Security Trade-off Analysis of Service-oriented Software Architecture

Hassan Reza*, Washington Helps

School of Aerospace Sciences, University of North Dakota, Grand Forks, ND 58202 USA

*Corresponding Author: reza@aero.und.edu

Copyright © 2013 Horizon Research Publishing All rights reserved.

Abstract This paper presents a qualitative analysis of security aspect of Web-Based applications that utilize Service Oriented Architecture (SOA). The architectural solutions that address security requirements are examined and compared with other quality attributes relevant to web-based systems. More specifically, a trade off analysis based on ATAM performed to show the correlation between security and other system wide qualities related to the successful selection of SOA. The optimal architectural solution should not only meet the security requirements of a web-based system but also meets other QAs such as performance, availability, usability, modifiability, etc.

Keywords Software Architecture, Service Oriented Software Architecture, Security, Quality Oriented Software Architecture, ATAM, Architectural Optimizations.

1. Introduction

The success or failure of any nontrivial system depends heavily on its overall structural representation known as Software Architecture [2]. Software architecture of a system has shown to be very effective in managing the complexity of a system and implementing the system wide qualities known as non-functional requirements (NFR). Examples of NFR include security, performance, etc [7], [22].

There is no unanimous definition regarding what constitutes software architecture of a system in the software engineering community. Two widely adopted definitions define software architecture either 1) in terms of components (computational elements), connectors (interactions or protocol of communications), configurations (overall structure), and constraints (rules) [2], or as a set of architectural decisions [19], [20].

Without suitable architectural presentation, it will be very difficult to guarantee that the end product will meet customers need in terms of functionalities (what the system do) and qualities (how the system does it).

The advent of the Internet resulted in the explosion of online businesses. Currently, every company and

organization regardless of its size has a website on the internet. It is now possible that websites to collaborate with each other to accomplish certain tasks (or services) that are very essential to overall customer transactions. For example, an online airline reservation system may need to collaborate with other websites such as payment service provider (e.g., PayPal) to allow customers to conduct shopping transactions.

Service Oriented Architecture (SOA) refers to a software architectural style that can be utilized to implement business services as a set of loosely coupled, autonomous, and distributed components aimed at delivering a well defined level of service [3], [8].

The architectural style of SOA, for most parts, shares a lot of similarities with client/server architectural style [2],[19]. At the heart of SOA is the notion of services (functional units). Solutions to the customer requests are generated by interfacing and integrating dispersed services using XML and SOAP (Simple Object Access Protocol) respectively. SOA begins with a customer submitting a request to a service provider. The service provider, upon receiving the request, processes the request by performing a set of tasks (actions); it then returns the results back to the requester.

The architecture of SOA is slightly different from the traditional Client/Server style. For instance, using SOA, servers can play dual role of clients and servers at the same time. Moreover, SOA provides higher degree of flexibility, reusability, and interoperability [4]. For example, flexibility can be achieved by adding/deleting new services, and reuse, by reusing existing services in different setting.

A typical architecture a SOA consists of of three essential components as well as three fundamental operations, regardless of its implementation [5]. As depicted in the figure 1, the three SOA components interact via three basic operations, namely, publish, find, and bind.

There exists enormous difficulty in the designing of a robust and secure web-based system. Security had been attributed to the failure of web-based applications regardless of what standards and protocols (e.g., SOAP and/or WSDL) have been enforced [6], [8], [9]. The security is considered to be dominant issue in the design of web-based systems such as SOA (or cloud computing [1]),

because sensitive information (e.g., financial/medical information) are on the net and hence can be compromised by hackers.

Figure 1. The SOA model according to IBM

As discussed in [9],[12], SOA presents a new set of challenges because components are loosely coupled and are exposed as independent services on a network. Developers are blamed to fail to pay close attention to the security aspect of SOA services. As such, vulnerabilities issues such as injection flaws (e.g., SQL Injections), issues related to XML, Denial of Service (DoS), insecure communication, and insufficient authentication are magnified in SOA.

With the focus on the security, there are many issues that are imperative to normal ecommerce transactions and must be considered when internal services are exposed to the outside world. Some of these issues include trust, data integrity, and confidentiality.

The initial concern in SOA security is exposed when one examines the traditional methods of providing security with e-commerce applications where each company hard coded authorization and authentication parameters that were guarded behind firewalls where communications take place through VPNs (Virtual Private Networks). This methodology is contrary to what SOA represents (i.e., it is costly, and difficult modify). As such, it is replaced with new open standards such as XML, Simple Object Access Protocol (SOAP), WSDL (Web Services Description Language) and UDDI (Universal Description, Discovery and Integration) which enable the transmission and description of data and procedure calls between systems. SOA demands that the architecture be more flexible and open to access from multiple systems which are critical for ecommerce transactions. As discussed in [8], the problem with this replacement is that none of these standards contain any inherent security aspects of their own.

Other non-functional requirements (NFR) relevant to web-based systems and SOA include performance, usability, modifiability and availability and inter-operability. Performance is a quality attribute with well-established area of its own. Performance refers to the degree of efficiency and effectiveness by which the system responds to external

stimulus [13][19]. Usability refers to the ease by which users can interact with the system in order to complete a task [13][19].

Modifiability is an attribute that describes the ease by which a system can be modified to add, delete, or update a feature system [7] [13] [19]. Availability refers to extent by which the system is up and running [13][19].

Therefore, when an architect considers a security solution for SOA implementation, s/he has to consider a number of mechanisms and alternatives needed to handle various aspects of SOA security [15]. The difficulty then is selecting the optimal (or near optimal) architecture that meets the security requirements without compromising other NFRs (performance, availability, etc.).

In following sections, we elaborate security requirements, and then we discuss our approach to select an optimal SOA that meets security requirement and other NFRs relevant to the successful deployment of SOA.

2. Background: Security Requirements and Solutions

In general, security refers to the ability of a system to withstand attacks and unauthorized accesses. Security is often studied in isolation and late in the development process. If security is perceived to be a major concern in a system, then the issues relevant to security must be fully understood and addressed early on during the architectural design of a system and not after the system is built.

Security must be considered as a part of the system overall design (both high and low levels). Sometimes due to the constraints placed on a project (e.g., budget or deadline), security is not properly dealt with.

On the other hand, the very nature of SOA demands the solution to the problem of the critical issues attributed to the security. As discussed earlier, security is a very broad term and very difficult to specify. As such, it means different concepts to different domains and/or entities.

Security is a collective term and therefore consists of a set of sub-attributes. The sub-attributes associated with security include: 1) confidentiality which is protecting unauthorized disclosure of information using encryption; 2) trust, which includes degree of confidence using authentication and authorization; 3) integrity, which is protecting unauthorized change of information from users; and 4) availability (Avl), which is the assurance that the authorized personnel can access information.

Security is a very broad concept and hence difficult to define. In this work, we attempt to narrow down the scope of security requirement to those issues related to web-based systems. The most significant of these concerns include access control, communication security, and availability (figure 2). As shown in figure 2, access control includes the notion of trust, msg, and availability (Avl), which are related to identity, authentication and authorization, and confidentiality and integrity.

Figure 2. The security requirements for an ecommerce system

An identity is a property of a user or a consumer that provides uniqueness. In a SOA environment, one of the design decisions for security must include identity. Identity must be decoupled (disconnected) from the services. In a SOA, identities exist for both users and services. These identities need to be properly distinguished so that appropriate security controls can be applied [10]. Furthermore, the identities might need to be propagated throughout the SOA environment. Propagation is the case where a user or service may need to access multiple layers of services. In many cases, service implementations can restrict the options and formats available for propagating a user's identity to/from the service. Therefore, identity services are required in the infrastructure to deal with these identity mediation issues, so that services can be easily interconnected without being concerned about how to map and propagate user identity from one service to the next. This approach can greatly reduce the amount of code written and hence improve the speed and ease of developing new services.

In Figure 2, authentication is the process of proving that the consumer legitimately has claimed its identity by evaluating additional information that is bound to his/her identity using proper credentials (e.g., passwords). Authorization is the process of evaluating if an authenticated identity is allowed to have its request fulfilled.

Communication security has to do with the notion of message security (figure 2), which relates confidentiality and integrity. Confidentiality ensures that the data cannot be read by unauthorized users such as is the case if an unsecured message is intercepted by a hacker. While integrity means protecting data against unauthorized modifications.

Availability (Avl in figure 2) of a service requires that a response should be provided by that service in a timely manner. Availability is a non-functional requirement separate from security, but it is also considered as sub-attribute of security; it is important in an ecommerce environment when one considers attacks such as denial of service (DoS).

As discussed in [11], managing identity authentication, authorization is often neglected in SOA. The sheer of complexities involved in managing identity in a distributed

network that requires some mechanism to deal with issue. In information technology, Federated Identity Management (FIM) is one such mechanism. FIM is defined as a set of policies, protocols, and practices to handle identity, and adds trust to the system. It is an industry framework built on top of industry standards. FIM allows subscribers from different organizations/entities use their internal identification data to obtain access to the networks of all enterprises in the group.

For instance, if a user from manufacturer A wants to order supplies from supplier B, then FIM will issue a standard security token or Security Assertion Markup Language (SAML) assertion that describes the user from manufacturer A, his/her role and organization. Based on this information exchange, supplier B can be sure that the user and order are legitimate.

FIM ensures confidentiality and integrity communications and data exchange in that the WS-Security and WS-Trust specifications protect SOAP envelope headers and content as it is exchanged between organizations. In this way FIM cannot be compromised or tampered with by man-in-the-middle attack.

Federated Single-Sign-On (SSO) and provisioning is also supported by FIM where users once logged to their company's system can easily access their accounts or services in other systems without the need for separate user names and passwords.

Identity management software works in tandem with FIM. It provides a single identity for a user that can be used throughout a computer network and that is enforced regardless of what the user attempts to do. It also manages the rights and permissions granted to the user so that the user can do only those operations (e.g., read, or update) for which s/he has been authorized [3]. For instance, if the user requests a business service through the portal, the portal then contacts the service broker, passing it a security token created by the identity management service. The security token contains credentials, including the identity of the user and the details of the access rights of the user. The security token is encrypted so that it can be read only by trusted software.

Identity administration is a costly and complex task; this requires provisioning users with accounts to access services and applications. As such, managing identities across multiple organizations is a significant administrative overhead. For this reason, there is a need to have FIM; it is important to have a proper method to handle identity. FIM is a reasonable solution for managing identity in a SOA system. FIM is flexible, dynamic, and scalable in a sense that its implementation is independent of the services or components that it secures.

Access control in terms of authentication and authorization is handled well, because credentials are checked and appropriate restrictions are enforced [3]. Authentication not only includes mechanisms such as user name and password, but can also support hardware tokens such as Media Access Control (MAC) addresses and biometric data.

A typical authentication process involves collecting the

consumer's identity and authentication credentials and evaluating that the credentials presented by the consumer correspond to credentials that are expected to be presented by the user. Authentication services together with identity propagation described in the previous section provide solution components that enable end-to-end identity flow, one of the most significant security challenges in SOA [10]. A common method of providing authentication for web based applications is SAML (Figure 3).

As shown in Figure 3, SAML is a secure XML-based communication mechanism for exchanging authentication and authorization data between entities (organizations), that is, between an identity provider (a producer of assertions) and a service provider (a consumer of assertions). SAML eliminates the need to have multiple authentication credentials such as passwords in multiple locations. This is important, because such a mechanism increases security by eliminating additional credentials, which in turn reduces the opportunities for identity theft and phishing (by limiting the number of times a user has to enter log in information in forms etc.). In Figure 3, the dashed arrows represent the flow of information.

Figure 3. Basic SAML structure and operation

Authorization follows authentication which means that once a user has been authenticated, authorization can then be performed next. In general authorization has to be flexible and supports various techniques such as role-based or attribute based access control. It also should be independent of the authentication in order to provide modularity [14].

Authorization is enforced at different levels in the overall process. Different components such as an operation or data within the system require differing levels of authorization and hence may enforce these restrictions. For example, there may be rules to access data based on an access list or permissions within a database. A role-based access control (RBAC) model is a common solution for dealing with authorization [24].

Confidentiality is a term used to describe the ability of a system to ensure non-disclosure of sensitive information travelling through un-trusted communication networks (such as over the internet) or at rest, such as in data stores and volatile memory. Confidentiality commonly relies on

cryptographic techniques such as encryption. This usually means encrypting the data so that data cannot be read by unauthorized persons; only people who have the decryption key can read the data.

Secure Sockets Layer (SSL) is a common example of a secure transport level scheme. However, the WS-security specification has been found to be more advantageous in providing message-level security. It can provide end-to-end message level security, which means that the messages are protected even if the message goes through multiple services, or intermediaries [10].

The architect must understand the business process perspective and the technical security concerns to design a meaningful authorization scheme and properly performs tradeoffs analysis. Adding levels of authorization to the architecture tends to negatively affect the same factors as authentication: performance, modifiability, usability, and interoperability [15].

WS-Security is flexible and is designed to be used as the basis for securing Web services within a wide variety of security models including Public Key Infrastructure (PKI), Kerberos, and SSL. It provides support for multiple security token formats, multiple trust domains, multiple signature formats, and multiple encryption technologies to provide integrity or confidentiality. XML encryption is also supported and since preserving the openness of SOA while instituting tight message-level security standards, XML encryption is a good measure to use.

Protection of data from unauthorized modification and disclosure is a key requirement within SOA. A policy must be in place to ensure that data is protected in both transit and at rest, with consistent security measures applied [16]. Data protection is especially important when data moves outside the organizational boundary, which can happen without the knowledge of the consumer. For example, an internal service might require and outsourced external service, with data now flowing to the external organization. If the data is business or privacy sensitive, then the service provider might need to ensure appropriate protection is in place to satisfy the policy requirements of the calling organization.

Protecting against undetected data modifications is usually achieved using digital signatures or message authentication codes. Data integrity techniques can indicate whether information has been altered and such techniques commonly rely on cryptographic mechanisms. Integrity and confidentiality often share the same solutions since both requirements address the issue of message security.

Availability of a service or resource implies that it is able to provide a response to a request in a timely manner. Ensuring that services are available when required is a key design decision in many SOA environments. Architecting for ultra availability requires application clustering, database clusters, etc. Also, if a service provider can build into its applications contingencies such as exception handling when an invoked service is not available (dynamically locating another source for the needed service), availability would not decrease and could actually be improved [15].

Table 1. Mitigation mechanisms used to implement security requirements

Security Requirement	Mechanism
Identity	FIM
Authentication	SAML, FIM
Authorization	DAC, XACML
Integrity	WS-Security, Encryption, Digital signing
Confidentiality	WS-Security, Encryption, Digital signing
Availability	Database and Application clusters, exception handling

2.1. Security and QAs

Quality attribute requirements, such as performance, security, modifiability, reliability, and usability, have a significant influence on the software architecture of any system and also are important in web-based system using SOA [17].

The FIM solution handles identity of user can positively impact usability since users do not have to provide credentials multiple times (single-sign-on). Scalability is positively impacted as well because any number of users can be added or removed or modified easily. Reusability and modifiability are other QAs that are positively impacted by FIM. On the other hand, performance as a QA may be negatively impacted because validating an identity has to be done through another service which can be separated from the requested service. This, in turn, may cause overhead stemming from additional message passing.

To authenticate user, the security solution must ensure that a consumer that has already been authenticated is accepted by the service provider. SAML facilitates this by binding security tokens to the request. Another issue is how rigid the authentication policy must be. This is important from an architectural point of view, because stronger authentication requirements (i.e., adding multiple levels of authentication) tends to negatively affect a number of QAs including performance (i.e., overhead from authentication calls and increased message size), and usability (i.e., complexity in managing certificates and tokens and the difficulty for users to create stronger passwords).

To authorize user, the architect must understand the business process and the technical security issues to design a meaningful authorization protocol. This requires the tradeoff analysis, which can be very difficult due to understanding of the access permission matrix needed by different participants [18].

For message security, WS-Security is the standard that is enforced in a SOA environment. WS-Security allows for end-to-end message level security. In addition to this, security can be embedded as part of the message. However, the downside is that complexity will be increased because it

requires careful management of parts needed to be secured. For message security, the design must be the one that provides a mechanism, which mitigates against a man-in-the middle security breach.

Architecting with the mechanisms for availability as a security requirement will have a negative effect on modifiability since the complexity involved in clustering and exception handling (the dynamism involved) will be significant. In general, availability as a security requirement has a very positive impact on usability because highly available system makes the system more usable.

3. Related Works

Security and its impacts on other quality attributes important in web-based (or ecommerce) applications together with the different types of mechanisms which address the security requirements must be analyzed from an architectural standpoint. These investigations must be done in order to arrive at a security solution that is optimal in achieving the required security level for a SOA based ecommerce system. And at the same time the proposed optimal solution will not compromise other important quality attributes vital to the implementation of successful SOA based-systems.

There are two broad categories of software architecture optimizations approaches: 1) quantitative and qualitative approaches. In [21], the authors describe the comprehensive literature review of 188 papers using search-based AI techniques to optimize quality attributes of software architecture of a system from a large set of architectural solutions. To manage the complexity of approaches, the optimization techniques focused only on specific aspects of a system.

The second approach is based traditional and qualitative approaches that utilize various aspects of a system to select the best possible architectural solutions using qualitative measurements such as good, very good, relative poor, etc. [7] [22] [23]. Our work basically fits in this category.

4. Architectural Optimization of SOA-Based Systems Using Qualitative Method

Security as a quality attribute is critical in web-based systems, but it is also one for which there are tradeoffs with other quality attributes in the system under construction [17]. Security is a before-fact and hence must be considered early on in the development process. Therefore, it is important that a tradeoff analysis be performed early on in order to understand the impact of security on the overall software quality and consequently select an optimal architecture that meets the security requirement without compromising other QAs. More specifically, the optimal architecture is the

based approach (SBA) discussed in [19] employed to validate the non-functionality of a typical web-based ecommerce system. SBA has been devised to properly document quality attributes that are important in an ecommerce system. SBA consists of 7 elements as follows: source (source of stimulus), Stimulus (request), Artifact, Environment, response, and response measure [19]. Table 3 depicts an example of a security scenario using quality attribute scenario [19,20]. The scenario attempts to capture a security requirement involving an invalid user who submits incorrect credentials to the system during normal model of operation; the request is denied every time this scenario occurs. More comprehensive tables (4, 5, 6) which show the security requirements and its impact on other QAs. For examples, table 5 documents security requirements using quality based scenarios (appendix A), and table 6 shows the relationship between security attributes important for SOA, their corresponding sub-attributes, solutions in using tactics, and finally architectural styles to realize those tactics.

To begin with, a utility tree (UT), which is a tree showing the relationship between QAs and their sub-attributes, is constructed [20]. UT, in turn, can be derived from quality-based scenarios (see table 5 and 6).

Table 3. Security Scenario example

Element	Description
<i>Source</i>	Invalid User
<i>Stimulus</i>	User submits incorrect password or username
<i>Artifact</i>	Ecommerce application
<i>Environment</i>	Normal operation
<i>Response</i>	User is denied access
<i>Response Measure</i>	Denial 100% of the time

A tactic is macro architectural solution method used to realize a quality-attribute [18]. This is done by selecting those refined lower level design decisions; the design decisions implement some aspect of the quality attribute requirements. Each tactic achieves one or more of the quality attributes. Example of tact in achieving high performance and availability is to incorporate concurrency and distributeness.

4.2. Using ATAM to Document Security Requirement of SOA

In this work, we have applied Architectural Trade off Analysis Method (ATAM) to specify security requirement [19][20]. In the simplest form, ATAM consists of four main phases; each phase, in turn, consists of steps: ATAM phases are: 1) scenario and requirements gathering, 2) architectural views, 3) model, and 4) tradeoffs analysis [19]. The first two phases, scenario and architectural design, discussed in the previous section. For the purposes of this paper the focus will be on the attribute specific analysis, sensitivity, and tradeoff identification of the security attribute.

One iteration will be performed to show how the security design can be modeled to enhance the desired qualities. The system resists attacks by employing the architectural patterns of publish-subscribe, repositories, and maintaining trust and integrity. The publish/subscribe pattern (P/S) [19] benefits SOA in a sense that new services can be easily published to a broker and any valid consumer will be able to discover and subscribe to it.

Modifiability benefits from P/S, but performance is negatively affected, because of the overhead that results from service lookup in directories and network delays. Since a key tactic in achieving the benefits of SOA is this loosely coupled distributed nature of services (facilitated by the publish/subscribe pattern and repositories) [2], it is important that this pattern, P/S, be used. However, this pattern, P/S, can be modified to allow one service to act as both an identity manager and broker where once a consumer discovers its required service, identity validation is done simultaneously for the service (see Figure 4). This strategy should reduce the intermediaries and hence improves performance without compromising modifiability and security.

Maintaining trust and integrity is vital to SOA and it is a pattern which addresses authentication, authorization, confidentiality and data integrity. Encryption and digital certificates are popular methods of ensuring confidentiality and integrity. Encryption adds overhead which negatively affects performance. Encrypting messages which involve external entities cannot be compromised and so any enhancement would have to be done with the performance attribute. One possible enhancement would be to increase the resources such as server processing power, which may be a costly venture dependent on business constraints.

FIM enables end-to-end authentication which is good for usability but it may compromise security. With end-to-end authentication, a consumer after initial authentication does not need to re-authenticate regardless of the number of additional services (internal or external) that may be accessed in order to complete a task or transaction. For example, if on one of the hops a hacker intercepts a message, that hacker may be able to access a service because the authentication for the hacked user is assumed to be validated by 'downstream' services. Point-to-point security will eliminate this kind of security breach, but it will compromise usability, because a user may have to re-authenticate multiple times in a single session; this may impact the unique benefits of SOA. In another sense, point-to-point security may also be a bad strategy, because it is forcing the user to input additional credentials; this increases the opportunities for identity theft and phishing by increasing the number of times a user has to enter log-on information in forms etc.

Another important consideration involves the availability quality attribute. For the scenario is unavailable, consumers will have to re-discover that service. This eliminates the possibility of spoofing or service masquerading, because the consumer would have to be re-authenticated. This may however affect reliability because if an order was being

processed and payment was already made, recovery should not result in the consumer being billed twice for the same order for instance.

Caching is a popular tactic in client/server architectures which could be employed in a SOA setting that could keep a record of the transaction state that is checked upon re-initiation with a service. Figure 5 shows the revised version of architecture illustrated in figure 4. The figure has the incorporated caching as part of the interaction between a consumer and a service provider. In addition to addressing the problem security and reliability, caching may also reduce contention and demand for server (service) resources, thus improving other QAs such as performance and scalability.

Figure 5. SOA model after ATAM is applied

The above analysis demonstrated how ATAM can be used to discover various tradeoff points, which can be negative and positive [19]. Another round of ATAM iteration is still needed to re-evaluate and incorporate the changes for those attributes, which have been affected by the security analysis. For instance, the way security is handled when a service fails (i.e., requiring rediscovery and subsequent re-authentication) leads to the addition of caching as a pattern to benefit reliability, performance and security. Therefore, both performance and reliability need to be re-evaluated to incorporate this change.

An ecommerce system based on SOA involves the connection of many systems, external entities and technologies, which adds complexity that must be considered when designing the system's architecture. This is why a method such as ATAM is needed evaluate any desired quality attribute perspective. Such a method can uncover

tradeoffs and sensitivities, which can impact the system or other quality attributes in negative or positive ways.

In this work, the analytical framework provided by ATAM helps us to determine the useful characteristics of each of the architectural patterns and highlights their costs and benefits [20] (see table 6). The ATAM also helps us (architects), to determine an architectural tradeoff point, which in turn provides a better understanding of the limitation of each option. The end result is the development of informed action plans for modifying the architecture, leading to new evaluations and possibly new enhancements using iterations of the method.

The complexity that is intrinsic in most real-world software design demands an architecture tradeoff analysis. As was the case in this paper, each step of the method answers some design questions and raises others.

5. Conclusions and Future Work

This work discusses the current works regarding the security requirements for ecommerce systems that utilize SOA. To this end, we discussed security requirements and also examined the mechanisms used to implement those requirements. In this work, we examined how security requirements have a significant bearing on the architectural design of the system. The nature of SOA dictates that after understanding the security requirements, the engineer must accommodate security in the system's architecture in a concrete way such that security as a quality attribute is not isolated in a way that will affect other NFRs. The possible effect each security mechanism may have on other QAs relevant to SOA has been discussed using ATAM approach, and Quality Attribute Scenarios.

This work is not by means complete. As such, the limitations of our work can be translated into what the future work will entail. For one thing, to select optimal SOA, we need to pay more attention on both qualitatively and quantitatively approaches for analyzing security schemes to fully address security requirements so that their impacts on the overall system quality is clearly defined.

In addition, future work demands experimentations and testing using case studies to validate our proposed approach for its usefulness, its practicality to identify optimal solutions automatically. The selected solutions, in turn, must not only meet security requirements, but also must satisfy other secondary NFRs (e.g., performance, availability) important to successful deployment of SOA used in the design and implementation of today's web-based systems.

Appendix A

Table 4. Security requirements' impact on different qualities attributes (Sub-QA vs. QA)

Requirements	Quality Attributes						
	Performance	Usability	Modifiability	Availability	Reliability	Interoperability	Scalability
Identity	7	7	9	7	NA	9	10
Authentication	5	5	7	7	7	9	10
Authorization	5	5	7	7	7	9	10
Integrity	5	7	5	10	10	7	9
Confidentiality	5	7	5	10	10	7	9
Availability	NA	10	5	10	7	NA	7

Key:

8-10: Very Positive

6-7: Positive

4-5: Negative

0-3: Very Negative

NA: Not Applicable

Table 5. Scenarios expressing various quality attributes

Scenario	Source	Stimulus	Artifact	Environment	Response	Response Measure
Security 1	Invalid User	User submits incorrect password or username	Ecommerce application	Normal operation	User is denied access	Denial 100% of the time
Security 2	Authenticated User	Attempts to access a service for which the user is a part of the group which is authorized	Ecommerce application	Normal operation	User granted access	Granted access 100% of the time
Security 3	Attacker (invalid user)	Attempts to access confidential customer data	Ecommerce application	Normal operation	Attacker cannot break encryption and data is not changed. System logs event and notifies system administrators	This is done 100% of the time
Security 4	External to system	Denial of Service attack launched	Ecommerce application	Normal operation	Attack detected and any inconsistencies restored	This is done 100% of the time
Usability 1	Authenticated user	requests a service with correct credentials	External Service	Normal operation	Application grants access to user without user needing to re-enter credentials	Granted access 100% of the time
Usability 2	Registered User	Attempts to set password which does not meet criteria	Ecommerce application	Normal operation	Password not accepted and user notified that password complexity is insufficient	This is done 100% of the time
Modifiability 1	System administrator	Adds a new service from an external business partner to provide an additional payment method	Ecommerce application	Business partner is familiar with the necessary interfaces and web services used	New service is integrated into the system	No more than two person days of the system's team effort is required for implementation (not including legal and

						financial agreements)
Performance 1	User	User submits an order for an item	Ecommerce application (Order Processing Module)	Normal operation	The system notifies user that order has been accepted and successfully processed	System responds to user in less than 4 seconds
Performance 2	Users	10,000 users concurrently access order processing sub-system	Ecommerce application (Order Processing Module)	Normal operation	The system is able to serve all users	System responds to all user requests in less than 5 seconds
Reliability	System failure	A user had submitted a credit card payment just before system failed	Ecommerce application	Failure Mode	The system is restored to a consistent state and users payment is accepted and user billed only once	This is done 100% of the time in such cases
Availability	Internal to system	Hardware fault	Ecommerce application	Normal operation	System administrator is notified and redundant server is given control and all current consistent transaction remain in a consistent state	Fault is detected and fault recovery action taken within 1 minute
Interoperability	System administrator	Adds a new service from a business partner which has a different software platform and web service protocols	Ecommerce application	New partner provides interfaces and vice versa	service easily incorporated into system and works well despite the heterogeneous platforms	New Service implemented within 2 person days
Scalability	System administrator	Changes made to system to handle 100,000 more concurrent users	Ecommerce application	Normal operation	Changes made and system performance still satisfactory	Changes made within 5 person days with interruptions to live system restricted to 1 hour.

Table 6. Showing Tactics along with the architectural patterns

Attribute	Sub-Attribute	Tactic	Pattern to Achieve Tactic
Security	Resisting Attacks	Breaking Dependency Chain, Identity Management and, authorization, authentication, Message Security (integrity and confidentiality)	publish-subscribe pattern, repositories, maintain trust and integrity
Security	Recovering and Detecting Attacks	Restoration, Identification	Limit exposure, intrusion detection, monitoring, restoration, audit trail
Modifiability	Changes to the system at runtime	Preventing ripple effect, localize expected modifications,	Raise abstraction level, publish-subscribe pattern, repositories, broker, modularity, high cohesion, low coupling, localization
Availability	Maximum uptime	Fault detection, Recovery, Prevention	Monitoring, restoration, redundancy
Performance	Low latency	Resource demand, resource management, resource scheduling	Reduce computational overhead, Parallelism, abstraction, increase resources
Usability	Ease of use of system	Accessibility, user friendliness	maintain trust, parallelism, dependability (security, performance, reliability, availability)

REFERENCES

- [1] A. V. Konstantinou, T. Eilam, M. Kalantar, Alexander A. Totok, W. Arnold, E. Snible. "An architecture for virtual solution composition," in Proc. 3rd international workshop on and deployment in infrastructure clouds Virtualization technologies in distributed computing, Barcelona, Spain.VTDC, June 15, 2009,
- [2] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, "Software Architecture: Foundations, Theory, and Practice," John Wiley and Sons, 2009.
- [3] J. Hurwitz, R. Bloor, C. Baroudi and M. Kaufman, "Service Oriented Architecture For Dummies," John Wiley & Sons, 2007.
- [4] M. Matsumura. "Intentional SOA for Real-World SOA Builders," Software AG, May 2007.
- [5] J. Bih, "Service Oriented Architecture (SOA) A New Paradigm to Implement Dynamic E-business Solutions," Ubiquity , 4, August, 2006.
- [6] Gib Trub, L. O. (2008, September). Global report on SOA/Web services security initiatives," http://www.ca.com/files/industryresearch/gmg_world_wide_soa_research_survey_08.pdf
- [7] H. Reza and E. Grant, "Quality-Oriented Software Architecture," International Conference on Information Technology (ITCC'06), Las Vegas, 2006.
- [8] E. P. Taylor, "Security in a Loosely Coupled SOA Environment," Retrieved September 29, 2010, from <http://www.developer.com/design/article.php/3605836>
- [9] National Security Agency. (n.d.). Service Oriented Architecture Security Vulnerabilities Web Services. Retrieved October 3, 2010, from www.nsa.gov/snac: http://www.nsa.gov/ia/_files/factsheets/SOA_security_vulnerabilities_web.pdf
- [10] IBM, "Understanding SOA Security Design and Implementation," IBM, November 2007. Retrieved from <http://www.redbooks.ibm.com/redpieces/abstracts/sg247310.html?Open&pdfbookmark> on November 20, 2010.
- [11] J. Oltsik, www.enterprisestrategygroup.com.
- [12] IBM, "Implement federated identity management to help accelerate your on demand business: www.ibm.com: [ibm.com/tivoli/products/federated-identity-mgr](http://www.ibm.com/tivoli/products/federated-identity-mgr).
- [13] E. Pulier and H. Taylor, "Solutions to SOA Security," Retrieved October 1, 2010, http://www.developer.com/design/article.php/10925_3607471_1/Solutions-to-SOA-Security.htm.
- [14] Z. Qin, H. Yi, L. Shundong, D. Jinchun, Yan Lixiang and Q. Jun, "E-commerce Architecture and System Design," in Introduction to E-commerce, Beijing, China: Tsingua University Press, 2009, , pp 271-301
- [15] K. Soo Hoo, A. W. Sudbury and A. R. Jaquith, "Tangible ROI through Secure Software Engineering," Secure Business Quarterly 1, 2 (2001).
- [16] J. Li and A. H. Karp, "Access Control for the Services Oriented Architecture," ACM, Fairfax, Virginia, 2007.
- [17] P. Bianco, R. Kotermanski and P. Merson, "Evaluating a Service-Oriented Architecture," Technical Report - Software Architecture Technology Initiative, Software Engineering Institute, Carnegie Mellon University, September 2007.
- [18] G. Boella, L. van der Torre, "Permission and Authorization in Normative Multiagent Systems" in ICAIL '05 Proceedings of the 10th international conference on Artificial intelligence and law, Bologna, Italy, June, 2005.
- [19] L. Bass, P. Clements and R. Kazman. "Software architecture in practice," Pearson Education Inc., 3rd edition, 2013.
- [20] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson and J. Carriere, "The Architecture Tradeoff Analysis Method," Software Engineering Institute, Carnegie Mellon University, July, 1998.
- [21] A. Aleti, B. Buhnova, L. Grunske, A. Koziolok. Software Architecture Optimization Methods: A Systematic Literature Review. IEEE Transactions on Software Engineering, vol.39, no.5, May 2013.
- [22] H. Reza, D. Jurgens, J. White, J. Anderson, J. Peterson. An architectural design selection tool based on design tactics, scenarios and nonfunctional requirements. IEEE International Conference on Electro Information Technology (EIT), May 2005.
- [23] A. Martens, F. Brosch, R. Reussner Optimising Multiple Quality Criteria of Service-Oriented Software Architectures. International Workshop on the Quality of Service-Oriented Software Systems (QUASOSS'9) Amsterdam, August 2009.
- [24] R. Elmars, and S. Navathe. Fundamentals of Database Systems, 6th edition. Addison Wesley. 2011.