Exploring Security and Privacy Risks of SoA Solutions Deployed on the Cloud

Abdullah Abuhussein, Harkeerat Bedi, Sajjan Shiva
Department of Computer Science
The University of Memphis
Memphis, USA
{bhussein, hsbedi, sshiva}@memphis.edu

***Abstract*— It has been widely accepted that service oriented architecture (SoA), has been a promising approach for business development and growth. SoA principles (also known as SoA qualities) attempt to guide development, maintenance, and usage of the SoA. These principles provide benefits like: ease of reuse, service automation, and lowering integration costs. However, they can also lead to security issues. These issues are augmented especially when SoAs are deployed in multi-tenancy third party clouds. SoA has benefited from the existence of cloud computing (CC) as it provided SoA with a flexible deployment medium. However, the advantageous collaboration of SoAs and CC has led to a larger set of privacy and security issues (e.g. compliance issues, QoS issues). Additionally, we observe newer kinds of security and privacy risks that are now required to be monitored and mitigated. In this paper we highlight the security and privacy challenges associated with the utilization of the SoA principles on cloud based solutions. We identify the origin and severity of these issues followed by several recommendations to guide the utilization of SoA principles in off-premise clouds.**

Keywords— service oriented architecture, cloud computing, security, privacy.

I. INTRODUCTION

Service oriented architecture (SoA) has provided the software development industry with flexibility and capabilities like bridging business and IT, lower cost by implanting reusability and providing autonomy in software services. SoA is defined as a set of architectural tenets for building autonomous yet interoperable systems [1]. SoA defines eight principles that guide its development, maintenance, and usage. These principles are: abstraction, autonomy, composability, discoverability, formal contract, loose coupling, reusability and statelessness [2].

SoA principles offer a number of advantages (e.g. reusability, reduce integration and maintenance costs) [3] and therefore they can also be represented as qualities of SoA. SoA principles played a significant role in the adoption of the SoA paradigm in the last decade [4]. The tightly coupled nature among services in systems preoccupied developers' minds. The SoA principles alleviated these issues and enabled the software developers to produce software components that are reusable, autonomous and customizable.

In some cases, SoA principles like abstraction and independency of services help to reduce services exposure to the outside world and therefore reduce security risks. However, SoA security in general remains an issue due to the medium they are deployed on and delivered through.

Deployment and delivery of SoA can be performed using several methods. At present, cloud computing (CC) has become the most prominent means of SoA deployment and delivery. CC provides benefits like resiliency, elasticity and reliability but also raises several security and privacy risks [5]. The combination of SoA and CC together produces a larger set of security and privacy risks. CSA Notorious 9 of 2013 stated that Clouds that share PaaS, SaaS, and IaaS are more vulnerable [6]. This is generally the case when deploying SoA solutions on public clouds. .

The future of SoA is tightly interlinked with CC due to the use of Internet, changing nature of the customers, and the impact of social networking (e.g. sudden high consumer demand/traffic that was not an issue before). To handle such situations that are very common now, SoA needs CC to cater the needs of this newer generation of consumers. Therefore, SoA benefits from CC features like agility, scalability, and reliability to operate and conveniently perform upgrades to meet the consumer's needs.

The current research is primarily geared towards finding the security and privacy issues of SoA [7]. Researchers in [8] and [9] have shown some of the security challenges in deploying SoA in the cloud. In this work we study the relationship between the utilization of the SoA principles and the emersion of security and privacy issues . We also show the origin of these security and privacy issues then provide recommendations on how to secure the deployment. SoA is widely practiced today. Now, most companies are focusing on building services that are independent, can be discovered and requested automatically by consumers, and are able to monitor and manage themselves. However, this requires an extensive effort towards balancing the utilization level of SoA principles, while minimizing exposure to security and privacy risks.

Section 2 explores SoA deployments over the past decade. We will also go over current different form of delivering SoA. In section 3 we illustrate how utilizing SoA principles in the cloud may lead to potential security and privacy vulnerabilities. We show the severity of such risks and describe how they are originated. We also present various recommendations to

overcome these risks. Section 4 provides our observations on the presented problem, proposed solution and future work.

II. SoA Deployment and Delivery

Traditionally, SoA solutions like Customer relationship management (CRM) , Enterprise resource planning (ERP), payroll, etc. were deployed on private machines that lie within the premises of the end user's organization (on-premises) or deployed within the SoA provider's organization (off-premises) and accessed by end users through the Internet. Emergence of CC served to meet developers' increasing demands of infrastructure for their SoA solutions. With the advent of CC, the entities responsible for development of SoA and those of infrastructure became separated. This leads to change to the nature, severity and/or existence of SoA vulnerabilities. It also leads newer kinds of issues and risks that were not present earlier (e.g. governance and compliance issues, etc.) [5].

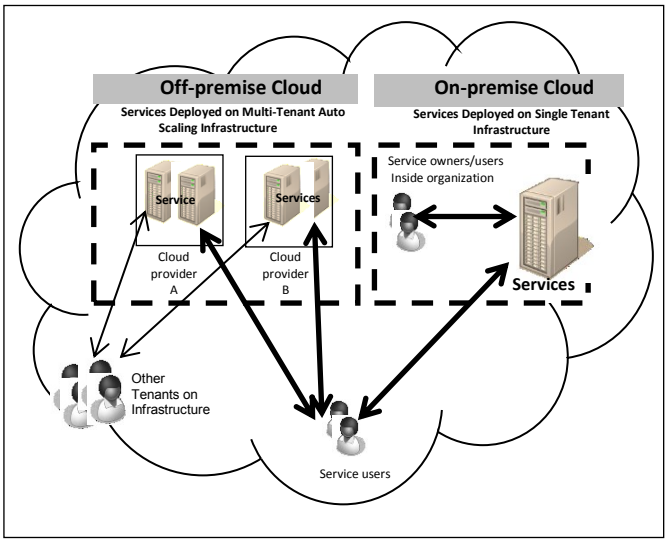

Fig. 1. SoA deployed on off-premises versus on-premises cloud computing

Fig. 1 shows the two possible cases of deploying SoA on the cloud. On the right, SoA is deployed on a cloud model that is on-premise. Services are hosted by the organization's infrastructure and the infrastructure is provisioned and managed by the organization itself. Since the entity responsible for the development of SoA and the infrastructure are the same, the risks are limited.

The case on the left shows off-premise cloud computing infrastructure being used to host the SoA services. Infrastructure is provisioned and managed by the CC service provider. In this case, features like auto-scalability and multi-tenancy are offered to provide SoA developers with as much infrastructure as they need at low costs. However, SoA developers share the infrastructure with other tenants. Also, services might demand more resources and scale up on more VMs on the same physical machine or distant machines on different regions. Moreover, CC service brokers might recommend a different service provider every time additional infrastructure is requested. These scenarios lead to new kinds of security issues and thus risks that were not present before.

CC providers do offer isolated hardware for interested consumers. This in turn would overcome the multi-tenancy drawbacks although at higher prices [9]. Nevertheless, denial of service (DoS) attacks, which are the CC's fifth top threat in 2013 [10], are a serious concern in isolated hardware [11].

III. Risks of CC on SoA Oriented Solutions and Recommendations

Despite the benefits that SoA principles add to the traditional software development life cycle, they bring new challenges. Some of these challenges are security and privacy issues that take place due to the technologies used in SoA based service development and operation. XML is the core of SoA and is not inherently secure. SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description Discovery and Integration) are all based on XML. A well-known XML exploit is the XML rewriting attack. Although WS-Security [12], WS-Policy [13] and other standards aim to secure the XML based application and avoid these attacks, the national vulnerability database [14] showed 14 SOAP vulnerabilities, and 4 WSDL related ones in 2013.

Beside the security problems of SoA [15], the fact that CC is becoming one of the most prominent means of SoA deployment worsened matters. Table 1 shows the (8) SoA principles in the first column, application area in column two, alongside the technologies required to foster each one of principles in column three. The forth column highlights how the deployment of SoA in an off-Premise CC can change the nature of the SoA vulnerabilities and the severity of security issues and risks. In the same table, we map these risks to the CSA notorious nine cloud attacks observed in 2013.

Technically, the application of the 8 service-oriented-architecture principles can be segregated into two categories based on the part of SoA that they are utilized in. The first category is for the principles that can be utilized in service contract and registry like: abstraction, discoverability, and formal contract. Other SoA principles like: (composability, Autonomy, loose coupling, reusability, and statelessness) can be utilized in services themselves, which is the second category. We need categorization to enable exploring technical security and privacy issues. For instance, WSDL and UDDI together with SOAP are standards in service registry [16]. Knowing these standards we can look for security breaches that can be exploited using them. Matters can be even worse in an exposed off-premise cloud computing infrastructure.

Below we explain the SoA principles in brief, discuss the security and privacy issues related to the utilization of each principle and suggest several security recommendations.

TABLE I. SECURITY ISSUES RELATED TO THE UTILIZATION OF EACH SOA PRINCIPLE IN OFF-PREMISE CLOUD

SoA	Applicable to	Vulnera-ble Tech	Risks due to CC [5]	CSA Notorious 9 Threats [6]
Abstraction	Service Contract	SOAP UDDI WSDL XML HTTP	1. Exposure 2. Redundancy and integrity issues 3. Access control 4. Trust	1.0 Data Breaches 4.0 Insecure Interfaces and APIs 3.0 Account or Service Traffic Hijacking
Discoverability	Service contract and Service Registry	SOAP UDDI WSDL XML HTTP	1. Exposure to cloud risks 2. Authentication and access control	1.0 Data Breaches 4.0 Insecure Interfaces and APIs 3.0 Account or Service Traffic Hijacking
Composability	Services	TCP communi-cation XML	1. Lack of standards 2. QoS. 3. Availability 4.Trust 3. Compliance 4. Governance	1.0 Data Breaches 4.0 Insecure Interfaces and APIs 3.0 Account or Service Traffic Hijacking 5.0 Denial of Service 6.0 Malicious Insiders
Autonomy	Services	TCP communi-cation XML	1. Lack of standards and safe patterns 2. Exposure to cloud risks	1.0 Data Breaches 4.0 Insecure Interfaces and APIs 3.0 Account or Service Traffic Hijacking 9.0 Shared Technology Vulnerabilities
Formal contract	Service Contract	SOAP UDDI WSDL XML HTTP	1. Trust 2. QoS 3. Authentication and access control	6.0 Malicious Insiders 5.0 Denial of Service
Loose coupling	Services	TCP communi-cation XML	1.QoS 2.Exposure to cloud risks 3. Data Interception	5.0 Denial of Service 4.0 Insecure Interfaces and APIs 1.0 Top Threat: Data Breaches
Reusability	Services	TCP communi-cation XML	1. Compliance 2. QoS 3. Exposure to cloud risks 4. Authencation and access control	6.0 Malicious Insiders 5.0 Denial of Service
Statelessness	Services	TCP communi-cation XML	1.Exposure to cloud risks 2.Compliance 3.Trust	1.0 Data Breaches 4.0 Insecure Interfaces and APIs 3.0 Account or Service Traffic Hijacking

A. *Abstraction*

This principle of SoA aims to hide the logic behind services from the outside world, while providing descriptions in the service contract. To utilize service abstraction service developers need to categorize service meta information into (1) Functional (2) Technology (3) Programmatic and (4) Quality of service categories. Service meta information is then used to create service contracts and service registry. After that, an access control procedure is applied to limit open access to service owners only and give controlled access to others including interested consumers.

Less abstraction indicates more information about the logic of the service and therefore more exposure and more vulnerability. However, over utilization of abstraction indicates over-hiding service logic and technology information and therefore, limits the reusability of the service which leads to creating similar services and raises redundancy and integrity issues.

Another possible problem is access control. Traditionally, service owners have access to all parts of the service, like design specifications, source code, etc. However, the off-premise CC nature exposes the SoA and increases the possibility of account hijacking. For example, an attacker who is successfully able to hijack the account of a service owner will have access to all the parts of a service.

Also as a result of abstraction, service consumers and program designers will not be aware of composite services. Due to this, service consumers won't know what is wrong with the service once a composing part of the whole service goes down as we will see later in the composability principle.

Due to the exposed nature of cloud computing these issues will have a bigger chance to occur. Thus it is recommend that service developers and implementers balance the amount of abstraction and monitor services for risks appropriately. One should look for CC with good access management to mitigate the risk of account hijacking.

B. *Discoverability*

Service registry is a central repository of service meta data that is hosted on off-premise cloud. Service consumers access service registry to find desired functionalities. That's how a consumer discovers a desired service and then retrieves the service contract. Then the service will be ready for usage.

The discoverability principle enforces that services have communicative meta-data so that they can be efficiently discovered and interpreted. One of the ways this principle is implemented is through using the Web Proxy Auto-Discovery Protocol (WPAD). Browsers in an organization are required to be supplied with the same proxy policies. These polices are created and maintained by using a configuration file based on the Proxy auto-config (PAC) standard. WPAD is used to discover the URL of this configuration file so that proxy policies on all browsers in an organization can be set concurrently.

With SoA being implemented on the cloud, we are adding more exposure to these vulnerable PAC files. Previously an attacker had to be within a company's network to attack these PAC files. Now due to the ubiquitous nature of the cloud, this is not the case. The above two problems become difficult when the service broker comes into picture as this adds another layer of communication exposed to cloud vulnerabilities. In 2012 a summary by the national vulnerability database shows WPAD) functionality in Microsoft .NET Framework 2.0 SP2. WPAD was not validating configuration data that is returned during

acquisition of proxy settings. This vulnerability may allow remote attackers to execute arbitrary JavaScript code.

In 2013, the same database reported a Cross-site scripting vulnerability in the UDDI administrative console in IBM WebSphere application server. UDDI is the core of the registry along with WSDL and SOAP [14].

Thus it is recommended to use some form of authentication among services, or between services and the service browser. Also, balance the amount of discoverability, and monitor the services. Another recommendation can be to enable automatic updating for your services to benefit from security frequent patches provided by SoA vendors.

C. Composability

This principle encourages that services become effective participants for composition. It promotes composing new solutions by reusing existing services. However, lack of standards in how to securely and safely compose a service from other services on a cloud is a possible security issue due to the multi-tenancy nature of the cloud. As mentioned before, service contracts hide service composability details so; consumers can never tell whether the service is a standalone service or composed of others.

Availability of the composing services will affect the availability of the parent service. Moreover, quality of the service (QoS) depends on the QoS of the CC infrastructure.

Also, QoS of a composed service depends on the QoS of the sub-services and the infrastructure they are deployed on. Because of the multi region infrastructure of the cloud, compliance and distributed ownership security issues may also apply if the regulations in the countries of the composing sub-services do not match.

Moreover, too composability denotes more transit time due to communication among composing services. Attackers can steal or modify information if not protected while in transit. Again, the exposed nature of the off-premise cloud computing may worsen matters [17].

Therefore, it is recommended to follow safe service composition patterns when composing solutions [8]. It is also essential to review SLA of the underlying CC infrastructure and make sure that hosting countries have no problem with the content and the function of the service. Auditing the underlying CC service for hypervisor security is another recommendation. It helps to overcome multi-tenancy security issues. Encryption and digital signature of data on transit must be considered too in order to secure data in transit. Another recommendation is to balance the amount of composability, and monitor composed services and participant services.

D. Autonomy

This principle of SoA aims to build services with self-control over the logic they contain. When services are made autonomous, they become independent of the underlying technologies, i.e., these services will be resilient to the issues in these technologies. But at the same time, since they can be implemented on more diverse platforms, we are also increasing their exposure to security flaws of these platforms. This will increase the possibility of compromising services due to variations on the underlying technologies.

Service autonomy implies greater emphasis on explicit management of trust between applications to avoid malicious modification and avoid service integrity issues especially due to the nature of public clouds [18].

The Autonomous nature of services implies that services communicate to maintain control over the resources and to coordinate with other components of the SoA. A significant increase in the messaging must occur as service autonomy increase which will also increase exposure to vulnerabilities on off-premise CC. The greater the number of resources, that are accessible for attack, the greater the attack surface and therefore, the more insecure the software environment [19].

The recommendations to overcome these issues are to do a thorough assessment of whether or not it is necessary to increase autonomy at the expense of exposure. It is also important to verify the security practices that can be applied to the underlying technologies. A strong input validation is required to verify input from other applications. Finally, apply WS-Security to achieve trust among autonomous services and applications.

E. Formal contract

When a service is implemented as a Web service, the service contract is normally comprised of a WSDL definition, multiple XML schema, policy definitions, as well as supplementary documents, such as an SLA. This principle enables a standard design of services in terms of policies, WSDL, and XML Schema within the service inventory.

As aforementioned, the formal contract principle is utilized on service contracts. So, it is also subject to WSDL, XML, and SOAP security issues.

In cases where SLA parameter deals with response time and there is a delay, the service consumer would not know whether the problem lies with the service or the CC infrastructure. The service consumer will have to trust the SoA provider for the promised QoS. Moreover, the QoS could get worse if the service is a composed service [20].

Standardization of services within the inventory might give a pattern of how these services are built. This might lead to unveil information about the logic, and/or the technology used to build other services if one service is attacked.

To safely and securely apply this principle we present the following recommendations. The first one is to avoid automated tools when creating contracts as it might lead to inaccuracies. Verify the created contracts to make sure that the underlying infrastructure provides the promised QoS by the SLA. A good access control and authentication system is also required here to avoid illegitimate communication.

F. Loose coupling

Loose coupling enforces that services are built in such a manner that they are decoupled from their surrounding environment. Services must be designed in such a way that it is not tightly coupled to other services or resource. Decoupling a service from its environment has several advantages (e.g.

Hiding service implementation from attackers) however; it increases the message exchanges between the service and the environment. Deploying services on CC makes it worse since the messages are transmitted through the Internet which adds latency to response time and reduces throughput. Also, messages passing between two services or between a service and the service container can be intercepted as mentioned before [21].

Thus, it is recommended to use secure communication by applying encryption on transmitted data. Another recommendation is to use compression techniques to reduce the bandwidth and latency overhead.

G. Reusability

Services need to be as generic as possible so that they are of interest to multiple service consumers, however, larger granularity may lead to larger incompatibilities that might in turn lead to security issues. To utilize reusability, developers need to produce solution in forms of services with the intention of promoting reuse. Compliance issues can rise by producing reusable SoA services [22]. For example, rules and regulations in different countries can limit the extensibility of use of such reusable services. Another issue in enforcing reusability on off-premise deployed SoA is the difference in the CC infrastructure configurations. Different CC providers have difference configurations, thus QoS variance is expected. Also, changes to standards or upgrades applied to infrastructure may have a large impact on the security of services.

Therefore, it is recommended that SoA developers test services on various infrastructure configurations before releasing them to public. As suggested previously, a thorough walkthrough over the rules and regulation of countries hosing the CC infrastructure should be performed. It is also important to verify that the service data is lawful in all stages (input, process, output, and storage).

H. Statelessness

This principle of SoA promotes minimizing resource consumption by services. This is achieved by deferring the management of state information when necessary. In other words developers should try to avoid service consumption of resources so that services can handle more requests in a reliable manner. Also saving state in an external component requires additional infrastructure. On the cloud, since the external component can be placed anywhere, it becomes necessary to ensure that the latency limits are met. While communicating to and from different clouds, we are exposing the state of the service and increasing the message exchange between the service and its infrastructure.

Thus, similar to the recommendations provided for loose coupling, it is suggested to use secure communication by applying encryption. Also it is recommended to use compression techniques to reduce the bandwidth and latency overhead and thus, increase service availability.

IV. Conclusion

Service oriented architectures (SoA) and cloud computing (CC) are accelerating to provide consumers with reliable, resilient, and efficient solutions. Increasing the utilization of SoA principles indicates adding more qualities to applications however, it also exposes developed services to newer vulnerabilities. These vulnerabilities can occur due to the broad attack surface of these SoA solutions. In this paper we showed the importance of balancing and monitoring the services that utilize the SoA principles in off-premise cloud computing. We presented several security and privacy risks (challenges). We also provided recommendations that developers of SoA in public cloud computing need to consider to overcome these risks.

In this work, we have demonstrated how enforcing the eight principles of SoA can add risks when deployed on off-premise CC environments. Porting SoA to the cloud will not have only benefits, but it will also add some security risks that developers need to consider. Further investigation is needed on additional security risks and safe SoA patterns to strengthen the SoA industry. A common issue when developing SoA is the overhead of utilizing these principles. In addition, we have seen other security and privacy issues like exposure, QoS, trust, compliance, data interception, and availability. There are general recommendations [23] when porting SoA to an off-premise CC like (1) look for secure CC services which exhibit adequate security attributes [24, 25] to overcome the most possible security issues. (2) Test services on different infrastructures and different scenarios before releasing them to be used by public and (3) encourage SoA developers to find and publish safe SoA development patterns so that others can benefit from them.

We are now investigating the factors that affect the over-utilization of the SoA principles. We also intend to identify safe SoA utilization patterns that can help others in overcoming the security risks presented in the paper.

References

[1] Jammes, François, Antoine Mensch, and Harm Smit. "Service-oriented device communications using the devices profile for web services." In Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, pp. 1-8. ACM, 2005.

[2] Thomas Erl: SOA Design Patterns, Prentice Hall PTR; 1 edition (2009)

[3] Bean, J. :SOA and web services interface design: principles, techniques, and standards. Morgan Kaufmann. (2009)

[4] Inaganti, S., & Aravamudan, S. (2007). SOA maturity model. BPTrends, April.

[5] NITS, "Guidelines on Security and Privacy in Public Cloud Computing", http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144. pdf

[6] The Notorious Nine Cloud Computing Top Threats in 2013, https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_ Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdF

[7] F5, SOA: Challenges and Solutions- White paper, http://www.f5.com/pdf/white-papers/soa-challenges-solutions-wp.pdf

[8] Wei, Y., & Blake, M.. : Service-oriented computing and cloud computing: Challenges and opportunities. Internet Computing, IEEE, 14(6), 72-75. (2010)

[9] Pal, P., Atighetchi, M., Loyall, J., Gronosky, A., Payne, C., & Hillman, R. : Advanced Protected Services-A Concept Paper on Survivable Service-Oriented Systems. In Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2010 13th IEEE International Symposium on (pp. 158-165). IEEE. (2010, May)

[10] Ristenpart, Thomas, et al. "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds." Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009.

[11] Amazon Web Services, EC2: http://aws.amazon.com/ec2/faqs/

[12] WSS: http://www.oasis-open.org/wss/

[13] Web Services Policy Framework (WS-Policy): http://www.ibm.com/developerworks/library/specification/ws-polfram

[14] National Vulnerability Database: http://nvd.nist.gov/

[15] Phan, C. (2007, October). Service oriented architecture (soa)-security challenges and mitigation strategies. In Military Communications Conference, 2007. MILCOM 2007. IEEE (pp. 1-7). IEEE.

[16] García-González, Juan Pablo, Verónica Gacitúa-Décar, and Claus Pahl. "Service registry: A key piece for enhancing reuse in SOA." (2010).

[17] Venkatasubramanian, Nalini. "Safe'composability'of middleware services." Communications of the ACM 45, no. 6 (2002): 49-52.

[18] Cabrera, Luis Felipe, Christopher Kurt, and Don Box. "An introduction to the web services architecture and its specifications." Microsoft, Microsoft Technical Article, Oct (2004).

[19] Manadhata, Pratyusa K., Kamie M. C. Tan, Roy A. Maxion, and Jeannette M. Wing. " An Approach to Measuring a System's Attack Surface." CMU Technical Report CMU-CS-07-146, August 2007.

[20] Bianco, P., Lewis, G. A., & Merson, P.. Service Level Agreements In Service-Oriented Architecture Environments (No. Cmu/Sei-2008-Tn-021). Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst. (2008)

[21] Joshi, Rajive. " Data-Oriented Architecture: A Loosely Coupled Real-Time SOA." Real-Time Innovations, Inc., August 2007.

[22] Is SOA Being Pushed Beyond Its Limits? , from: http://msdn.microsoft.com/en-us/architecture/aa699422.aspx

[23] Microsoft: Security Fundamentals for Web Services, http://msdn.microsoft.com/en-us/library/ff648318.aspx#WebServicesSecurityPrinciples

[24] Abuhussein, Abdullah, Harkeerat Bedi, and Sajjan Shiva. "Evaluating security and privacy in cloud computing services: A Stakeholder's perspective." Internet Technology And Secured Transactions, 2012 International Conferece For. IEEE, (2012)

[25] Softwareag.com. " Best Practices for SOA Governance User Survey." Software AG, Summer 2008.