TGM - HTBLuVA Wien XX
IT Department

SOA, JSON and REST
Dezsys-Elaboration

Authors: Siegel Hannah & Vogt Andreas

February 28, 2015

Contents

1 Introduction

1.1 Services

In order to understand SOA, an very important concept is a *Service*.
"Services are what you connect together using Web Services. A service is the endpoint of a connection. Also, a service has some type of underlying computer system that supports the connection offered. ",[7]
Furthermore, a Service has to have a clearly defined function and very often they should belong to one business process. It can be seen as an Interface which provides a specific function.

When do we speak of a service?
"A service is also a unit of logic to which service-orientation has been applied to a meaningful extend. It's the application of service-orientation design principles that distingishues a unit of logic as a service compared to other units of logic that may exist solely as objects, components, Web services, REST services or cloud based systems",[2, page 29] These patterns and principles are discussed in section 3.1.

1.2 SOA

Service-Oriented Computing
"Service-oriented computing is an umbrella term that represents a distinct distributed computing platform. As such, it encompasses many things ,including its own design paradigm and design principles, design pattern catalogs, pattern languages, and a distinct architectural model, along with related concepts, technologies, and frameworks.",[2, page 22]
Service-Oriented Architcture
"Service Oriented Architecture is a technology architectural model for service-oriented solutions with distinct characteristics in support of realizing service-orientation", [2, page 27]. Service orientation means, that services of any kind are put into the center of the system, enabling flexible business process (re-)modelling due to a very high business process orientation and loose coupling of the services. bullshit.

"Service-oriented architecture (SOA) is an approach used to create an architecture based upon the use of services. Services (such as RESTful Web services) carry out some small function, such as producing data, validating a customer, or providing simple analytical services.",[10] SOA is not a product or framework, it is a design approach or paradigm for good software design.
SOA has it's underlying business functions provided as Services which can be used by all Application on a shared basis. The applications are using a middleware, for example an ESB, in order to access it's services.

Because SOA is using the technology of WebServices, it is quite platform independent. "It is important to view and position SOA and service-orientation as being neutral to any one technology platform. By doing so, you have the freedom to continually pursue the strategic goals associated with service-orientation computing by leveragng on-going service technology advancements.",cite[page 29]grau
SOA is often used as a newer approach to EAI. [9]

"One of the keys to SOA architecture is that interactions occur with loosely coupled services that operate independently. SOA architecture allows for service reuse, making it unnecessary to start from scratch when upgrades and other modifications are needed. This is a benefit to businesses that seek ways to save time and money.",[10]. The aspect of an ROI is very important within the concept of SOA.

1.3 REST

1.4 JSON

2 Existing Problems

2.1 Historical Overview

With the rise of computers, companies have started to invest into the new information technology during the second half of the 20 century. Mostly, the first thing that has been bought were expensive mainframes, so that some processes could be done with more reliability. Naturally, employees have done some faults and they were not as efficient as computers.
Because of very expensive Hardware and Software which was by far not as evolved as it is today, the implementation of new technologies was quite slow. Computers and Mainframes were mostly used for industries like astronautics or applied researching and eventually even for automating processes such as book keeping.
These implementations were quite easy to implement and there was no need for any intercommunication between services.

In the early 80s, the whole industry changed. Suddenly, the personal computer (PC) made it possible to afford information technology on a large scale. Computers were not only used for difficult arithmetic operations but they started to be an everyday-life tool to improve the work-flow and the business processes in companies.
Many new kinds of technologies, ranging from OS with GUIs to the rise of the World Wide Web, were leading to an really fast expansion of information technology.
But with an higher demand on PCs, the demand for infrastructure rose as well. The need for computer specialist was higher than ever and every company had to invest a lot into their, mostly newly founded, IT department.

Because of an lack of know-how and systems that had the characteristics to change a lot, the implementation of all the services became a big challenge. Enterprises used a variety of customised applications, at least one for every type of service.

ODER: "Historically, many IT projects focused solely on building applications designed specifically to automate business process requirements that were current at theat time. This fulfilles immediate (tactical) needs, bit as mpre pf these single-purpose applications were delivered, it resulted in an IT enterprise filled with islands of logic and data reffered to as application *silos*. As new business requirements would emerge, either new silos were created or integration chan-

nels between silos were established. As yet more business change arose, integration channels had to be augumented, even more silos had to be created, and soon the IT enterprise landscape became convoluted ad increasingly burdensome, expensive, and slow to evolve."[2, page 522]

2.2 Problems

Out of these systems, there are many problems that have eventually emerged.
First of all, systems like that are not really agile. Because changes and new technologies were inevitable, time consuming integrations had to be done. If these have simply not been, legacy systems emergered. These were often not remotly changeable.
Also, new challanges such as cloud computing and a more common globalization of Processes made it harder to stick to the old systems.
Because Applications were always providing some kind of service, but hardly seen like a service, they were not as easy reconfigurable and changeable. All the Applications need to communicate with each other and transmit data,so often a start topology was used. It then changed more and more into a Middleware, which had the benefit of only docking the Application to the Middleware once. Nowadays, mostly a bus system gets used, as described in section 4.1.
Due to compatibility concerns, IT-Infrastructure were often Vendor dependent. Therefore we often talk about an SAP-System, because mostly all the components have been bought from SAP, which decreases the agility and may increase the costs.
Furthermore, before SOA, Applications were not divided into processes, and therefore a BPM was made difficult for both the management or the IT-department.
All these restriction lead to increased overall costs and an reduced ROI.

3 SOA as an solution

"In many ways, service-orientation emerged in response to these problems. It is a paradigm that provides an alternative to project-specific, silo-based, and integrated application development by adamantly prioritizing the attainment of long-term, strategic business goals.",[2, page 522]
The target state of service-orientation is to not have these traditional problems any more. In some cases, due to legacy systems or other problems this is not possible, but still SOA tries to realize it to whatever extend possible.

Service-orientation emerged as a formal method in support of achieving the following goals an benefits associated with service-oriented computing:

- Increased Intrinsic Interoperability
- Increased Federation
- Increased Vendor Diversification Options
- Increased Business and Technology Alignment
- Increased ROI

- Increased Organizational Agility
- Reduced IT Burden

[2, page 23] These strategic goals the are put into more low-level design principles. These goals are especially interesting not only to IT-staff members but also for a organization's management. It is one step above EAI already and it combines important aspects of BPM, OOP and AOP in it.

3.1 Design Principles

Because SOA is only an Design Paradigm and not a concrete implementation, service orientation Patterns and principles are used. The design paradigm consists of the following points:

1. Standardized Service contracts
2. Loosly coupled systems
3. Abstraction of Services
4. Reusability
5. Autonomity
6. Statelessness
7. Discoverability
8. Composability

[2, page 25]

3.1.1 Standardized Service contracts

"A service contract expresses the technical interface of a service. "

3.1.2 Loosly coupled systems

3.1.3 Abstraction of Services

3.1.4 Reusability

3.1.5 Autonomity

3.1.6 Statelessness

3.1.7 Discoverability

3.1.8 Composability

3.2 SOA Manifesto

The SOA Manifesto has been developed 2009, and it is quite similar to the Agile Manifesto, which is widley known. Of the Value-Groups, both values are important and should be achived,

but the left one is always more important.
SOA Manifesto:

"We have been applying service orientation to help organizations consistently deliver sustainable business value, with increased agility and cost effectiveness, in line with changing business needs.

Through our work we have come to prioritize:

- Business value over technical strategy
- Strategic goals over project-specific benefits
- Intrinsic interoperability over custom integration
- Shared services over specific-purpose implementations
- Flexibility over optimization
- Evolutionary refinement over pursuit of initial perfection

" [11]

never done at full extend

List of Tables

List of Figures

6.6 Easy Bibliography

References

[1] **Who**, When
url
last used: dd.mm.yyyy, hh:mm

[2] **Who**, When
url
last used: dd.mm.yyyy, hh:mm

[3] **Who**, When
url
last used: dd.mm.yyyy, hh:mm

[4] **Who**, When
url
last used: dd.mm.yyyy, hh:mm

[5] **Who**, When
url
last used: dd.mm.yyyy, hh:mm

[6] **Who**, When
url
last used: dd.mm.yyyy, hh:mm

[7] **Service**,service-architecture.com
http://www.service-architecture.com/articles/web-services/service.html
last used: 03.02.2015, 13:39

[8] **Service-Oriented Architecture (SOA) Definition**,service-architecture.com
http://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html
last used: 03.02.2015, 13:40

[9] **SOA (service oriented architecture)**,IT Wissen
http://www.itwissen.info/definition/lexikon/service-oriented-architecture-SOA-SOA-Architektur.html
last used: 03.02.2015, 13:50

[10] **service-oriented architecture (SOA) definition**,Margaret Rouse
http://searchsoa.techtarget.com/definition/service-oriented-architecture
last used: 03.02.2015, 14:05

[11] **The SOA Manifesto**
http://www.soa-manifesto.org/default.html
last used: 28.02.2015, 14:05