Ausarbeitung - AIC

Questions taken from VoWi. Questions in German but answers in English – since the lecture is in English ☺

WebServices und CORBA and Hand von mindestens 4 Dimensionen vergleichen

	WebService	**Corba**
Coupling	Loosely coupled	Very tight coupled
Parameter passing	Value only	Reference and Value
Service Discovery	UDDI or alternatives	Naming service
Location/Addressing	URL	Object reference
Data model	SOAP message exchange	Object Model

3 Eigenschaften von Services allgemein + Unterschied/Gemeinsamkeiten von Software Services und Real World Services

Generally - what is a service: A service is a software application that provides some, usually more complex, functionality to a variety of clients. A service may be used by different applications, either by other services or by mobile applications. **A service is published -> discovered -> invoked.**

A Service generally:

Is coarse grained: it usually receives more complex parameters (e.g. XML file) to process – not like a method in java that only receives some parameters.

Is self contained: No dependencies to other services

Has a standardized interface.

Is easy to use – little integration needed.

Is context independent.

SW Services (SWS) and Real World Services (RWS):

SWS should extend/implement RWS especially in a business process (I don't really understand this ******* slides).

* 5 Vorteile, die (theoretisch) durch die Benutzung von WebServices eintreten sollen benennen und erklären. Mind. 2 kommerziell und mind. 2 technisch.

Technical benefits:

More reuse: Due to the loose coupling and the usage of standards it is very easy to reuse WebServices in a variety of other applications.

Easy maintenance: Since other applications do not know about the actual implementation, the Service can be modified without any problems for the client – just the interface needs to stay the same.

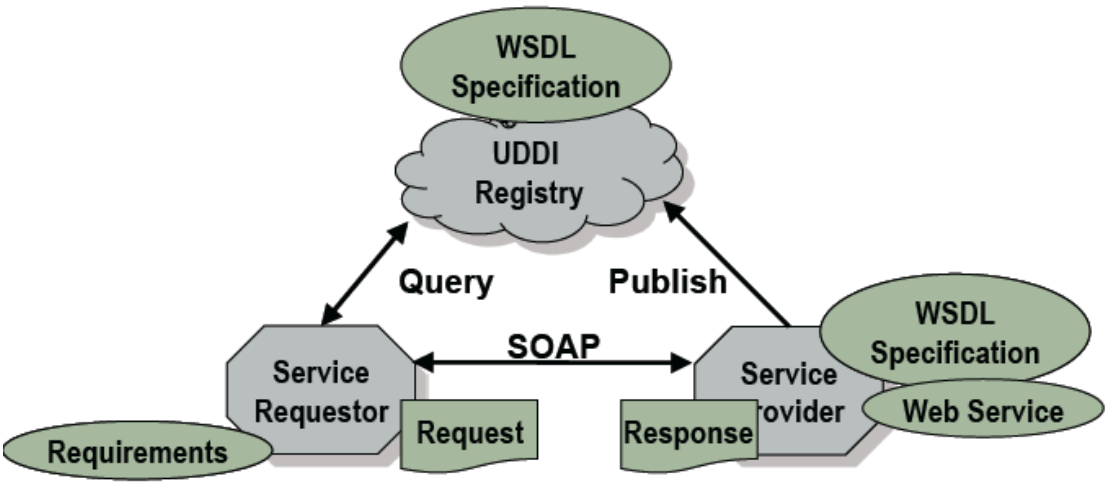
Business benefits:

Increased business agility: Finding the right service is easier, if one isn't Satisfied with a service/provider it is easy to change the service. Easier to include new functionality into own products/applications.

Low integration costs: Integrating Services is easy -> costs are low.

Technology/Vendor dependency is reduced.

Die drei grundlegenden Technologien von WebServices und deren Interaktionen erklären. (SOAP, WSDL, UDDI)



SOAP: A XML based messaging Protocol used for communication between client and provider.

WSDL (Web Service Description Language): XML based language to describe a web service. Each WS Description consists of two parts – the abstract and the concrete part.

UDDI: A registry service where service providers can register their services. Service clients then query the UDDI Registry, search for a suitable service and invoke the service.

Services are described using WSDL, accessed through SOAP and

Was ist BPEL und dessen Grundzüge erklären. Besonderes Augenmerk auf exception handling.

Business Process Execution Language for Web Services.
Through BPEL larger web services can be created from smaller ones. BPEL is basically just a language that groups existing web services together in order to model a business process. BPEL is mostly intended for Orchestration but with some support for Choreography. BPEL defines:

- Activities
- Messages
- Partners
- Data
- Fault handling

Exception Handling: Each Activity exists inside a scope – such a scope is a behavioral context (probably like in python or java and any other decent programming language) – variables existing in a scope are visible only inside this scope (as I said- like in python). Additionally a scope contains several handlers (fault handlers, event handlers). Exception -> try-catch-throw semantics. Per Scope several fault handlers can be defined.

```
Choreography vs. Orchestration
```

Orchestration: Everything is guided by a central instance that guides the execution. In the case of modeling a business process via web services, the business process itself is the central instance.

Choreography: Every web service knows exactly what to do – not central instance or the like is necessary. Used for business collaboration.

```
Message Correlation bei WS-BPEL erklären
```

Correlation = Mapping messages to specific instances. It is of course possible to create more instances of a BPEL service (just one instance would be quite useless). Since the composed services remain the same, it is necessary to map the messages, sent to the services, to a specific instance of the BPEL service. This is done with message correlation. An correlation set defines which part of a message is used for correlation e.g. first and second name could be used for correlation. A correlation set can even be associated to a pair of messages -> this pair then consist of answer and response.

```
WS-CDL erklären, dabei besonders auf Unterschiede zu WS-
BPEL eingehen
```

WS-CDL (Web Service – Choreography Description Language) is a declarative language to define interaction patters i.e. it specifies the interaction between

web services in B2B scenarios if no central authority can be defined. Description for me: If two companies work together in some way, it is hard to define a central authority. The two business partners are still somehow competitors and none of them want's the other one to be the central authority. Of course it would be possible to get a third party in, but that would be too expensive for both ☺

WS-CDL	**WS-BPEL**
Intended for choreography – designed to complement WS-BPEL	Intended for orchestration – limited choreography support.
Not Executable (just declarative)	Executable
Global View (companies working together).	Since it is focused on orchestration – BPEL models business processes as seen from the outside. View on one entity (one process in one company)

Die 5 Merkmale von REST benennen und erklären.

1) Resource drive: Rest is completely resource drive – all operations are focused on resources. There are no special Activities/Methods to e.g. calculate the sum of two stored values. Both values have to be retrieved, summed up and the result has to be stored.
2) Rest is stateless: Each request is self-contained.
3) Naming: Every resource is accessible via a defined unique name.
4) Layering: It is easy to add a new layer e.g. a proxy. Since Rest consists only of HTTP requests it is of course possible to insert all kinds of additional "layers" like firewalls, proxies, load balancer, … .
5) Uniform interface: Every resource is accessed via the same interface.

2 Technologien um REST-Services zu beschreiben aufzählen und beschreiben

To be honest – it is not really necessary to describe a Rest Service since the HATEOAS principle says that knowing one URL should be enough to discover the whole rest webservice. Since there is no real machine readable format to discover a rest service we need some description stuff. Of course hardly anyone uses this description mechanisms.

WSDL 2.0 – The second Version of WSDL allows the description of Rest Services. This description is done in the same way a SOAP Service is described – therefore a unified description of SOAP and REST Web services is possible. There is even some tool support. BUT WSDL 2.0 has very limited expressiveness for Rest Services and the description is not very REST like (there are still operations and messages used).

WADL: The Web Application Description Language is used especially to describe

REST services. It is XML based but **simpler** then WSDL 2.0. It defines resources and not operations (**very rest like**). WADL has 4 basic constructs: Grammar, Resource, Methods, Representation.
The is just one real tool for WADL (sun jersey) and hardly anyone makes use of WADL.

`Unterschied/Gemeinsamkeiten von Service Composition und Mashup`

Mashup describes the integration of several services to one new service that should be more then just the sum of both. Mashup focuses very much on existing data and how to make this data more useful e.g. support weather data from mountains with geographical data to develop a service that warns people currently climbing a mountain. Mashups are usually "easy" to build and easy to use. Mashup uses of course data driven communication – mostly REST.

With composition usually business processes are implemented and mostly SOAP services are used. The slides even say that composited services are mostly intended for many users while mashups are intended only for a few users.

Btw: The development process of mashups is often adh-hoc and the one of composition mostly structured/planned.
Recap: Composition = boring business logic, mashup = might be fun.

`SCA beschreiben, dabei besonders auf Components, Composites und Services eingehen`

SCA = Service Component Architecture
Is a methodology to describe components and how these components interact with each other. It describes a model on how to compose SOA applications. Therefore SOA is the idea/topic and SCA a possibility to implement a concrete SOA. A Web service is a building block of a SCA, but to create an SCA not only web services may be used.

Components: The concrete implementation of an application. Components offer their functions as services.

Composites: A composite is the unit of deployment of an SCA and holds services. It contains on ore more components.

Services: Services are abstractions from a concrete implementation.

UDDI erklären - wie funktioniert das Dynamic Binding von UDDI

UDDI (Universal Description, Discovery and Integration)
UDDI is a universal and flexible discovery service for Web services. A Web service can be registered in UDDI and clients can search the UDDI for suitable services.

Dynamic binding:
A Service's public interface is published as a tModel in a UDDI. A developer implements his application with this interface. At execution time the application queries the UDDI for die tModel key and then gets information to actually bin to the service. The service of course uses the published interface.

Die 5 Alternativen von UDDI, die in der Vorlesung vorgestellt wurden benennen und erklären + Beispiel.

1) Service Portals: Classical website where one can "register" a service. People can search the Website manually for proper services. The Problem is that there are no standards to create such a portal therefore it is hard to parse such portals automatically.xmethods
2) Service Search Engine: Like the service portal, just that the search engine actively crawls the web for services and lists them. Example: seekda
3) P2P Service Register. Like a classical service register, just that it is distributed over all member. Classical P2P concept – information about available services are shared across all users.
4) Dynamic registers like VRESCo. Like UDDI but with a lot more features like QoS, dynamic binding, richer metadata, …
5) Manual/direct exchange of service information.