Trends in Information Management (TRIM)
ISSN: 0973-4163
10 (1), pp. 40-50

An Analytical Review of Quality Attributes of Service-Oriented Architecture

Parminder Kaur
Hardeep Sing

Abstract

Purpose: *Service-Oriented Architecture (SOA) is becoming an extensive field in research as well as popular architecture pattern because of its support towards quality attributes like performance, scalability, interoperability, reliability etc. Every architecture pattern provides benefits, having positive impacts on quality attributes. On the other hand, each architecture pattern comes with certain liabilities, having negative impacts on quality attributes. This paper makes an effort to illustrate the various factors related to quality attributes of SOA. A trade-off between various quality attributes is also discussed. The existing as well as future efforts to maintain the quality of SOA are reviewed.*

Methodology: *The quality attributes like Interoperability, Performance, Security, Reliability, Availability, Modifiability, Testability, Usability and Scalability are very well explained along with their current status as well as future requirements.*

Future research: *Future work will focus on the analysis of service-level agreements which help in providing necessary level of services to service consumers. Still, a great work is required to deal with the quality attributes and quality requirements in SOA life cycle*

Keywords: *Service-Oriented Architecture (SOA); Interoperability; Performance; Security; Reliability; Availability; Modifiability; Testability; Usability; Scalability*

Paper Type: *Technical*

Introduction

The selection of software architecture is very difficult with respect to the fulfillment of functional as well as quality requirements like performance, scalability, interoperability, availability, security, modifiability, testability, usability and reliability. Now-a-days, for development in distributed environment, first choice is Service-oriented architecture (SOA) due to its quality attributes. It refers to a software design pattern based on discrete pieces of software, which provide services in the form of application functionality to other applications or one can say that it is an architectural style where systems consist of service users as well as service providers **(Bianco, Kotermanski & Merson, 2005).** Each service that makes up an SOA application is designed to perform one activity. It is possible to reuse the code in different ways throughout the application by changing only the way an individual service interoperates with other services that make up the application instead of making code changes to the service itself. SOA is defined in many different ways **(Web Services Glossary, 2004; O'Brien, Bass, & Merson, 2005)** such as:

- "A service-oriented architecture (SOA) is an application framework that takes everyday business applications and breaks them down into individual business functions and processes, called services. An SOA lets you build, deploy and integrate these services independent of applications and the computing platforms on which they run."—IBM Corporation
- "Service-Oriented Architecture is an approach to organizing information technology in which data, logic, and infrastructure resources are accessed by routing messages between network interfaces."—Microsoft
- An SOA is "a set of components which can be invoked, and whose interface descriptions can be published and discovered."—Worldwide Web Consortium

SOA Life Cycle

According to **Seeley (2007)**, the SOA lifecycle starts with gathering the requirements, followed by architecting a solution, developing, testing, deploying and managing it. At the end of life, either modify it or retire it. What differentiates the SOA lifecycle from the traditional application lifecycle is the need for governance to maintain order with loosely coupled services.

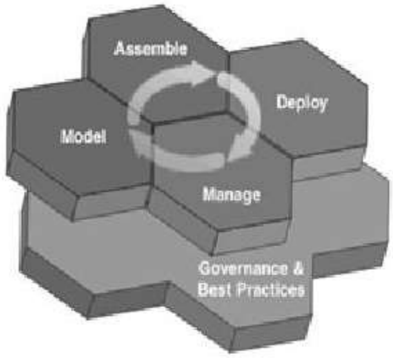

Figure 1: SOA Life Cycle (www.ibm.com)

Model includes the definition of Business Service Level Agreements (BSLAs), IT Service Level Agreements (SLAs), and the associated performance objectives, as well as performance modeling and simulation. *Assemble* contains some component-level performance measurement and monitoring, or at least the enablement for that and the inclusion of heartbeat components. The major part of performance testing would be under *Deploy*. *Manage* can include the management of the composite application and its involved subsystems to achieve performance objectives. *Governance and Processes* includes SLA and BSLA reporting and, for example, capacity management **(Metzger, 2010).** An eight-step outline of the basic SOA lifecycle for an application built with services is provided by Bradley F. Shimmin, principal analyst for application infrastructure at Current Analysis LLC. It differs in some aspects from the steps other analysts, but not radically, so it provides a starting point. Eight steps for building an SOA with services include: *Data collection* including gathering of business requirements and use cases; *Design* including determining service requirements; setting service policies; establishing compliance tasks; building and testing models; and constructing data integration followed by *Development* which includes development of the service and composition of the application from the services; Quality Assurance (*QA) / Test / Acceptance, Deployment, Monitoring/management, Change* and *Retirement.*

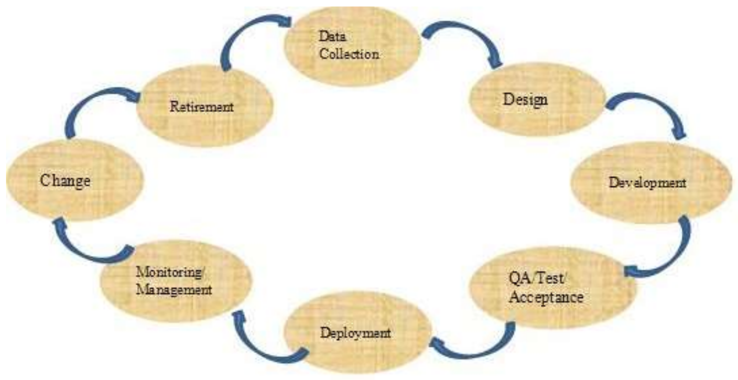

Figure 2: Eight steps for building SOA with services

SOA Quality Attributes

The success of any system depends upon its functional as well as quality attributes. The quality attribute requirements are determined by following factors:

- Interoperability
- Performance
- Security
- Reliability
- Availability
- Modifiability
- Testability
- Usability
- Scalability

Interoperability

It means ability to work of any component or application with other component or application without making any special effort. In other words, the ability of a collection of communicating entities to share specific information and operate on it according to an agreed-upon operational semantics is known as interoperability **Brownsword, et al. (2004).** It is determined by factors like:

- *Number of different platforms* (like Microsoft .NET, Java from SUN Microsystems, Perl, PHP),
- *Complexity of Platforms* (due to non-availability of universal standard for interoperability)
- *Coupling between Platforms* (via Web Service Definition Language (WSDL), Simple Object Access Protocol (SOAP), Web services standards (e.g., Business Process Execution Language [BPEL], WS-Security, WS-ReliableMessaging and ebXML)).

Web Services-Interoperability Organization (WS-I) was chartered in 2002, to promote the interoperability of Web services across platforms, applications, and programming languages. Still more research work with respect to Data Access, Encoding Style and Conversion Format is desired to achieve smooth interoperability between distributed systems.

High Interoperability $\longleftarrow$ {Multiple Platforms, Interface format, Communication Protocols}

Performance

Performance of a SOA can be measured with respect to factors like

- *Response time* (i.e. how long does it take to process a request),
- *Throughput* (i.e. how many requests can be processed per unit of time), or
- *Timeliness* (i.e. desired time to process a request).

Factors affecting the performance of SOA include distributed environment, number of intermediates, number of directory lookups and message format. Communication over the network increases response time, making SOA a poor choice. Intermediates like Simple Object Access Protocol (SOAP) engines, proxies, and Enterprise Service Bus (ESB)'s cause performance overhead. Directory lookups help in reducing response time and increasing throughput. The use of a standard messaging format increases the time desired to process a request. The main parameter which helps in increasing the performance of SOA is Location Transparency of the deployed service. Still there is a need to build performance models of the highly complex run-time environment for SOA-based systems and deriving the performance parameters for these models.

Security

Security of SOA and web services is linked with four factors i.e.

- confidentiality

- authenticity
- integrity
- availability

Web servers that host Web services must be configured to use Secure Sockets Layer (SSL) i.e. a cryptographic protocol which is designed to provide communication security over the Internet. Digital certificates are also to be provided to encrypt data transmission and authenticate the communicating parties. In 2002, a comprehensive security model for Web Services (WS) has been proposed by IBM, Microsoft, and VeriSign. Along with WS-Security policy, Standards like WS-Authorization, WS-Privacy, WS-Trust, WS-Federation, WS-Security Policy, and WS-Secure Conversation came into existence. WS-Security defines a standard set of SOAP extensions that can be used to provide message content integrity and confidentiality. It accommodates a variety of security models and encryption technologies and is extensible to support multiple security token formats [5-6]. Another two proposed standards relevant to Web-services security are Security Assertions Markup Language (SAML) and eXtensible Access Control Markup Language (XACML). SAML provides a standard, XML-based format to exchange security information between different security agents over the Internet. It allows services to exchange authentication, authorization, and attribute information without organizations and their partners having to modify their current security solutions **McGovern, Tyagi, Stevens, & Matthew (2003).** XACML provides a language to specify role-based, access control rules in a declarative format. Security mechanisms, sometimes, may have a negative impact on performance, modifiability and interoperability of web services. These issues can be handled by WS-I, a basic security profile, which ensures interoperability of security features among vendors [8]. Besides all these existing efforts, a detailed research is required to find a suitable security standard for SOA-based system.

Reliability

Reliability defines as the ability of a system to keep operating over time without failure **(Clements, Kazman & Klein, 2001).** From the SOA point of view, the reliability of the messages that are exchanged between service users, service providers and the reliability of the services themselves is of major concern. Reliability can be categorized as

- Message Reliability
- Service Reliability

Message Reliability can be ensured by Microsoft Message Queuing (MSMQ) technology, which enables applications running at different times to communicate across heterogeneous networks and systems that may be temporarily offline **(Microsoft, 2014).** Another technology introduced by IBM, known as WebSphere MQ, can transport any type of data as messages, enabling businesses to build flexible and reusable architectures. It works with a broad range of computing platforms, applications, web services and communication protocols for security-rich message delivery **(IBM, 2014).** OASIS (Organization for the Advancement of Structured Information Standards) consortium introduced a specification in the year 2004 named as WS-Reliability, which is a SOAP-based specification that fulfills reliable messaging requirements critical to some applications of Web Services **(OASIS, 2014).** Service Reliability is associated with Service Accessibility (the ability to initiate a transaction in the service when desired), Service Continuity (the successful continuation of a successfully-initiated transaction to its completion) and Service Release (the successful close of a transaction when it is complete) **(Tortorella, 2013).** A transaction refers to a unit of activity within which multiple updates to resources can be made atomic such that all or none of the updates are made permanent. This reliability can be achieved by Distributed Transaction Model, which supports the implementation of services in different languages as well as platforms. OASIS, the international standards consortium, introduced Web Services Transaction (WS-Transaction) version 1.1, an extensible framework for providing protocols that coordinate the actions of distributed applications, proves a significant step in the evolution of SOA and Web services **(OASIS, 2014).** According to **Sun, Hammer, Biemolt, and Groefsema (2006),** OASIS Business Transaction Protocol (BTP), an eXtensible Markup Language (XML)-based protocol, helps in representing and managing complex, multi-step business-to-business (B2B) transactions over the Internet. Other Web Service Transaction standards include Business Process Execution Language (BPEL) and Web Services Choreography Description Language (WS-CDL), which help in defining business rules, executable business processes and cross-enterprise collaborations of Web Services with respect to their common observable behavior **(Web Services Glossary, 2004).**

Availability

Availability of services is of great concern for the success of SOA. It can be seen from two perspectives: user's as well as provider's perspective. From user's perspective, if the service is not available, then the functional requirements of the system cannot be met. Availability is defined as a proportion of time when a system or a component is accessible for use. It is determined by

- Percentage of performance requirements being fulfilled
- Percentage of available services
- Downtime/uptime for a user and service provider

Service provider provides the service under a Service Level Agreement (SLA). Monitoring the availability of service and fulfillment of the quality-of-service requirements such as performance are taken care by service providers. Techniques like replication and load-balancing are used to increase the availability of services. Services must also have built-in contingencies, so that they can find an alternative provider for themselves in case of unavailability of services. But still there is lot to do with availability of contingency mechanism within a service for its automatic availability.

Modifiability

Clements, Kazman and Klein (2001) state modifiability refers to the ability to make changes in a cost-effective and quick manner within a system. Two factors which affect modifiability are:

- Extensibility
  - How many new services have been added or modified with/without new interfaces?
  - How many old services deleted/discontinued?
- Changeability
  - How many changes have been added to existing services?

SOA is loose-coupled in nature, which results in reduction of cost of modifiability. In SOA, service capability can be extended without affecting other parts of the system. Extension in SOA includes addition of new services according to web standards and extension of existing services with/without changing the interfaces. A great research is required to classify the processes and techniques which deal with identifying the impact of services and incorporating new versions of service within the existing SOA environment.

Testability

It refers to a degree with which a service/system facilitates the establishment of test criteria and the performance of tests to determine whether that criterion has been satisfied **(IEEE Computer Society, 1990).** Testing of SOA is very difficult due to:

- System elements that reside on different machines across the network
- Unavailability of source code of external services to service users,
- Use of different platform or operating system or middleware technology by a service and error in XML document.

It is very difficult to find the problem in runtime environment, therefore issues like how to carry out testing and debugging, is still a matter of concern in an environment where services can be discovered dynamically during runtime.

Usability

When a service interacts with the user, then its level of quality determines usability. It depends upon

- Availability of services
- Complexity of Interface
- Coupling of services
- Reliability of services

Due to distributed nature of SOA, user actions involve calls to remote service providers. If the service call takes a long time to respond then it is good to move the service communication on the separate thread which contains service-users. But in the case of web-services solutions, there is no option to give the user-effective feedback or control over the communication **(Loughran & Smith, 2005).** The SOAP protocol does not have the option for progress notification or cancellation of an active call. But these two options are required when a lengthy process is requested. There is still a need to find the mechanisms which provide user's effective feedback or control over communications.

Scalability

The ability of SOA to work well when the change in size or in volume occur in the system to meet user's need is known as scalability. Web service technology do not consist any inherent scalability features. Platform vendors provide the mechanisms like

- Horizontal scaling (addition in load-balanced servers),
- Vertical scaling (increment in the capacity of the server),
- Stateless services (avoid session management) and
- Service scope (creation of an instance of a service).

Scalability can be determined by sources like type of transport protocol, the XML parser, the load-balancing algorithm and SOAP runtime. The performance of the system depends upon the magnitude of the scalability like how many service users like 10, 100, 1000 or 10,000 are handled by the system. The strategies which can affect the quality of system are needed to develop for making system more scalable.

Trade-off between Quality Attributes

Table 1 shows the trade-off between quality attributes of SOA. A +ve sign shows that if there is an increase in the attribute of a subsequent row, it has positive impact on the quality attribute in the given column. A -ve sign shows that if there is an increase in the attribute of a subsequent row, it has negative impact on the quality attribute in the given column. For example, if service is more interoperable, then its performance will be decreased because it becomes more complex in nature. On the other side, if performance increases, interoperability decreases. Trade-off analysis is essential to achieve an improved system **(Becker, Trifu, & Reussner, 2008; Schropfer et.al, 2009)** because a match is required among the functionality that is required and the other that is provided by SOA.

Table 1: Tradeoff between SOA Quality Attributes

	Interoperability	Performance	Security	Reliability	Availability	Modifiability	Testability	Usability	Scalability
Interoperability	▨	–				+			
Performance	–	▨		–		–	–	–	–
Security			▨	+					
Reliability		–	+	▨	+	+	+	+	+
Availability				+	▨				
Modifiability		–		+		▨	+		+
Testability		–		+	+	+	▨	+	
Usability		–					+	▨	
Scalability		–		+	+	+	+		▨

Conclusion and Future Work

In this paper, the concept of SOA, an architectural style for building systems along with its life cycle is explained. The quality attributes like interoperability, performance, security, reliability, availability, modifiability, testability, usability and scalability are very well explained along with their current status as well as future requirements.

Future work will focus on the analysis of service-level agreements which help in providing necessary level of services to service consumers. Still, a great work is required to deal with the quality attributes and quality requirements in SOA life cycle.

References

Atkinson, B., et al. (2002). *Specification: Web Services Security (WS-Security), Version 1.0.* Retrieved from http://www-128.ibm.com/developerworks/library/ws-secure/

Basic Security Profile 1.1–Working Group Draft (2012). *Web Services-Interoperability Organization (WS-I).* Retrieved from http://www.ws-i.org/profiles/basicsecurityprofile-1.0-2004-05-12.html

Becker,S., Trifu,M., & Reussner,R. (November, 2008), Towards Supporting Evolution of Service-Oriented Architectures through Quality Impact Prediction. *Proceedings of 23rd IEEE/ACM International Conference on Automated Software Engineering(ICASE-08)*. L'Aquila,Germany. DOI: 10.1109/ASEW.2008.4686297

Bianco,R., Kotermanski,R., & Merson,P. (2007). *Evaluating a Service-Oriented Architecture.* Retrieved from http://www.sei.cmu.edu/reports/07tr015.pdf

Brownsword,L.L., et al. (2004). *Current Perspectives on Interoperability*. Retrieved from http://resources.sei.cmu.edu/asset_files/TechnicalReport/2004_005_001_14390.pdf

Clements, P., Kazman, R., & Klein, M. (2001). *Evaluating Software Architectures*. Boston: Addison-Wesley

IBM (2014). *A flexible system for efficient transport of messages and data*. Retrieved from http://www-03.ibm.com/software/products/en/wmq

IEEE Computer Society. (1990). In *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries: 610.* doi: 10.1109/IEEESTD.1991.106963

Loughran,S., & Smith,E. (July, 2005). Rethinking the Java SOAP Stack. *2005 IEEE International Conference on Web Services (ICWS 2005).* Florida, USA. Retrieved from http://www.hpl.hp.com/techreports/2005/HPL-2005-83.pdf

McGovern, J., Tyagi, S., Stevens, M., & Matthew, S. (2003). *Java Web Services Architecture.* San Francisco: Morgan Kaufmann Publishers.

Metzger, R. (2010). *Best practices for BPM and SOA performance*. Retrieved from http://www.ibm.com/developerworks/websphere/library/techarticles/1008_metzger/1008_metzger.html

Microsoft (2014). *Message Queuing*, Retrieved from http://msdn.microsoft.com/en-us/library/windows/desktop/ms711472(v=vs.85).aspx

O'Brien,L., Bass,L., & Merson,P. (2005). *Quality Attributes and Service-oriented Architecture*, Retrieved from http://repository.cmu.edu/cgi/viewcontent.cgi?article=1440&context=sei