



TGM - HTBLuVA Wien XX
IT Department

SOA, JSON and REST

Dezsys-Elaboration

Authors: Siegel HANNAH & Vogt ANDREAS

March 9, 2015

Contents

1	Introduction	2
1.1	Services	2
1.2	SOA	2
1.3	REST	3
1.4	JSON	3
2	Existing Problems	3
2.1	Historical Overview	3
2.2	Problems	4
3	SOA as an solution	4
3.1	Design Principles	5
3.1.1	Standardized Service contracts	6
3.1.2	Loosly coupled systems	6
3.1.3	Abstraction of Services	6
3.1.4	Reusability	6
3.1.5	Autonomy	6
3.1.6	Statelessness	6
3.1.7	Discoverability	6
3.1.8	Composability	6
3.2	SOA Manifesto	6
3.3	SOA Lifecycle	7
3.4	BPM	7
3.4.1	BPML	7
3.4.2	BPEL	7
4	JAX-WS	7
5	Implementation	8
5.1	ESB	8
5.2	RestFul Services	8
5.3	Migration of legacy systems	8
5.4	Protocols	8
5.4.1	XML	8
5.4.2	JSON	8
5.4.3	SOAP	8
5.4.4	WSDL	8
5.4.5	UUID	8
6	Code Snippets	8
7	Comparison	8
7.1	Pro and Contra of SOA	8
7.2	SOA VS. EAI	8

7.3 SOA VS. OOP 8

7.4 SOA VS. REST 8

7.5 Practical Context and Real life implementation 8

7.6 Easy Bibliography 9

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. ",[8]

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 distinguishes 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.

A service contains it's clearly defined functionality, a description of this functionality and Basic-operations such as binding, selection, publication or discovery [5, page 8].

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 Architecture

"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.",[11] 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. [10]

”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.”,[11]. The aspect of an ROI is very important within the concept of SOA

1.3 REST

Rest stands for Representational State Transfer. It is a software architecture style consisting of guidelines and best practices for creating scalable web services. The main idea behind REST is that you are working with the HTTP protocol. Rest is basically using HTTP verbs, GET, POST, PUT, DELETE and HEAD, in order to act on resources, represented by individual URIs. A perk of those verbs is that they are mostly self-explanatory.

1.4 JSON

”JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C sharp, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.”,[12].

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 the time. This fulfills immediate (tactical) needs, but as more of these single-purpose applications were delivered, it resulted in an IT enterprise filled with islands of logic and data referred to as application *silos*. As new business requirements would emerge, either new silos were created or integration channels between silos were established. As yet more business change arose, integration channels had to be augmented, even more silos had to be created, and soon the IT enterprise landscape became convoluted and 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 emerged. These were often not remotely changeable.

Also, new challenges 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 star 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 5.1.

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 restrictions lead to increased overall costs and a 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. Autonomy
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. ”, cite[page 33]grau This means that there are interfaces which do contain the metadata to an service.

If these services are implemented as a Webservice, for example, most commonly an description document in Form of an WSDL definition will be used, or an XML Schema.

Furthermore, A service contract will often be concluded in Human Readable documents. These are called Service Level Agreements (SLAs) and they often contain information like quality requirements or overall business information.

3.1.2 Loosly coupled systems

3.1.3 Abstraction of Services

3.1.4 Reusability

3.1.5 Autonomy

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 widely known. Of the Value-Groups, both values are important and should be achieved, 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

” [13]

3.3 SOA Lifecycle

3.4 BPM

3.4.1 BPML

3.4.2 BPEL

”BPEL (Business Process Execution Language) is an XML-based language that allows Web services in a service-oriented architecture to interconnect and share data.”,[14]. When using Webservices, it also is often called WSBPEL. ”Programmers use BPEL to define how a business process that involves web services will be executed. BPEL messages are typically used to invoke remote services, orchestrate process execution and manage events and exceptions.

BPEL is often associated with Business Process Management Notation (BPMN), a standard for representing business processes graphically. In many organizations, analysts use BPMN to visualize business processes and developers transform the visualizations to BPEL for execution. BPEL was standardized by OASIS in 2004 after collaborative efforts to create the language by Microsoft, IBM and other companies.”,[14].

It therefore helps to produce code out of a graphical description, and there is no need to code it all out once again. Such systems should interact with other systems such as ERP-systems and without SOA it would be hard to realize. Using BPEL, smaller parts of an application can be easily combined to a greater one. [5, page 18]

4 JAX-WS

SOAP-based Web services support in Java EE 5,6 and 7 is based on the Java API site 112
grey book

5 Implementation

5.1 ESB

5.2 RestFul Services

5.3 Migration of legacy systems

5.4 Protocols

5.4.1 XML

5.4.2 JSON

5.4.3 SOAP

5.4.4 WSDL

5.4.5 UUID

6 Code Snippets

7 Comparison

7.1 Pro and Contra of SOA

7.2 SOA VS. EAI

7.3 SOA VS. OOP

7.4 SOA VS. REST

7.5 Practical Context and Real life implementation

never done at full extend

List of Tables

List of Figures

7.6 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] **Who**, When
url
 last used: dd.mm.yyyy, hh:mm
- [8] **Service**,service-architecture.com
<http://www.service-architecture.com/articles/web-services/service.html>
 last used: 03.02.2015, 13:39
- [9] **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
- [10] **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
- [11] **service-oriented architecture (SOA) definition**,Margaret Rouse
<http://searchsoa.techtarget.com/definition/service-oriented-architecture>
 last used: 03.02.2015, 14:05

- [12] **JSON**,json.com *<http://www.json.org/>*
last used: 02.02.2015, 22:15
- [13] **The SOA Manifesto**
<http://www.soa-manifesto.org/default.html>
last used: 28.02.2015, 14:05
- [14] **BPEL (Business Process Execution Language) definition**,Margaret Rouse
<http://searchsoa.techtarget.com/definition/BPEL>
last used: 08.03.2015, 13:21