these instances of the application. Again in the context of enterprise systems, where the problem is magnified, this translates to higher cost of IT costs and inefficiencies, something that is not desirable.

VI. WEB SERVICES: WHY TO USE IT?

Web Services are pervasive, simple and platform-neutral. Web services are able to execute different sized business applications. It is a technology that can be used to implement SOAs. It is necessary to note that the SOA is an architectural style or design approach that is independent of any technology platform. As the name indicates, web services offers services over the web. Web services communicate with other web applications for exchanging data. It can convert existing applications into web based applications. The World Wide Web Consortium (W3C) provides a more specific and accurate definition:“A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format(specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.”[7].

The web services are of two types-
1. Static Web Services
2. Dynamic Web Services

1.9. Web Services Architecture

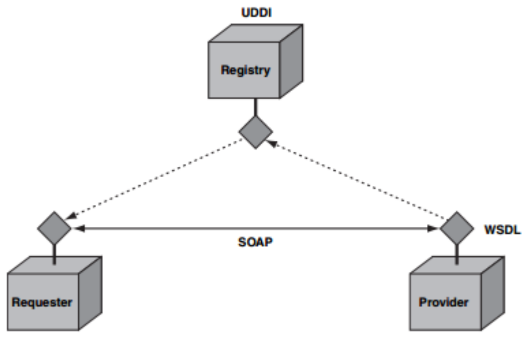

Figure 3. Basic Web Service Architecture [2]

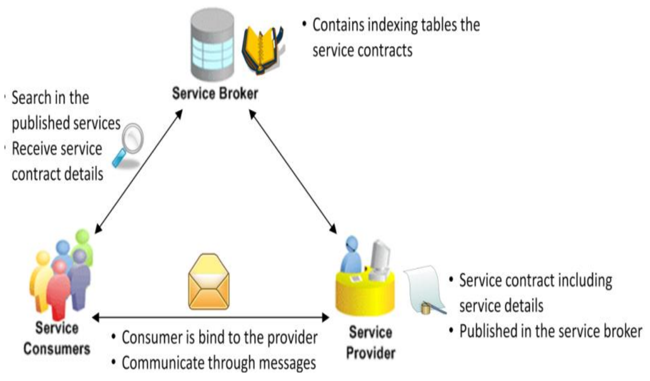
• Contains indexing tables the service contracts
Service Broker
Search in the published services
Receive service contract details
Service Consumers
• Consumer is bind to the provider
• Communicate through messages
Service Provider
• Service contract including service details
• Published in the service broker

Figure 4. Web Service Generic Architecture [4]

- The main components shown in Figure are as follow: Service provider: is the component that implements the web service and informs its existence to other requester by publishing its interface and access information in the service registry.
- Service broker (registry): is responsible for the availability of both interface and implementation access information for the Web service to any service requester.
- Service requester: searches the service within the service broker to find its service provider then connect to the latter using specific communication protocol.[4]

1.10. Web Services Core Standards

The following are the standards used in implementing web services:

- Simple Object Access Protocol [SOAP]

SOAP is the standard format which is based on XML protocol and it is independent of programming languages, used to exchange messages between service users and providers when web services technology is used.[1]

- Web Services Description Language [WSDL]

A WSDL document is written in XML format for describing the implementation of a service as a set of operations on messages containing either document-oriented or procedure-oriented information. It is used by the service provider and the service requester. The WSDL documentation includes the information about the location of the web services and message format.[3] WSDL specifies the operational

characteristics of a Web service using an XML document. It provides a notation to answer the following questions:
-What (is this service about)?
-Where (does it reside)?
-How (can it be invoked)? [6]

- Universal Description Discovery and Integration [UDDI]

UDDI is a platform-independent, XML based registry that allows service providers to list their services and define how service consumers can locate and interact with those services[1].

Table 1: Web Services Core Standards [4]

	SOAP	WSDL	UDDI
Stands for	Simple Object Access Protocol	Web Services Description Language	Universal Description Discovery and Integration
Usage	A format for sending/receiving messages	Describe and locate web services	A directory for storing information about web services
Language	XML	XML	WSDL

1.11. Hypertext Transfer Protocol [HTTP]

HTTP is an application protocol for distributed and collaborative information system. It is protocol to exchange or transfer hypertext. It is the foundation of data communication for the World Wide Web.

1.12. Extensible Markup Language [XML]

XML is a language which is simple and very flexible text format. It is set of rules for encoding documents in a format which is both human-readable and machine-readable. It is designed to store and transport data.

1.13. Advantages & Disadvantages of Web Services

Advantages include:

- Interoperability

This is the most important benefit of Web Services. When different applications are integrated and provided with a service, it may happen that it may not be suitable for each but thanks to the use of standards-based communications methods, Web Services are virtually platform-independent.

- Usability

Web Services allow the business logic of many different systems to be exposed over the Web. This gives your applications the freedom to chose the Web Services that they need. Instead of re-inventing the wheel for each client, you need only include additional application-specific business logic on the client-side. This allows you to develop services and/or client-side code using the languages and tools that you want.

- Reusability

Web Services provide not a component-based model of application development, but the closest thing possible to zero-coding deployment of such services. This makes it easy to reuse Web Service components as appropriate in other services. It also makes it easy to deploy legacy code as a Web Service[11].

- Deployability

Web Services are deployed over standard Internet technologies. This makes it possible to deploy Web Services even over the fire wall to servers running on the Internet on the other side of the globe.

Disadvantages include [11]

- Although the simplicity of Web services is an advantage in some respects, it can also be a hindrance. Web services use plain text protocols. This means that Web service requests are larger than requests encoded with a binary protocol. The extra size is really only an issue over low-speed connections, or over extremely busy connections.
- As Web Services are public over the network for all user, there is huge question of security of the data accessed by individual or business organizations.
- The problem with HTTP and HTTPS when it comes to Web services is that these protocols are "stateless"—the interaction between the server and client is typically brief and when there is no data being exchanged, the server and client have no knowledge of each other. More specifically, if a client makes a request to the server, receives some information, and then immediately crashes due to a power outage, the server never knows that the client is no longer active. The server needs a way to keep track of what a client is doing and also to determine when a client is no longer active.
- Typically, a server sends some kind of session identification to the client when the client first accesses the server. The client then uses this identification when it makes further requests to the server. This enables the server to recall any information it has about the client. A server must usually rely on a timeout mechanism to determine that a client is no longer active. If a server doesn't receive a request from a client after a predetermined amount of time, it assumes that the client is inactive and removes any client information it was keeping. This extra overhead means more work for Web service developers.

VII. BENEFITS OF SOA

- Reuse - The ability to build services that are reusable in many applications.
- Efficiency - The ability to quickly and easily create new services.
- Loose technology coupling - The ability to model services independently over distributed networks.
- Greater flexibility in strategic applications.
- Faster time to value from IT.
- Modernized strategic applications.

- Lower the lifetime cost of applications or infrastructure.
- Reuse as a goal to bring products or capabilities to the market faster[4]

VIII. DRAWBACKS OF SOA

- SOA is only an architecture: It's a set of best practice, not technology. SOA can do what you want to do, not what you can buy.
- The problem of SOA is organization, culture and politics: People are unwilling to accept change and share resources, neither devote for other people's items.
- Core architecture problem of SOA is control, quality and management: It's destined to fail without control.

IX. CONCLUSION

With these study of SOA architecture, we can say that for higher value of business organization in modern technological market, systems will be designed using SOA and deployed to all using Web Services. This will probably help the IT related organizations to come up with innovative ideas to provide services to user with various applications.

REFERENCES

[1] Phil Bianco, Software Engineering Institute; Rick Kotermanski, Summa Technologies; Paulo Merson, Software Engineering Institute. *Evaluating a Service-Oriented Architecture.* Software Architecture Technology Initiative Unlimited distribution subject to the copyright. TECHNICAL REPORT CMU/SEI-2007-TR-015 ESC-TR-2007-015.September 2007. www.sei.cnu.edu/reports/07tr015.pdf

[2] *Chapter 1 Introduction to SOA with Web Services.* www.aw-bc.com/samplechapter/0321180860.pdf

[3] Goh Chun Lin; Koh Eng Tat Desmond; Naing Tayza Htoon; Nguyen Van Thuat; Chapter10 Service Oriented Architecture. A Fresh Graduate's Guide to Software Development Tools and Technologies. Software Development Tools and Technologies. www.comp.nus.edu.sg/~seer/book/2e/ch10.%20Oriented%20Architecture.pdf

[4] Mahmoud Mohamed AbdAllah, Senior R&D Engineer-SECC, mmabdallah@itida.gov.eg; Waseim Hashem Mahjoub, Senior R&D Engineer-SECC. *A Quick Introduction to SOA.* © Copyright Software Engineering Competence Center 2013. www.secc.org.eg/recocape/Documents/SECC_Tutorials_A%20 Quick%20Introductio%20to%20SOA.pdf

[5] Chief Editor: Duane Nickul; Contributors/Editors: Laurel Reitman; James Ward ;Jack Wilber. *Service Oriented Architecture (SOA)and Specialized Messaging Patterns.* Technical White Paper. www.adobe.com/enterprice/pdfs/Services_Oriented_Architecture_from_Adobe.pdf

[6] Mark Endrei; Jenny Ang ;Ali Arsanjani; Sook Chua; Philippe Comte; Pål Krogdahl; Min Luo; Tony Newling. *Patterns: Service Oriented Architecture and Web Services.* International Technical Support Organization. www.redbooks.ibm.com/redbooks/pdfs/sg246303.pdf

[7] *Introduction to Service Oriented Architectures(SOA).* Responsible Institutions: ETHZ(Concept), ETHZ(Overall), ETHZ(Revision). Introduction to Service Oriented Architectures (SOA). http://www.eu-orchestra.org - Version from: 26.10.2007.

[8] Erin Cavanaugh Product Marketing Manager Altova. *Web services: Benefits, challenges, and a unique, visual development solution.* www.altova.com/whitepapers/webservices.pdf

[9] Ying-Hong Wang; Jingo Chenghorng Liao. Department of Computer Science & Information Engineering, Tamkang University Tamshui, Taipei County, Taiwan. *Why Or Why Not Service Oriented Architecture*. 978-0-7695-3729-0/09 $25.00 © 2009 IEEE DOI 10.1109/SSME.2009.126. 2009 IITA International Conference on Services Science, Management and Engineering.tkuir.lib.tku.edu.tw/space/retrieve/77331/Why+or+Why+Not+Services+Oriented+Architecture.pdf

[10] http://www.buzzle.com/articles/advantages-and-disadvantages-of-service-oriented-architecture-soa.html

[11] https://social.msdn.microsoft.com

[12] https://en.wikipedia.org/wiki/Serviceoriented_architecture#/media/File:SOA_Elements.png