



TGM - HTBLuVA Wien XX
IT Department

SOA, JSON and REST

Dezsys-Elaboration

Authors: Siegel HANNAH & Vogt ANDREAS

March 10, 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	5
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	Service Re-usability	7
3.1.5	Service Autonomy	7
3.1.6	Statelessness	7
3.1.7	Discoverability	8
3.1.8	Service Composability	8
3.2	SOA Manifesto	8
3.3	SOA Lifecycle	8
3.4	BPM	8
3.4.1	Challenges of BPM [19]	9
3.4.2	BPML	9
3.4.3	BPEL	9
4	SOA Model if we need it	9
5	JAX-WS	10
6	Implementation	11
6.1	ESB	11
6.2	RestFul Web Services	11
6.3	Migration of legacy systems	11
6.4	Protocols	12
6.4.1	XML	12
6.4.2	JSON	12
6.4.3	SOAP	13
6.4.4	WSDL	13
6.4.5	UUID	13
7	Code Snippets	13

8	Comparison	14
8.1	Pro and Contra of SOA	14
8.2	SOA VS. EAI	14
8.3	SOA VS. OOP	15
8.4	SOA VS. REST	15
9	Conclusion	15
10	If we need it!!	15
10.1	Implementation Topologies	15
10.1.1	Peer-to-peer	15
10.1.2	Hub & Spoke	15
10.1.3	Bus	15
10.2	Easy Bibliography	16

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].

Seen out of an more business related approach, they usually provide clear incident, problem, change, configuration, release, availability and cost management - which helps to gain information and overview the services.

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 leveraging 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 (Uniform Resource Identifiers). A perk of those verbs is that they are mostly self-explanatory. REST is a lightweight alternative to mechanisms like RPC (Remote Procedure Calls) and Web Services (SOAP, WSDL, et al.)

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 6.1.

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.

The two approaches nowadays are either Top-Down (designing the service contracts at first and then implementing them) or Bottom-up(Building the contracts out of already existing services). [2, page 151,152]

REST

”For REST services, capturing and communicating various aspects of resources can be necessary, such as the set of resources, relationships between resources, HTTP verbs allowed on resources and supported resource representation formats. Standards such as WADL(Web Application Description Language), can be used to satisfy the mandatory requirements. [...] Even the self-describing contract of HTTP verbs for a REST service establishes a standard-based service contract.”, cite[page 151]grau

SLAs

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.

Generation Possibilities within Java

”JAX-WS defines the wsimport tool, which takes an existing WSDL definition as input to generate Java skeletons [...] Similarly, the wsgen tool generates WSDL from existing Java code.”,[2, page 151]

3.1.2 Loosly coupled systems

Loose coupling is a well known design principle. Basically, it means that services or applications are as independently of each other as possible.

”Loose coupling is an approach to interconnecting the components in a system or network so that those components, also called elements, depend on each other to the least extent practicable. Coupling refers to the degree of direct knowledge that one element has of another.”, [24]

3.1.3 Abstraction of Services

”The appropriate level of abstraction at which services are described achieves additional agility and alignment between business and IT areas of an enterprise. Abstracting information means taking technical details out of a problem to be solved on a higher level” [2, page 184]

For example, abstraction would be an abstraction into an API.

”Service-orientation continues the evolution of higher-level abstraction use to make creating and changing solutions easier.” [2, page 184]

3.1.4 Service Re-usability

The following are common design characteristics associated with reusable services:

- The service is defined by an agnositic functional context
This means that an agnostic functional context is not specific to any one purpose and is therefore considered multi-purpose. It means, that it is idependent of the particular business process or the area of application.
- The service logic is highly generic
- The service can be used concurrently

In general, reusability is a very important design principle which gets used in normal OOP as well. [2, page 140-150]

3.1.5 Service Autonomy

”Service-orientation revolves around building flexible systems. Flexibility is, to a large degree, archieved through making services decoupled and autonomous to enable them to be composed, aggregated and changed without affecting other parts of the system. For a service to be autonomous, the service must be as independent from other services with which it interacts, both functionally and from a runtime environment perspective. Java and Java EE provide a highly efficient runtime environment that supports service composition. For example, a Java EE application server support concurrent access to its hosted components, making each access to such a component autonomous from the others.”, [2, page 194]

3.1.6 Service Statelessness

”Each invocation of a service operation is completely independent from any other invocation, whether by the same consumer or any other service consumer. The service Statelessness principle offers various benefits centered around improved scalability by which additional stateless service instances can be easily provisioned on available environments.”, [2, page 197]

”Many real-life business scenarios can be expressed as business processes that include automated steps, which can require manual intervention. Designing and implementing such a process requires some state to be maintained for the duration of the process. Executing an instance of the process definition forms the notation of a session or transaction across multiple service invocations. Therefore, a service implementing the execution of a business process cannot be stateless and may need to even maintain contex information over extended periods”. [2, page 197]

3.1.7 Service Discoverability

”The two primary aspects of the Service Discoverability principle are discovery at design-time (which promotes service reuse in a newly developed solution), and discovery at runtime (which involves resolving the appropriate endpoint address for a given service or retrieving other meta-data).”, [2, page 204]

Design-Time Discoverability

”At design-time, it is important for project teams to be able to effectively identify the existence of services that contain logic relevant to the solution they plan to build. This way they can either discover services that they can reuse or confirm that new service logic they plan to build does not already exist. For example, a service is designed to address an Order Management business process for which customer information is required. The service designer must investigate whether a **Customer** data type already exists and if so determine whether it meets the requirements for the Order Management Process service.” [2, page 204-205]

Run-Time Discoverability

”Runtime service discovery refers to the ability of software to programmatically search for services using APIs exposed by the service registry. Doing so allows for the retrieval of the physical location or address of services on the network. Because services may need to be moved from one machine to another or perhaps redundantly deployed on multiple machines, it may be advisable for service addresses not to be hardcoded onto the service consumer logic. [...] In addition to the lookup of a service endpoint address, other artifacts can be used by service consumers at runtime to identify an appropriate service and build the applicable request message. The JAX-WS Dispatch API allows a service request to be completely built at runtime.” [2, page 205-207]

3.1.8 Service Composability

”The composability of a service is an implicit byproduct of the extent to which the other service-orientation design principles have been successfully applied. [...] For example, a service contract that is standardized allows interaction and composition between services implemented in different languages using different runtime environments.” [2, page 189]

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 archived, but the left one is always more important.

The wording of the 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

”Business Process Management is a systematic approach to making an organisation’s workflow more effective, more efficient and more capable of adapting to an ever-changing environment. A business process is an activity or an set of activities that will accomplish a specific organisational goal. The goal of BPM is to reduce human error and miscommunication and focus stakeholders on the requirements of their roles. BPM is a subset of infrastructure management, an administrative area concerned with maintaining and optimizing an organization’s equipment and core operations.”, [18].

Figure 1: Business Management cycle [20]

3.4.1 Challenges of BPM [19]

- Lack of Business Case
- Too Slow to React to the Business Change
- Gap between the modelling and the implementation phases
- Seen as a One Time Project
- Not Investing in Staff
- Poorly Defined Measures of Success

3.4.2 BPML

3.4.3 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 SOA Model if we need it

The SOA-model has different layers. On the top level there is the Presentation Layer, which is the User Interface, which can be a normal Client Application as we know it but also a Fat Client. Then we speak of the Orchestration Layer, Where the flow of an Application is being a sequenz of certain services. In the Orchestration layer, a data transfer between these Services is possible as well. And then there comes the service layer, in which the Services are. In between there is some kind of integration Architecture, which would normally be an ESB. In the application layer, applications that already exists and Databases or systems are put. In the Visualized Infrastructure, the Hardware can be found. These layers can be seen in figure ??.

Orchestration

Orchestration means, that a new Service can be build out of other Services. These other services can be of any kind (also external) and the communication between them is usually koordinated out from an central controlling instance. Orchestration is usually done using (WS-)BPEL as described closer in section 3.4.3. This layer is also a reason why SOA is so flexible and why the systems must be loosely coupled. [5, page 29]

SOA models There exist different SOA models, such as the OASIS model, The SOA Meta model of the W3C and each of the bigger companies such as IBM, SAP or Oracle have their own interpretation.

5 JAX-WS

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

6 Implementation

6.1 ESB

6.2 RestFul Web Services

RESTful web services are built to work best on the Web. RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations. If you want to access resources like data and functionality in the REST architectural style you have to use Uniform Resource Identifiers which are typically links on the Web. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol. // The following principles encourage RESTful applications to be simple, lightweight, and fast:

- Resource identification through URI: A RESTful web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery.
- Uniform interface: Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource.
- Self-descriptive messages: Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others. Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.
- Stateful interactions through hyperlinks: Every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields. State can be embedded in response messages to point to valid future states of the interaction.

[16]

6.3 Migration of legacy systems

”Due to the frequency of change in business requirements and evolution in technology, the need to evolve existing software systems is ever increasing. This results in demand for new methods to support this process, in particular where the transition towards modern architectures is

concerned. SOAs are steadily becoming mainstream software engineering practice. Reports show that more than 50 percent of large, newly developed applications and business processes designed during the year 2007 used service-oriented architectures to some extent.”,[15].

6.4 Protocols

6.4.1 XML

XML stands for EXtensible Markup Language. XML was designed to describe data. XML is a software- and hardware-independent tool for carrying information.

Besides HTML and XHTML the most widespread RESTful HTTP-environment.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<orders xmlns="http://example.com/schemas/ordermanagment">
  <order href="/orders/442820205">
    <customer>
      ...
    <customer>
  </order>
</orders>
```

Buch REST und HTTP Seite 83

6.4.2 JSON

The focus lies, if compared to XML, in datastructure and not text. Does not support namespaces and schemed based validation but is easier to understand than XML.

Example:

```
{ "widget": {
  "debug": "on",
  "window": {
    "title": "Sample Konfabulator Widget",
    "name": "main_window",
    "width": 500,
    "height": 500
  },
  "image": {
    "src": "Images/Sun.png",
    "name": "sun1",
    "hOffset": 250,
    "vOffset": 250,
    "alignment": "center"
  }
},
```

```

    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "hOffset": 250,
        "vOffset": 100,
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}

```

todo better Example

6.4.3 SOAP

SOAP was originally a shortcut for "Simple Object Access Protocol", but since it isn't simple and it isn't used to access object only the protocol was correct. Since version 1.2 the shortcut was abolished and SOAP now stands for itself. todoExample + what is does SOA in der Praxis S 268

6.4.4 WSDL

Web Service Description Language is a XML based Interface communicationinterface description language which is used to describe web services which are called by SOAP-Messages. Currently there are two Versions:

- 1.1 : Which is used Worldwide and is supported by nearly every tool
- 2.0 : Brings some improvments but they are not relevant in the praxis

todo SOA in der Praxis S 263 Example

6.4.5 UUID

"A UUID (Universal Unique Identifier) is a 128-bit number used to uniquely identify some object or entity on the Internet. Depending on the specific mechanisms used, a UUID is either guaranteed to be different or is, at least, extremely likely to be different from any other UUID generated until 3400 A.D.A guaranteed UUID contains a reference to the network address of the host that generated the UUID, a timestamp (a record of the precise time of a transaction), and a randomly generated component.A UUID is specified as part of the tModel data structure, which represents a service type (a generic representation of a registered service) in the UDDI (Universal Description, Discovery, and Integration) registry. This mechanism is used to discover Web services." [17]

7 Code Snippets

Rest Servlet

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:xsi="http://www
  xmlns:p="http://www.springframework.org/schema/p"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-4.0.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">
  <context:component-scan base-package="at.xion.watcher" />
  <mvc:annotation-driven />
</beans>

```

Web XML

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://
  <display-name>WatcherTomcat</display-name>
  <servlet>
    <servlet-name>rest</servlet-name>
    <servlet-class> org.springframework.web.servlet.DispatcherServlet</servlet-
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>rest</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>

```

REST JAVA

```

@RequestMapping("/addToken")
    public String addToken(@RequestParam(value = "token") String token, @Reques

@RequestMapping("/removeToken")
    public void removeToken(@RequestParam(value = "token") String token)

```

8 Comparison

8.1 Pro and Contra of SOA

8.2 SOA VS. EAI

"Differently than EAI, which deals with linking enterprise applications so they can communicate with one another (by means of an intelligent reasoning engine) and carry out 'batch' data

transfers, is the service oriented architecture (SOA) that provides 'transactional' data transfers, with no third-party software required. SOA is different from the EAI approach in that it does not depend on a third-party solution.” ,[21]

8.3 SOA VS. OOP

8.4 SOA VS. REST

9 Conclusion

Whereas SOA can not be seen as an implementation but more as an design principle, it comes with many advantages. Using the Web-oriented approach and also putting Services and Business Processes into the focus, it leaves an very flexible IT-Infrastructure and leads to an cost efficient way of operating.

Nowadays, many of SOA's principles are already used in what we declare as good software design, but still many companies have problems with applying the SOA Principles at a full extend. Often only parts of it get used, if even. This is mainly because changing an infrastructure is not that easy.

10 If we need it!!

10.1 Implementation Topologies

10.1.1 Peer-to-peer

The peer-to-peer (Figure ??) implementation is not very useful. It is quite outdated and it is not really usable for modern systems. There is practically no way to achieve any scalability and it gets complicated very soon.

10.1.2 Hub & Spoke

Messages get directed to an centralized unit, where they can be processed. (Figure ??) This centralized unit can be a bottleneck though.

10.1.3 Bus

Every service is connected to an service bus (ESB).(Figure ??) The processing distribution in this case is very good. The bus-topology is therefore the most recommended one, because there is no bottleneck when comparing it to a Hub.

List of Tables

List of Figures

1	Business Management cycle [20]	9
---	--	---

10.2 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
- [15] **Migrating Legacy System to Service-Oriented Architecture** ,Carlos Matos
<http://kobv.de/opus4>
last used : 09.03.2015, 21:50
- [16] **What Are RESTful Web Services**,Oracle <http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>
last used : 09.03.2015, 22:30
- [17] **UUID (Universal Unique Identifier) definition**,Margaret Rouse
<http://searchsoa.techtarget.com/definition/UUID>
last used : 09.03.2015, 23:55
- [18] **Business Process Management (BPM)**, Margaret Rouse
<http://searchcio.techtarget.com/definition/business-process-management>
last used : 06.10.2014, 22:13
- [19] **Top Ten Biggest Challenges to BPM Initiatives**, Pearl Zhu
<http://futureofcio.blogspot.co.at/2013/03/top-ten-biggest-challenges-to-bpm.html>
last used : 06.10.2014, 22:32
- [20] **What is Business Process Management?**
<http://www.scc-co.com/enterprises-processes/>
last used : 06.10.2014, 22:04
- [21] **Key differnces between ESB, EAI and SOA**
www.innovativearchitects.com/KnowledgeCenter/Business%20Connectivity%20and%20Interoperability
last used : 06.10.2014, 22:18
- [22] **Torsten Horn**
<http://www.torsten-horn.de/techdocs/eai.htm>
last used : 05.10.2014, 20:25

- [23] **SOA with Java**, Thomas Erl
ISBN: 0-13-385903-4
- [24] **loose coupling definition**, Margaret Rouse
<http://searchnetworking.techtarget.com/definition/loose-coupling>
last used : 10.02.2015, 20:18