Considerations Regarding Designing and Administrating SOA Solutions

Vlad DIACONITA
The Bucharest Academy of Economic Studies
diaconita.vlad@ie.ase.ro

Solutions like SOA, Cloud, SaaS, Iaas or PaaS are not only buzzwords, they became a business reality because they are relative cheap and easy to use. SOA and Cloud are tightly linked because most cloud solutions are being defined using SOA making them feasible from the business perspective, because it's hard to move to cloud when you are using a tightly coupled architecture. Big companies such as Oracle, Microsoft, IBM or Amazon offer many commercial solutions providing software as a service, as well as hosted and managed alternatives to classical deployment. For firms that are building private clouds and for service providers that are building public clouds, diverse solutions are offered by the big players for platform as a service and infrastructure as a service.
Keywords: *SOA, web services, modeling, cloud*

Introduction

From an evolution perspective, some authors say ([1], [4]) that the last decade is marked by the development of SOA and cloud computing. The main characteristic of SOA is the ability to be reused in various applications, using service communication by sending information in a loose coupled environment [5]. The idea of exposing resources as web services, making them accessible is older but building the components, tools and infrastructure to accomplish this was the problem. Development of virtualization in organizational environment allowed hooking up applications with various operating systems, enhancing the portability. Like shown in [6], moving towards implementing Web applications that consume a large variety of Web services is the current hype in application space and the mobile application market is searching for solutions to empower mobile devices with Web services integration while minimizing the existing performance issues. By using service-oriented strategies, companies even starting with few resources can run their businesses entirely using cloud. For example, as part of Amazon's AWS Relational Database Service (RDS) someone can rent an Oracle database license if they don't have it on their own. This starts at 16 cents an hour for a small instance going up to $3.96 for a quadruple extra large. The price includes the software, the hardware resources and Amazon's RDS management capabilities [9].

Cloud and SOA

Researchers are trying to bring SOA, web services and cloud services under a common terminology and approach.

Gartner defines cloud computing as a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service to external customers using Internet technologies. This is a slight revision of Gartner's original definition published in 2008. Gartner has removed *massively scalable* and replaced it with *scalable and elastic* as an indicator that the important characteristic of scale is the ability to scale up and down, not just to massive size. The five attributes of cloud computing are: service-based, scalable and elastic, shared, metered by used, uses internet technology [16].

SOA is providing the architecture, governance and orchestration for services to be delivered using cloud mechanisms, both internally and externally across the Internet.

An author that wrote many books on SOA, Dave Linthicum has shown that cloud computing should be a logical extension of SOA best practices [14].

As explained in Thomas ERL' latest book [13], the rise of cloud computing put SOA back in the spotlight, *even organizations that shunned SOA now have one. It's called the cloud.* He also renamed his online publication, formerly called SOA Magazine, as Service Technology Magazine:

The SOA Magazine has been renamed to the Service Technology Magazine. Articles will of course continue to be focused on service-oriented architecture and service-orientation, but will also address topics related to service technology innovations, such as those fostered by the on-going emergence of cloud computing platforms. I'd like to invite you all to contribute your expertise as we continue to explore how this new generation of architectural models, paradigms, and technologies is changing the way we view and leverage IT [15].

So, we can consider *service technology* as the common identifier.

Defining SOA

There are many definitions of SOA targeted to different audiences (managers, designers, programmers, sales persons etc).

When an enterprise level SOA application is being developed, many people are involved, some of whom are end-user developers. For example, business process experts know about the business context but may not necessarily be professional programmers, and are often responsible for identifying and selecting which services will be used ([2],[3]).

Executives are increasingly frustrated with their inability to quickly access the information needed to make better decisions and to optimize their business [10]. To them, SOA can be showed as a set of services that can be exposed to customers, partners, and between the different departments of the enterprise. These services can be combined and recombined to serve the needs of the business. Applications serve the business because they are composed of services that can be quickly modified or redeployed in new business contexts, allowing the business to quickly respond to changing customer needs, business opportunities, and market conditions.

To an IT designer, SOA is the architectural solution for integrating diverse systems by providing an architectural style that promotes loose coupling and reuse.

To a programmer, SOA is a model where web services and contracts are used for interoperability. The web services, used as part of SOA, facilitate communication using messages, without detailed knowledge of each other's IT systems.

The term *service orientation* is often seen as identical to SOA but some authors [1] consider it broader and represent a way of thinking about services in the context of business and IT.

Like shown in [7] and [8], web services are self contained, modular business applications that have open, Internet-oriented, standards-based interfaces. They allow flexible and dynamic software integration that is often referred to as the *Find-Bind-Execute* paradigm. Using standard Internet technology, Web services facilitate cross-organizational transactions and thus outsourcing of software functionality to external service providers. Thus, service-oriented computing requires an infrastructure that provides a mechanism for coordination between service requesters and providers. Three main technologies are currently used to implement Web services: SOAP, WSDL and UDDI.

Everware-CBDI, a global technology consulting company sees Service Oriented Architecture (SOA) as the principles, patterns and policies that enable application functionality to be provided and requested as services published at a granularity relevant to the Service Consumer, which are abstracted away from the implementation using a single, standards-based form of interface. The evolution of the implementation strategies is offering services in SOA in the CLOUD. SOA should provide reference architecture for service classification, policy implementation and governance, contracts, determining sharing and generalization at many levels.

SOA can also be seen as a collection of services, classified into types, arranged into layers and governed by architectural patterns and policies as shown in figure 1.

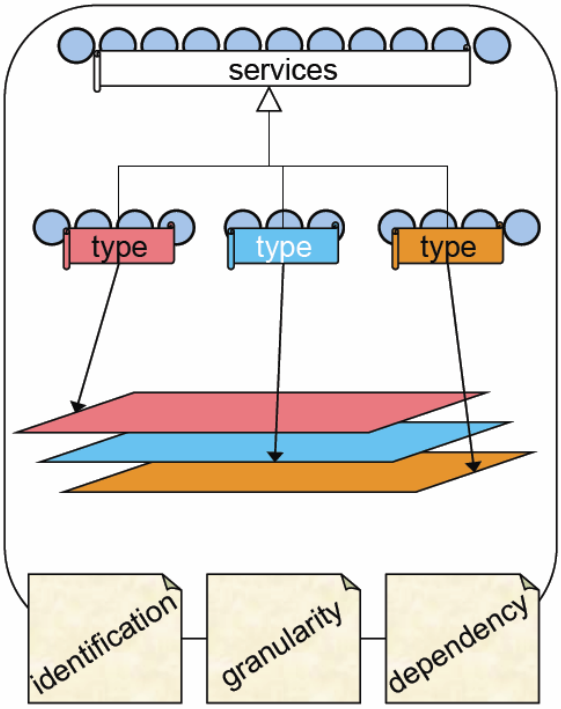

Figure 7. The service arhitecture, source: http://everware-cbdi.com/cbdi-forum

Modeling SOA Infrastructure

A SOA infrastructure can be divided into three categories: consumer, functional, and operational to provide an abstraction that can be used and reused across an organization. Infrastructure architects can use this model as a basis for determining the necessary software products or technologies necessary to provision that building block.

The consumer access component includes the infrastructure needed by the people to access the services, including: browsers, data channels and portals.

Internet browsers are used to expose functionality using a Web interface rendered on a user's browser.

A data channel is where consumers can provide or consume large amounts of data. The movement of bulk data over networks addresses raw unstructured data, structured data, images, and any large data that requires high performance. The infrastructure provides a software-based mechanism designed to move large data files using compression, blocking, and

buffering methods to optimize transfer times. Infrastructure architectures need to determine whether there are business needs for such a bulk data transfer component and provision accordingly.

A Web portal presents information from diverse sources in a unified way. The ESB makes possible the communication between service requestors and service providers. It enables the substitution of service providers or implementations transparent to service requesters. The ESB usually many ways to attach requesters and providers, and it allows intermediary services to be sequenced between them. The ESB can also supply an extensive set of capabilities dependent on business needs and implementation in areas like integration, communications, security, signal processing, QOS and service management.

In SOA, the integrations are pushed outward, toward the applications themselves, leaving the bus to speak a standardized language [11]. Like modeled in figure 2, an ESB is a connectivity infrastructure for integrating applications and services, while EAI focuses on the applications integration. ESB infrastructure does more than integration because it performs routing of messages between services, converts transport protocols between consumers and providers, transforms message formats between requesters and providers, and distributes business events from disparate sources. ESB is the central integration backbone fulfilling the various integration patterns. Enterprise service bus, being the essential and core infrastructure for application integration, ought to be versatile, adaptive, high-performing and assuring, highly available and scalable. In order to accomplish the much-acclaimed process integration, service orchestration capability has to be bestowed with ESBs. The importance of orchestration as a shared component gives ESBs the flexibility and the power towards the success demanded and desired.

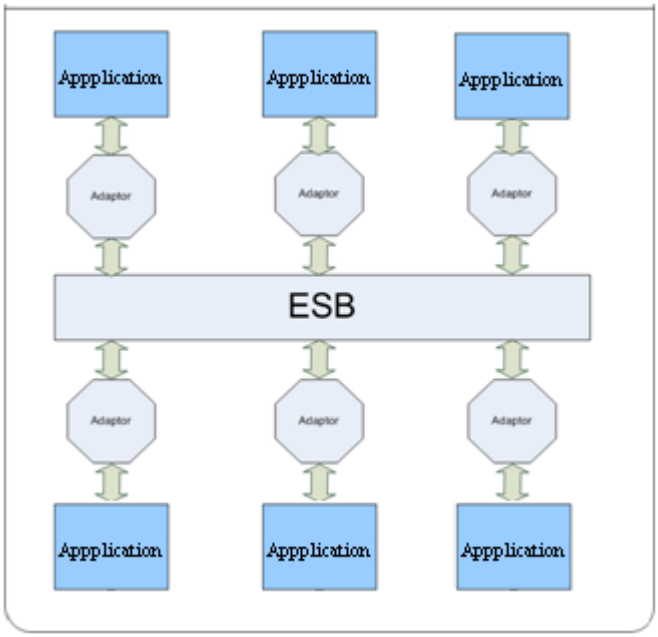

Figure 8. ESB communication

Although EAI solutions can address all of these aspects, integration technologies are usually much more narrowly focused. ESB handles a variety of

interaction patterns, including events. ESB requires management such that the status of a business transaction can be assessed. Although the ESB will not be the only technology to assist in business activity management, it will be a part. ESB product technologies will be federated such that various technologies (e.g., gateways and appliances) can be used to fulfill a single purpose and provide a single interface to applications. Diverse platforms can be supported allowing different ESB technologies to operate as a single logical one. Even if both EAI and ESB can use web services, the latter takes more advantage of the technology and also promotes greater levels of modularity and decoupling of the infrastructure using services. ESBs use registries to assist with locating services, while EAI infrastructures often couple the requester and provider.

From a physical point of view, SOA architecture is very similar to the 3tier web one, the logic is on a server where it is divided into several units. Differences arise from the criteria for sharing the logic, the place where logical units exist and how they interact. In Web architecture there are components that are designed with different levels of functionality and granularity and some emphasis is put on reusing them.

SOA is based on components, modeling takes account of the creation of services that will encapsulate some of the components or all components. The encapsulation of service logic is used to provide interface functions using an open, independent of the technologies used to implement logic. Properly designed, loosely coupled services can easily be combined, aggregated and reused, hence resulting in high scalability SOA solutions. In figure 3 it's presented an endpoint from which 3 different web services can be called, separately or they can be aggregated [17].

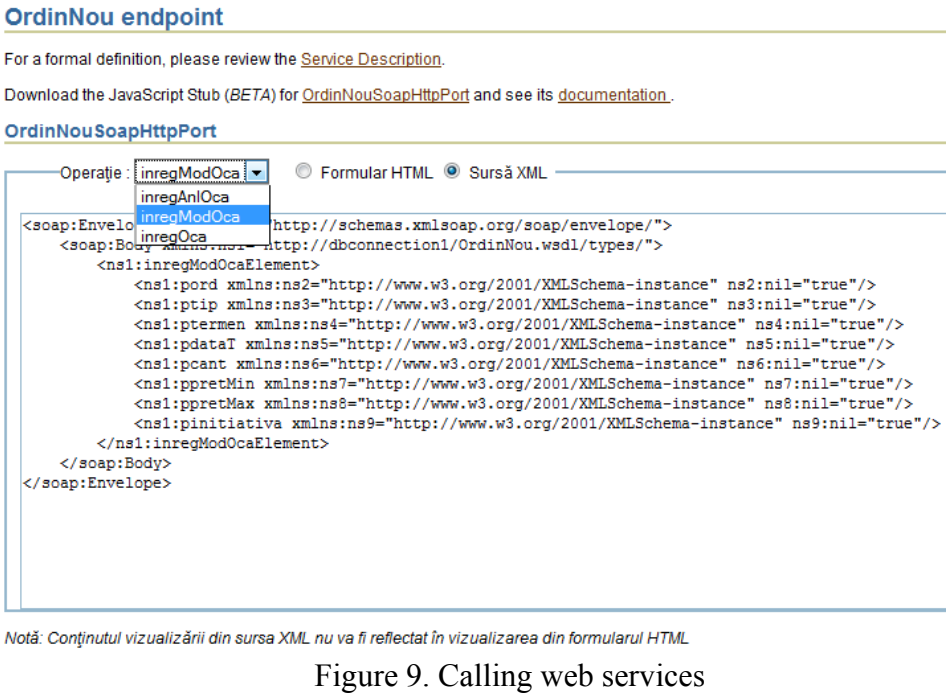

Figure 9. Calling web services

The result of calling a web service, inregAnlOca in this case, is in the form of a XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns0="http://dbconnection1/OrdinNou.wsdl/types/"><env:Body><ns0:inregAnlOcaResponseElement><ns0:result>I-It is confirmed the anulment of the 4328884 order, quantity 500</ns0:result></ns0:inregAnlOcaResponseElement></env:Body></env:Envelope>
```

SOA projects are of different sizes and not all of them require service modeling. SOA projects whose goals include engineering business applications that are built to change service modeling requirements. Some say that new modeling techniques are not required because nothing fundamental is changed. Others say that classic Agile methods or object development methods provide insufficient guidance for SOA projects [1]. Agile methods focus on iterative development, allowing requirements and the solution to grow by collaboration using cross-functional teams including various stakeholders. Object methodologies focus on object modeling and object technologies. In both methods, which represent best practices in system development methods, the focus on service development is absent, mainly because this wasn't popular at the time of these methods development. Organizations developing SOA will need to improve these methods to identify and build reusable, reconfigurable, and flexible services by identification, specification, and realization of these constructs: business processes, services, components, information, rules and policies.

Models help people to deal with complexity by representing complex things at a higher level of abstraction. SOA can help to elevate the level of abstraction by separating the provider from the consumer of the service. The service model identifies the business processes that uses services or identifies the services and the components that recognize the business functionality. Service modeling, as an iterative and business centric process, should focuses on the set of business capabilities and related IT functionality as a set of services, the components that implement them, and the processes that invoke them or put the services together into a composite service or application, should be seen as a whole and address the modeling of activities or flows, services, and their components. Modeling is an iterative and business centric process. A service needs to be modeled from a business and a runtime perspective such that the service fulfills a part of a business process that can be shared and reused.

Like shown in [12], services should be defined and described top-down at enterprise level in Service Oriented Enterprise (SOE). From a functional decomposition of well defined business functionality the business functions can be identified. These business functions can be decomposed in lower level services. As shown in [18], business logic is the defining element for a business being in the process of modeling and automation, and it includes both business rules and workflow (process), which describes the transfer of documents or data from one participant (person or software system) to another. Business Rules refers to the

multitude of policies, procedures or definitions that govern how an organization works together with its interaction with customers or partners. These may be external rules, coming from legal regulations that must be observed by all organizations acting in a certain field, or internal rules which define the organization's business politics and aim to ensure competitive advantages in the market. Starting from the previous observations, it is obvious the important role that business rules play within the development process of a software system. Top-down domain decomposition using process modeling and decomposition, variation-oriented analysis, policy and business rules analysis, and domain specific behavior modeling should be done in parallel with a bottom-up analysis of existing legacy assets that are candidates for componentization (modularization) and service exposure [12].

In SOA, the system operates as a collection of services and each service may interact with various other services to accomplish a certain task. The operation of one service might be a combination of several low level functions. In that case, these low level functions are not considered services.

Most web services are based on document type messages that are designed to be as independent as possible. Using SOAP headers, the actual messages may be accompanied by a wide range of metadata, and general processing instructions.

Processing is distributed; each service has a specific functional border and specific resource requirements. Communication between the supplier and the consumer of services can be synchronous or asynchronous, can use templates and a large part of the business logic is contained in the messages.

Information processing is accomplished at the application server and/or database level. Communication in classical architecture is achieved using protocols such as TCP/IP, DCOM (Distributed Component Object Model) or CORBA (Common Object Request Broker Architecture), protocols which have reached maturity. The first service-oriented architecture for many people in the past was with the use DCOM or Object Request Brokers (ORBs) based on the CORBA specification. Nowadays SOA is based on communications using services which imply: serialization, de-serialization and transmission of SOAP messages containing XML documents. Operations that are executed include: conversion of data from relational databases in XML, XML document validation, document transmission and retrieval of information needed by the recipient. Although progress has been made for SOAP classical communications, Remote Procedure Call (RPC) are faster. A SOAP-based communications network facilitates the creation of services that can communicate according to various templates. Even if the synchronous communications is well implemented, asynchronous communications is encouraged to optimize processing and communications. WS-* specifications can be used, especially WS-BPEL.

The technologies used on the Internet have undergone many changes and improvements but remained basically the same: HTTP, HTML or XML. If in traditional web architectures, web services use XML messages are optional, if modern implementations of SOA are almost mandatory.

When the logic of a system is divided and distributed, the implementation of security measures such as authentication and authorization is not as straightforward as it was in client-server architecture. Information travels through multiple servers (bumps) and it is often necessary at least to encrypt it or at least the sensitive information: password, bank account, etc. SOA brings some changes to this model, being based on the WS-Security which puts the logic of security-

related messages. In SOAP messages, the header may store security related information will be accompany the messages. This approach is necessary to maintain autonomy and loose coupling between services.

Conclusion

Although at first glance SOA administration may seem simple, things usually develop to a point where services are highly aggregated and reused and the administration becomes difficult. Then it is necessary to use stronger service agents or private service agents. When we have an application that includes various components, management is not easy, the following have to be monitored: the connections, the instances, the problems with the connections, the resources employed and the tasks related to database administration. The clients connect at first at first to the web server which interacts with the application server, so it is important to carefully administer it to ensure scalability. Most application servers and database management systems provide mature interfaces that can be accessed with a web browser. In SOA solutions, additional problems may arise with regard to communications using SOAP messages. Management errors can be done using the exception mechanism provided by different WS-* extensions. A good strategy to encourage the reuse and aggregation of an internal solution is to create a private agent service. UDDI can be used to standardize the interface of the agent services and so the system services can be easily discovered.

References

[1] Kerrie Holley, Dr. Ali Arsanjani, *100 SOA Questions Asked and Answered,* Prentice Hall, ISBN 978-0-137-08020-5, 2010

[2] S.Y. Jeong, Y. Xie, J. Beaton, B.A. Myers, J. Stylos, R. Ehret, J. Karstens, A. Efeoglu, and D.K. Busse, *Improving Documentation for eSOA APIs through User Studies*, in Proc. IS-EUD, 2009, pp.86-105.

[3] Thomas Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall PTR, 2005, ISBN: 0-13-185858-0

[4] Marinela Mircea, Marian Stoica, Cloud Computing Solutions For Service Oriented Organizations Management, Proceedings of The Tenth International Conference on Informatics in Economy IE 2011

[5] Mircea, M. ,Andreescu, A.I., *Extending SOA to Cloud Computing in Higher Education*, The 15th IBIMA conference on Knowledge Management and Innovation: A Business Competitive Edge Perspective, Cairo, Egypt 6-7 November 2010. Norristown: International Business Information Management Association

[6] Alin COBÂRZAN, *Consuming Web Services on Mobile Platforms,* Informatica Economică vol. 14, no. 3/2010, ISSN 1453-1305

[7] S. Agarwal, S. Lamparter and R. Studer, *Making Web services tradable: A policy-based approach for specifying preferences on Web service properties*, Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 7, No. 1, January 2009, pp. 11-20.

[8] Vlad DIACONIȚA, *Hybrid Solution for Integrated Trading,* Informatica Economică vol. 14, no. 2/2010, ISSN 1453-1305

[9] Maureen O'Gara, *Oracle Goes to Amazon*, .NET Delopers Journal, Mai 2011

[10] Mario Godinez, Eberhard Hechler, Klaus Koenig, Steve Lockwood, Martin Oberhofer, Michael Schroeck *The Art of Enterprise Information Architecture: A Systems-Based Approach for Unlocking Business Insight,* IBM Press, ISBN: 978-0137035717, aprilie 2010

[11] Vlad Diaconita, Ion Lungu, Adela Bara, *Technical solutions for integrated trading on spot, futures and bonds stock markets (extended version),* WSEAS Transactions on Information Science and

Applications Volume 6, Issue 5, 2009, Pages 798-808 , ISSN: 1790-0832, Indexed by Scopus, ACM
[12] http://www.enterprise-architecture.info/EA_Services-Oriented-Enterprise.htm
[13] Stephen Bennett, Thomas Erl, Clive Gee, Robert Laird, Anne Thomas Manes, Robert Schneider, Leo Shuster, Andre Tost, Chris Venable, *SOA Governance: Governing Shared Services On-Premise & in the Cloud,* Prentice Hall/PearsonPTR, Aprilie 2011
[14] David S. Linthicum, *Cloud Computing and SOA Convergence in Your Enterprise: A Step-by-Step Guide*, Addison-Wesley Professional; 1 edition, October 2009
[15] http://www.servicetechmag.com/, ISSUE June 2011, Editorial
[16] Daryl C. Plummer, David Mitchell Smith, Thomas J. Bittman, David W. Cearley, David J. Cappuccio, Donna Scott, Rakesh Kumar, Bruce Robertson, *Five Refining Attributes of Public and Private Cloud Computing*, May 2009
[17] Pethuru Cheliah, *Empowering the Discipline of Cloud Integration – Part II*, Service Technology Magazine Issue LI, June 17, 2011
[18] Alexandra FLOREA, Anca ANDREESCU, Vlad DIACONITA, Adina UTA, *Using SOA for achieving enterprise interoperability,* Proceedings of The Tenth International Conference on Informatics in Economy IE 2011

Vlad DIACONIȚA is an Assistant Lecturer at the Economic Informatics Department at the Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies of Bucharest. He has graduated the faculty at which he is now teaching in 2005 and since 2010 holds a PhD in the field of Cybernetics and Statistics. He is the co-author of 2 books in the domain of economic informatics, 3 articles in ISI journals, 4 articles in Scopus journals, 4 articles in ISI proceedings, 6 papers in B+ journals and 8 papers in the proceedings of international conferences. He participated as team member in 3 research projects that have been financed from national research programs. He is a member of the IEEE Computer

Society and the INFOREC professional association. Domains of competence: Database systems, Data warehouses, OLAP and Business Intelligence, Integrated Systems, SOA.