Web Services

Sanlig Badral
236 828

Betreut von Dipl.-Inform. Thomas Haase

Zusammenfassung

Web Services are a key component of the enterprise integration technology. In this seminar work will be explained what Web Services are and how they work including main base technologies (that are XML, WSDL, SOAP, UDDI) and their importance in the e-commerce environment (known as business world). I will also explain in brief case what are Web Services standards, potential, problems, current and future state. Main focus of this writing is the integration of Web Services. In this work will be given an overview about Web Services technology.

Inhaltsverzeichnis

1 Introduction

The Internet is a long-reaching web of networks and a very big repository of the information, that are increasing all the time. Thus it has become necessary to automat in many areas of the Internet. i.e. It is a purpose to communicate applications with each other, without by hand regulated data exchange. Web Services are a new way to implement networked and distributed applications.
In this seminar work I will explain what Web Services are and how they work including main base technologies (that are XML, WSDL, SOAP, UDDI) and their importance in the e-commerce environment (known as business world). I will also explain in brief case what are Web Services standards, potential, problems, current and future state. I will mainly focus on integration of Web Services. Therefore I will implement a small example Web Services in programming language PHP. The example name is „City distance service“. User can enter 2 city names and get the distance as result.

This work is organized in three main parts that are fundamentals of Web Services, integration of Web Services and summary. In this work will be given an overview about Web Services technology.

1.1 Overview

I will give Web Services overviews and inquire the realization of Web Services purposes with this work. Web Services include many other technologies such as XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), UDDI (Universal Description, Discovery and Integration) etc. I would therefore explain all technologies as shortly as possible, which are used for Web Services.
„Web Services make software functionality and data available over the Internet."
„Web Services allow you to encapsulate old applications standardized, consistent, and reusable.“
„Applications can communicate with each other independent of platforms, programming languages or protocols.“[2]
The aim of this work is an examination of the realization of the above goals.

1.2 Construction of work

I have organized my writing in three parts.

- Fundamentals of Web Services
- Integration of Web Services
- Summary

Fundamentals of Web Services: In this chapter I will introduce following terms.

Web Services definition
Web Services are a term of wide comprehension thus it is not so easy to define. There are many miscellaneous definitions.

Illustration of Web Services architecture
I will observe the architecture for all below technologies. Here will be defined how the Web Services work and their interactions.

What are XML-RPC, SOAP?
These are the main protocols, which are used for Web Services. Certainly these play very important use for the Web Services communication.

What is WSDL and why it used?
WSDL is the abbreviation of Web Services Description Language. Its goal is to define the services understandable by communication protocols. „WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME. [`http://www.w3.org/TR/wsdl/`]“

What is UDDI and how it works?
In this section will be introduced UDDI term. UDDI is the standard of Web Services publication.

Web Services standards
Web Services are still in the early development stage. „We have not created multiple standards, but rather we have gone out and supported interoperability“, said Philip DesAutels, Microsoft's product manager for XML Web Services. It is safe to assume the goal is to have one standard. The Web Services standards are mentioned here briefly.

Integration of Web Services: It is the main part of this writing.

Service creating
You can experience here how do create or use Web Services. What is interesting? What do we need to create or to consume a Web Service? What do we need to communicate with different Platforms and programming languages?

A simple implementation of SOAP
I will explain the main procedure of programming Web Service with SOAP. I will implement here a very simple Web Service.
Service describing
Here will be defined how will be used WSDL document to describe a Web Service and described what ßelf-describingmeans.
Service discovery
How can a developer publish a Web Services? How can a consumer discover the Web Services to use? In this section will be mentioned the Usage of UDDI.
How to integrate two or more various applications?
The topic explains us for all. I will try to describe Web Services integrations techniques and characteristics.
XML application interface
It looms large for loosely coupling of old applications and encapsulation of the applications and their data. So I will briefly repeat about this interface.
Securing Web Services
Web Services security is today very critical topic. Why? So, I will mention about the security of Web Services very briefly.
Web Services problems
What are the problems? How can be solved or more efficiently solved these?

Summary: The advantages and future/potential of Web Services go here.

2 Fundamentals of Web Services

In this part we will concern oneself in fundamentals of Web Services and their basis technologies. Web Services are based on the SOAP, XML, WSDL and UDDI and it can be described a protocol stack, as we see, each aspect is characterized by one or more protocols defined on top of the lower layers. The layers described in figure 1 take over the following tasks. The transport layer used as a base. Here the used transport protocols and the most common one being HTTP are defined. The higher layer is responsible to format and package the information to be exchanged. This layer is represented by SOAP. The functionality of the service is defined in the next layer (description). UDDI is a layer on top of standards-based technologies such as TCP/IP, HTTP, XML, and SOAP to make up a uniform service description format and service discovery protocol.

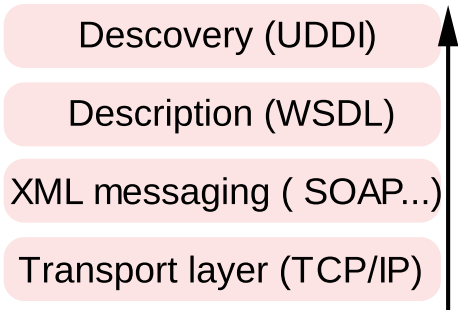

Abbildung 1: Service interaction stack

2.1 Web Services definition

There are many different definitions of Web Services. Two of the most common definitions are given below:

> „A Web Service is any service that is available over the Internet, uses a standardized XML messaging system and is not tied to any one operating system or programming language [1].“

> „A Web Service is seen as an application accessible to other applications over the Web [4].“

However, despite the commonness of the two definitions, they are still too rough for practical usage. In a broader sense of the word, anything that has an URL can be considered as a Web Services.

UDDI consortium [6] provides us with a more precise definition.

> „Web Services are self-contained, modular business applications that have open, Internet-oriented, standards-based interface.“

Though this definition has narrowed down our scope, it is still not precise enough. For instance: what are the open modular business applications and self-contained is not clearly defined. An alternate definition has also been provided by the W3 consortium.[4] This definition is excellent in its clarity, and is as given below:

> „A Web Service is a software system identified by an URL, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web Services in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.“

This definition allows us to have an excellent understanding of Web Services.

From the above definitions, we can now derive a clearer perspective of the term Web Services, and this definition can be given as follows:

„A Web Service is a communicating mechanism with standardized interface of XML messaging, integrates loosely coupled and distributed applications via the Internet independent from platforms, programming languages and operations systems.“

XML messaging uses XML as the data standard format. XML is an easy extensible powerful markup language that enables users to create their own vocabularies. This makes increment of interoperability of application services. XML is very popular and widely used today. The use of Web Services enables application-to-application communication. The particular notable properties of Web Services are self-describing and discoverable.

2.2 Architecture of Web Services life cycle

The lifecycle architecture shown in figure2 compendiously summarizes client and server side and illustrates the interactions between them with business operations in the real world. The lifecycle covers three primary entities, which are Service Provider, Service Registry, and Service Requester. Also this is a complete B2B Web Services architecture model.

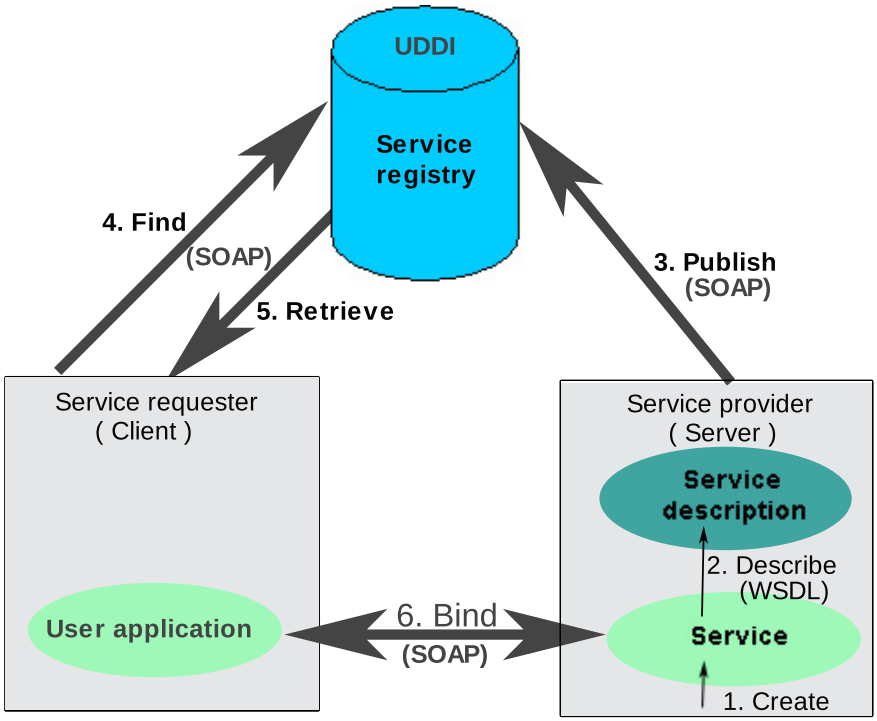

Abbildung 2: B2B Web Services architecture model

Web Services are independent from programming languages, so any programming language can be used by the developer to create a service. The service will then be described with WSDL for deploying. The service provider publishes the service in a Service Registry, which can be used for service requesters in public. The Service Registry contains complete information about the service. The service requester will look for services, which it needs and retracts service information. Now, service requester can communicate with its corresponding service. Certainly, if requester has enough information about the service, then the required service could be accessed directly.

2.3 What is XML-RPC?

XML-RPC emerged in 1998, published initially via UserLand software and implemented in their Frontier product. XML - RPC (Extensible Markup Language - Remote Procedure Call) is a simple protocol using XML messages for the function or procedure call between remote hosts. XML-RPC provides an XML- and HTTP-based mechanism for making remote procedure calls across a network. The authors' definition is as given below:
„. . . XML-RPC is XML over HTTP, and a great way to develop Web Services. But there's actually more going on here - there's a philosophy to XML-RPC that is different from other software projects. The philosophy is choice, and from choice comes power, and interestingly, a disclaimer of power. . . " [Dave Winer, Userland Software , 2001]
XML-RPC builds own instance parameter of objects based on XML elements and posts as XML-RPC request. This object will be decoded simply at the Server. A response will be generated and it converted in XML again then will be sent to client. This response is called XML-RPC response. XML-RPC is structured as three small parts: XML-RPC data model, XML-RPC request structures and XML-RPC response structures.

2.3.1 XML-RPC data

XML-RPC parameter allows the following types.

Type	Value	Examples
Int or i4	32 bit integers 2,147,483,648 - 2,147,483,647	<int>11</int><i4>11<i4>
Double	64-bit floating-point numbers	<double>11.27134</double>
Boolean	1(true) or 0(false)	<boolean>1</boolean>
String	ascii or unicode text	<string>Hello!</string>
DateTime.iso8601	dates in ISO8601 format CCYYMMDDTHH:MM:SS	<dateTime.iso8601>20040111T 6:8:5 </dateTime.iso8601>
base64	binary information encoded as base 64, as defined in RPC 2045	<base64>zisfSDf=/hf</base64>

Tabelle 1: Number of turns and distance between top and bottom.

XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned. Following two examples show us the complex types, which are allowed by XMP-RPC data.
1. Array type. This one dimensional array contains three elements which are one string, one integer, one boolean, and two integer values.

```
<array>
   <data>
      <value><i4>135</i4></value>
      <value><string>Koeln</string></value>
      <value><boolean>1</boolean></value>
      <value><i4>368</i4></value>
   </data>
</array>
```

2. Object type. In this example, the city struct contains a city name and a value.

```
<struct>
   <city>
      <name>Aachen</name>
      <value><i4>200</i4></value>
   </city>
   <city>
      <name>Freiburg</name>
      <value><i4>520</i4></value>
   </city>
</struct>
```

2.3.2 XML-RPC request

XML-RPC requests consist header information and a body. The body is implemented in XML. The Root element of body is methodCall. Each methodCall contains one methodName element and one params element.

```
<methodCall>
   <methodName>myservice.getDistance</methodName>
   <params>
      <param>
         <value><string>Aachen</string></value>
         <value><string>Frankfurt</string></value>
      </param>
   </params>
</methodCall>
```

2.3.3 XML response

XML-RPC response is like request. MethodCall is replaced by methodResponse. XML-RPC response can contain only one parameter namely params. If a problem occurs in processing of the request then methodResponse element contain fault information. But HTML header information would be always "HTTP/1.1 200 OK"!

```
<?xml version="1.0"?>
<methodResponse>
   <params>
      <param>
         <value><string>260</string></value>
      </param>
   </params>
</methodResponse>
```

If you want to develop a Web Services with XML-RPC you can see more details from Programming Web Services with XML-RPC/O'Reilly". [3] A more detail description of XML-RPC is available at the `http://www.xml-rpc.com`

2.4 What is SOAP?

SOAP is an abbreviation for Simple Object Access Protocol. SOAP allows applications to pass messages, which are a unit of communication with a Web Service, representing the data exchanged in a single logical transmission, to one another independent from any platforms via HTTP, SMTP protocols. SOAP specification defines a XML structure for the action of requests and responses but doesn't define how the requestor or the responder transmit and receive messages. The implementation of receiving and sending is not the SOAP point but the software developer affair. SOAP message is structured as follows. The Messages are used as an *envelope* where the application encloses whatever information needs to be sent. SOAP Envelope provides a mechanism to identify the contents of a message and to explain how the message should be processed. A SOAP envelope includes a *header* and a *body*. The SOAP header provides an extensible mechanism to supply directive or control information about the message. The SOAP body contains the payload that is being sent in the SOAP message.
All data transmitted through SOAP messages are encoded using XML. SOAP 1.1 defines bindings for HTTP and the HTTP Extension Framework. Services may be designed to work on the raw XML payload, but it is more common for the payload to be mapped or bound directly to data types in the host language. SOAP 1.2 defines a default binding for HTTP and provides for other bindings such as SMTP, JMS and others. Data can be passed as a literal XML document that validates against some XML Schema document. SOAP is simple extensible and lightweight alternative to CORBA and he supports peer-to-peer communications.

2.4.1 Creating a request

In my example „Myapplication" will call the „getDistance" function that is to stand on a server across the Internet. „Anyapplication" makes up a SOAP message and use HTTP to send it. The request message looks like as below.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1">
<SOAP:Body>
   <getDistance>
        <fromCity>Aachen</fromCity>
        <toCity>Frankfurt</toCity>
   </getDistance>
```

```
</SOAP:Body>
</SOAP:Envelope>
```

There is a tag for the method name and tags for the parameters in the SOAP Envelope tags.

2.4.2 Creating a response

When message is received on the remote server then the object is created and the method is called, sending onward two parameters. Once the method is completed processing, it creates a SOAP message to send the result back to Myapplication". The response message looks like as below.

```
<?xml version="1.0" encoding="UTF-8">
<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1">
 <SOAP:Body>
   <getDistanceResponse>
   <value>260</value>
   </getDistanceResponse>
</SOAP:Body>
</SOAP:Envelope>
```

SOAP message contains the name of the response and the value of the result. This message is returned to Myapplication"where it is worked up. SOAP is more flexible and more efficient in particular for requests vs. XML-RPC. SOAP will make possible for systems to become highly distributed. Developers will be able to simply rely on the survey and existing code of other developers to more quickly build more reliable systems.

2.5 What is WSDL and why it used?

WSDL is an abbreviation for Web Services Definition Language. WSDL is a Web Services description mechanism used to describe and locate Web services. Its goal is to define the services understandable by communication protocols. I can define WSDL as an Interface Definition Language, written in XML, for architecting Web Services applications. W3 has provided following more precisely definition.
„WSDL is extensible to allow the description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME“. [`http://www.w3.org/TR/wsdl`]
There is still no standard for WSDL. W3 Consortium has been busily working on

this task. [http://www.w3.org/]
More simply, WSDL describes what Web Services can act, where it is, and how is accessed by user. WSDL specifications are characterized by an abstract part that describes the interface and a concrete part that defines the binding information. This is personated below in figure 3. WSDL uses the same type of system as XML Schemas, although the WSDL document can specify a diverse type of system if necessary.

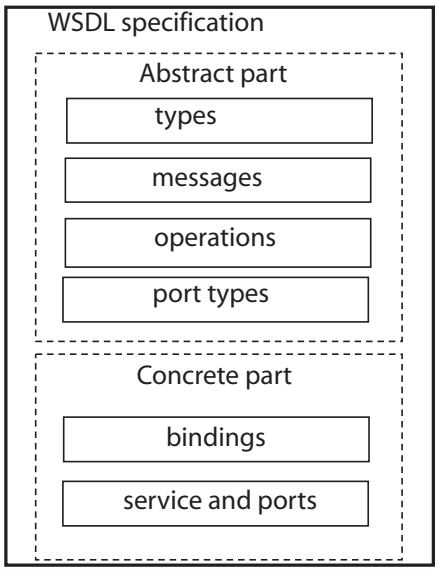

Abbildung 3: A WSDL service specification [5]

„*Definitions*“ element defines set of related services. Located at the root, inside the element, definitions of other elements are defined. It contains the attributes ‘name’, which specifies the name of the service, the target namespace, and other standard namespace definitions.

```
<?xml version="1.0" ?>
<definitions xmlns:SOAP-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:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:si="http://soapinterop.org/xsd" xmlns:tns="http://openmn.org/WS"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
     xmlns="http://schemas.xmlsoap.org/wsdl/"
     targetNamespace="http://openmn.org/WS">
     . . .
</definitions>
```

„*Types*“ element contains data type definitions. Usually, this element includes a schema element that defines various data types.

```
<types>
<xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://openmn.org/WS">
<xsd:complexType name="DistanceData">
<xsd:all>
<element name="dist" type="xsd:int"/>
</xsd:all>
</xsd:complexType>
</xsd:schema>
</types>
```

„*Message*“ element describes the message parameters and return values with their names. Messages that are defined are associated with their respective operations as input and output messages. A message is protocol independent, so it may be used with HTTP GET, SOAP, or any other protocol. To use web services in a remote procedure call model, there are two messages that must be described. They are the input (or request) message, which is sent from the client to the service, and the output (or response) message, which is sent back the opposite way. Each contains zero or more elements that describe the content of the message.

```
    <message name="getDistanceRequest">
        <part name="fromcity" type="xsd:string"/>
        <part name="tocity" type="xsd:string"/>
    </message>
    <message name="getDistanceResponse">
        <part name="return" type="tns:DistanceData"/>
    </message>
```

„*port type*“ element defines the operations using the <operation > element. The <operation > elements define the syntax for calling all methods in the Port Types and input output messages. An operation is a transmission primitive that an endpoint can support. A portType is „a named set of abstract operations and the abstract messages involved.“ It is a collection of one or more associated operations.

```
<portType name="DistancePortType">
      <operation name="getDistance">
        <input message="tns:getDistanceRequest"/>
        <output message="tns:getDistanceResponse"/>
      </operation>
</portType>
```

„*binding*“ element is used to specify message format and protocol details for each port. SOAP specific information stands here. The style attribute specifies the style of request, which can be rpc or document-RPC for messages containing parameters and return values, and „document“ for messages containing one of more documents.

```
<binding name="DistanceBinding" type="tns:DistancePortType">
```

```
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getDistance">
  <soap:operation soapAction="http://localhost/seminar/server" style="rpc"/>
     <input>
        <soap:body use="encoded" namespace="http://openmn.org/WS"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
     </input>
     <output>
        <soap:body use="encoded" namespace="http://openmn.org/WS"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
     </output>
  </operation>
</binding>
```

„service“ consists of a collection of related port addresses, with each port specifying a particular service. A WSDL may have 0, 1 or more <service> elements. Each port element references a unique <binding> element, specified in the binding section.

```
<service name="Distance">
<port name="DistancePort" binding="tns:DistanceBinding">
<soap:address location="http://localhost/seminar/server" />
</port>
</service>
```

WSDL can be used in different ways and for different purposes as any interface definition language. The W3C Web Service Description Working Group has documented and published use cases that illustrate how a Web Service could be described in WSDL. The Working Group has detected three potential uses of WSDL description. Three of the most important potential uses are:
1. A WSDL description indicates how to interact with the service, what data needs to be post, what data is to expected in return, what operation are involved and the protocol and format necessary to invoke the service.
2. A another important use is as input to stub compilers and tools that, given a WSDL description, will generate the required stubs and additional information for developing both the service and the clients that invoke the service. This usage is presented below in figure 5.
3. A WSDL description to capture information that will eventually allow designers to cause about the semantics of a Web Service. Currently, each Web Service must use a separate specification to pin down the actual semantics or use some other mechanism outside the WSDL description to establish the necessary conventions. One quite interesting to note that, in most scenarios, the WSDL itself is automatically generated based on the API of the application. From the WSDL, stubs and skeletons can be derived, as mentioned above. This is shown in figure 4.

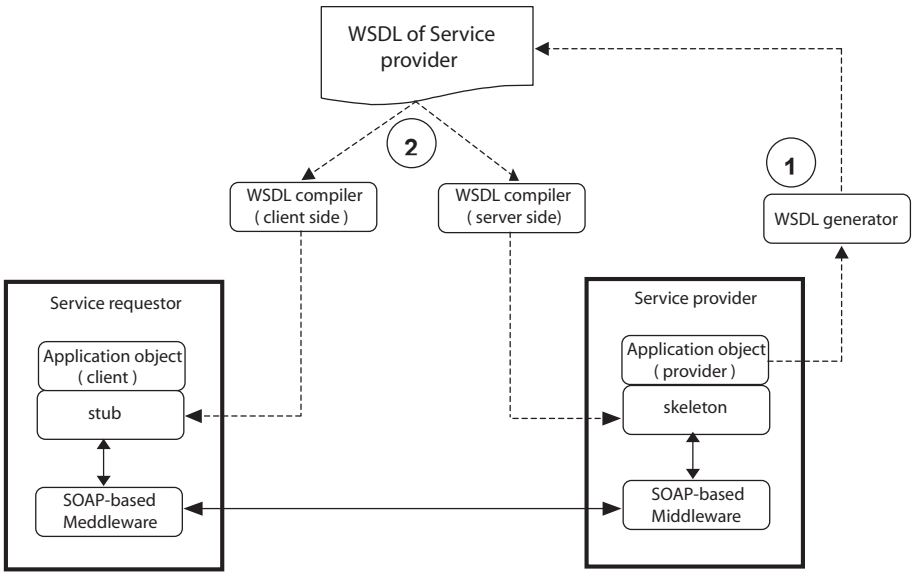

Abbildung 4: WSDL documents can be generated from APIs. Dashed lines represent compile-time activities. First, WSDL is generated. Next, stubs and skeletons are created. [5]

2.6 What is UDDI?

UDDI is a project that was launched by IBM, Ariba, and Microsoft as collaboration. „UDDI creates a standard interoperable platform that enables companies and applications to quickly, easily, and dynamically find and use Web Services over the Internet. UDDI also allows operational registries to be maintained for different purposes in different contexts. UDDI is a cross-industry effort driven by major platform and software providers, as well as marketplace operators and e-business leaders within the OASIS standards consortium.“ [6]

UDDI is itself a Web Services based on XML and SOAP and makes a list of address data and achievement data as well as application-interfaces of different Web Services available. It provides mechanism for run-time discovery from UDDI servers.
`www.uddi.org` encloses programmable interfaces to dynamic discovering and integrating of Web Services.
UDDI specification defines data structures, they contain four type of information: businessEntity, businessService, bindingTemplate and information about descriptions of services known as tModel.
A **businessEntity** contains information about an organisation that provides Web

Services, including its name, URL, a short description, and some basic contact information.
A **businessService** element is a group of business services offered by the businessEntity. Each businessService entry contains a business description of the service, a list of categories that describe the service, and a list of binding templates that point to technical information about the service.
A **bindingTemplate** element describes the technical information on where to find the service and how to use the service. A businessService entry can contain multiple bindingTemplate elements but bindingTemplate belongs to only one BusinessService. The bindingTemplate also associates the businessService with a tModel.
A **tModel** element, namely „technical model“ contains information such as the name of the organization that published the tModel, a list of categories that describe the tModel, and pointers to technical specifications for the tModel. For instance, a tModel may point to a WSDL document that describes an abstract service type. When looking for a Web Service, a developer queries the UDDI Registry, searching for a business that offers the type of service that he wants. From the tModel entry, the developer can obtain the WSDL description describing the service interface. From the bindingTemplate entry for the specific service, the developer can obtain the service binding and access point. Using the WSDL description, the developer can construct a SOAP client interface that can communicate with the Web Service.

UDDI is a rather wide and a bit formal concept. A registry system of a university could be a small UDDI! The main goal of UDDI is, on the one hand to support developers finding information about services, on the other hand to enable dynamic binding by allowing clients to query the registry and provide to services of interest. Information are stored in UDDI registry into three simple categories.
White pages These are listings of business names, of contact information such as names, phone numbers, websites, and known identifiers of company.
Yellow pages This category includes general classifications of both companies and Web Services according to taxonomies that are industry, product/services and geographical locations.
Green pages This information is used to describe how a given Web Services can be invoked. It can be also categorized.

2.7 Web Services standards

Web Services are still in the early development stage. „We have not created multiple standards, but rather we’ve gone out and supported interoperability,“ said

Philip DesAutels, Microsoft's product manager for XML Web Services. It is safe to assume the goal is to have one standard. Currently, two standards groups are working on the definition of official Web Services standards:
W3C and the Organization for the Advancement of Structured Information Standards (OASIS). W3C concentrates on core infrastructure specifications, and OASIS concentrates on higher-level functionality. W3C initiated its Web Services standardization efforts with the launch of the XML Protocol Working Group (XMLP) in September 2000. XML forms the basis of all Web Services-standards. It is a meta language with which languages can be described in which then data can be defined. All the other standards for Web Services refer to such data definitions. With XML every calculator can access directly the news on the Internet, irrespective of the operating system platform. SOAP takes in the pile of the Web Services-technologies a place as a standardized packaging protocol for messages which are exchanged between applications. This protocol fixes how functional calls with XML data are organized. It would be still pointed out to the fact that SOAP not only about HTTP can be transferred, but also about other transport protocols, as for example SMTP, FTP.
You can find more detailed information for standards and coherence of Web Services from following sources: `http://www.w3.org/2002/ws/`
„Programming Web Services with SOAP, By Pavel Kulchenko, James Snell, Doug Tidwell, Appendix A."

3 Integration of Web Services

The analysts are saying Web Service is the ideal solution for all IT systems integration tasks. "Web Services are today already interoperable and they simplify information exchange. Actually, the main consideration of the Web Service is lightweight integration of the new services and reusable, common-sense coupling for old software systems. It is mentioned above that Web Services are application to application oriented. UDDI registers the known Web Services and distribute to public area. It has enabled very nice future in the business environment. Because all company services can communicate with each other very effective, fast and independently from any platforms as well as without affecting the infrastructure of company programs. In this part will be implemented an example Web Services, which is namely City Distance Calculating". You can see details also following literatures.
„Web Services Building Blocks for Distributed Systems: By Dianne Kenne-

dy ISBN:0-13-066256-9“ „Microsoft .NET for Programmers: By Fergal Grimes ISBN 1-930110-19-7“

3.1 Service creating

The requirements for Web Services development from the developer point are summarized as: a standard way to represent data, a way to discover service providers, and a common, extensible message format and service description language. As standard representation of data used the XML Schema, which defines XMLs type system. Common Service Description Language provides a way for service providers to describe the basic format of Web Service requests over different protocols or encoding. WSDL is a template for how web services should be described and bound to clients. UDDI provides a mechanism for clients to dynamically find other web services. A UDDI registry is established to allow for clients to obtain services and bind programmatically to them.
As consumer we need only WSDL specification. What do we need to communicate with different Platforms and programming languages? The answer is standards! Standards are strongly recommended. There are surprisingly many SOAP Implementations and Toolkits today available to developers. The most popular tools: Apache SOAP for Java, SOAP::Lite for Perl, Tomcat, and Microsoft .NET. No matter which toolkit you use, the fundamental process of creating, deploying, and using SOAP web services are same. There are SOAP toolkits for all the popular programming languages and environments (Java, C, C++, Perl, PHP, Python, and much more). Now, how do we bind SOAP to a Transport protocol? SOAP does not force any transport protocol. In most common case it is associated with HTTP but it can be used with other protocols such as SMTP. The specification of which protocol to use is called a binding, which is defined as 6th activity in figure2 (Web Services architecture). There exist two type of binding, which are static and dynamic.
Static binding: Developers can bind clients to services either at runtime or at compile time. Using the WSDL how (concrete) part, a developer can compile a concrete SOAP client interface or stub that implements the binding required to communicate with a specific Web Service implementation. This pre-compiled stub can be included in a client application. The access point can be specified at runtime.
Dynamic binding: A WSDL document is machine-readable thus dynamic binding is supported. Using just the WSDL what (abstract) part at compile time, a developer can generate an abstract client interface that can work with any implementation of a specific service type. At runtime the client application can dynamically compile the WSDL where part (containing the how part) and create a dynamic proxy that implements the binding. When SOAP is used over HTTP, what is being sent is the SOAP envelope within an HTTP request. Also SOAP can use GET, POST or other HTTP primitives. I have written a small Web Service example in PHP. This example is very small but shows programmers how easy to build Web Services. Even though PHP does not have a SOAP extension, but there are some nice PHP SOAP tools. One of these is NuSOAP toolkit, which is used by my example. NuSOAP (formerly SOAPx4) is a toolkit that provides simple API for building Web Services using SOAP. NuSOAP current version is 0.6.4,

which provides simple API for making SOAP server/client applications and also supports features like WSDL generation, building proxy class, using SSL, using HTTP proxy, HTTP authentication.
Our server is going to create the Web Services. For a server we (as developer) need to create a soap_server object.

```
$soapServer = new soap_server();
```

We will register one method (getDistance) that accepts string type of two parameters (fromcity, tocity) and return an array with distance.

```
// register method
$soapServer->register('getDistance', array('
fromcity' => 'xsd:string', 'tocity' => 'xsd:string'),
array('return'=>'tns:DistanceData'),
'http://openmn.org/WS');
```

Than it would be invoked by a client. What follows is the body of actual „getDistance“ function. First we check if argument is a string and return soap_fault if it's not. Our string parameters (fromcity, tocity) are used in SQL query, where we hopefully get some data. In case of error, or if there is no data accessible, we also return soap_fault. If we got some data from the database, we return it like associated array (our complexType). We pass to our service incoming data $HTTP_RAW_POST_DATA. By the way, $HTTP_RAW_POST_DATA is only set if type of the data is unknown. Please see the complete source code, that is included in Appendix.
This module will be the code that sits behind our Web Service interface. Our Web Services are ready to be used.

3.2 A simple implementation of SOAP

The implementation of SOAP based interaction follows the same principles as the implementation of RPC. The client application invokes the service as a local call. The call is in reality an invocation of a proxy procedure located in a stub appended to the client at compile time. Then client stub invoke SOAP engine to prepare SOAP message. SOAP engine packages SOAP into HTTP format and passes it to HTTP client module that will forward the request to the remote location. The HTTP server module gets the HTTP message at the remote location and passes the content of the message to the SOAP router. There, the HTTP wrap removed and the XML document is extracted and analysed to retrieve its content. The router passes the message, identifies the appropriate stub, and delivers the parsed message. After that the server stub (skeleton) invokes the local procedure of the service. The responding activity is treated in a similar manner. The SOAP engine and the

stub can be combined or they can be independent modules. If the SOAP part is independent, then it is often represented as emphSOAP router, since one of its tasks is to route the call to the appropriate objects.

3.3 Service describing

The Web Services definition in fundemantals of Web Services part characterized that Web Services are emphself-described. Here, we describe what that means. The SOAP specification does not address any description. The standard specification used to make Web Services self-describing is the Web Services Description Language (WSDL). Using WSDL, a Web Service can describe everything about what it does, how it does it, and how consumers can use it. There are several advantages to using WSDL:
1. WSDL provides a simplicity to implement and maintain services by providing a more structured approach to defining Web Service interfaces.
2. WSDL provides a simplicity to consume web services by reducing the amount of code (and potential errors) that a client application must implement.
3. WSDL provides a simplicity to implement changes that will be less likely to „break“ SOAP client applications.

WSDL is not perfect, however. Another key point is that, for the most part, web service developers will not be required to manually create WSDL descriptions of their services. Here, it is interesting to mention that in most scenarios, the WSDL itself automatically generated based on API of the existing application. Many toolkits supports this property. A WSDL service implementation description must provide is the network location where the web service is implemented. This is done by linking a specific protocol binding to a specific network address in the WSDL service and port elements. In our example, at the next line of soap_server object we are starting to generate a WSDL file. This file basically describes the Web Services and let us know how to use Web Services or to access it.

```
// wsdl generation
$soapServer->debug_flag=false;
$soapServer->configureWSDL('Distance', 'http://openmn.org/WS');
$soapServer->wsdl->schemaTargetNamespace = 'http://openmn.org/WS';
// adding complex type
$soapServer->wsdl->addComplexType('DistanceData',
    'complexType',
    'struct',
    'all','',
   array('dist' => array('name'=>'dist', 'type'=>'xsd:string'))
);
```

You can see more details the complete source code in Appendix.

3.4 Services discovering

In the first part we have experienced what is UDDI registry. A registry is impossible to use without some way to access it. Generally the UDDI standard describes two SOAP interfaces for service providers and consumers to interact with the registry. UDDI InquInquire API includes operations to find a service, and UDDI InquInquire API enables to providers to add, modify, list, and delete entries in the registry. The biggest UDDI providers are:
`http://uddi.microsoft.com/` (Microsoft)
`http://uddi.ibm.com/testregistry/registry.html` (IBM)
`http://udditest.sap.com/` (SAP)
To register our Web Services we should use one of the above UDDI providers. Please visit to anyone, for example to the `http://uddi.ibm.com/testregistry/registry.html`. There exist four fields that are described in above part. (What is UDDI) You can enter all specific information and publish own Web Service as developer. If a service is registered into UDDI then everyone, who uses UDDI search. That is the advantage and main goal of UDDI. Now, it is possible someone to write own program, who has interest to use our Web Services. The full client implementation is attached in Appendix. We can also use WSDL to create proxy class, which will hold method (getDistance) of our Web Services.

```
$soapClient = new soapclient('http://127.0.0.1/distance.wsdl', 'wsdl');
$soapProxy  = $soapClient->getProxy();
```

So, basically the programmer only needs to know webmethods name and required parameters, and all this can be retrieved from WSDL. We can save WSDL file locally and use it. So let is make a SOAP client using WSDL. In „What is WSDL?“ section of Part 1 our WSDL file completely implemented.
Now, we can present more precisely the working scenario, how existing services can be made available as Web Services, based on above discussions, what are SOAP, WSDL and UDDI. We discuss the case of emphstored procedures. A similar way is used for conventional middleware services such as EJBs, CORBA objects, and COM object. The interaction is represented by figure. The stored procedures already have a defined interface that describes the name and parameters of procedure. This information translated into description of Web Services with WSDL structure. (step 1) The generated WSDL is stored at the provider's site. A WSDL compiler can then create a server stub and register it with the SOAP router, so that the router knows that it has invoke server whenever a certain uniform resource identifier (URI) is invoked.The stub will turn invoke the object. (step 2) This constitutes the imlementation of the newly created Web Services. When service is operational and can be invoked. But nobody knows about its existence as yet. Thus, one more step (3) is needed. The final step is performed by a UDDI client application and contains of two parts. The first part consists of publishing a tModel that associated to the generated WSDL in some UDDI registry accessible to the particular clients. The second part consists of publishing a new besinessSer-

vice and bindingTemplate elements, pointing the address at which the service is available, and referencing the newly created tModel. Now, the Web Service is online and ready to use. The consumers can develop client application using automatically generated client-side stubs through WSDL compilers.

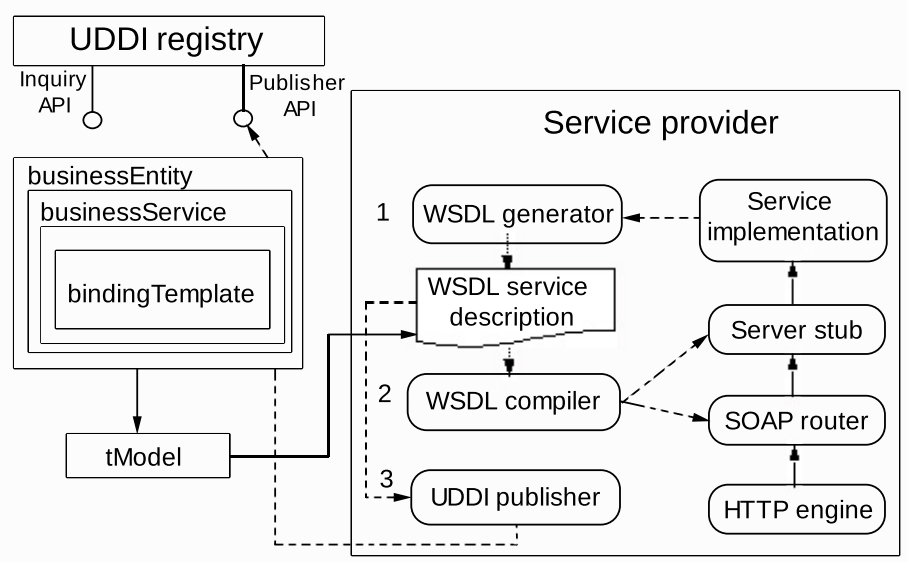

Abbildung 5: Exposing an internal service as a Web Service

3.5 How integrate two or more various applications?

Web Services provide better automation and better tooling and run time support, easier communication for applications. These make easier the enterprise application integration (EAI) but do not solve the EAI problems in and themselves. Any type of programs can be presented as a Web Services, which do not need to be a accessed via internet. It is possible to make web Services available to clients in LAN. We have to write XML specification adapters for each application additionally for integration of old applications. Please see figure7. Then these adapters could communicate each other with Web Services. We don't need to change a code in old applications. That is one of the great advantages of the use Web Services.

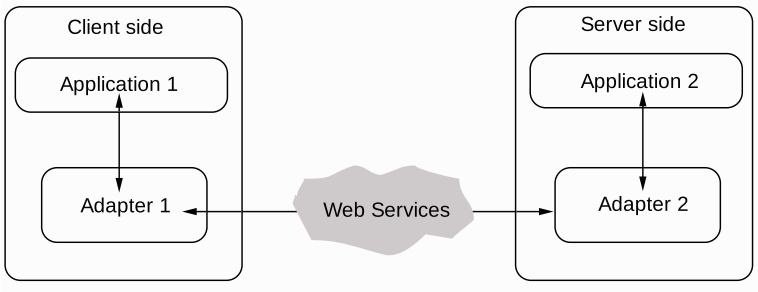

Abbildung 6: Integration

3.5.1 XML application interface

An XML Application Interface can be written as an XML-API. Thus it looks already better. Of course XML technology plays a prominent role for the interoperability of applications. XML provides a generic, application and human readable, consistent data format to manipulate all over enterprise and as well as a flexible, adaptable interface, which is simply maintained using standardized XML tools. XML-API illustrated as follows. The adapter translates application data to XML

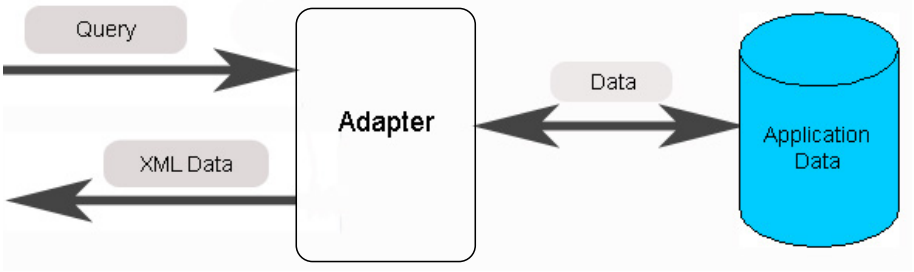

Abbildung 7: XML-RPC interoperability

data and vs. The interface is specified by an XML schema, which encapsulates the application and its data. Therefore, the resulting system is loosely coupled in the traditional way, that is to say XML schema may be extended without changing of the company's infrastructure. New applications call the adapter simply and get data but old applications integrated with help of this adapter without affecting.

3.6 Securing Web Services

Service requests and responses are encoded in XML as SOAP envelopes, and transported over HTTP. Thus, XML communications can be encrypted via the Service Sockets Layer (SSL). SSL is reliable technology, is widely applied and maintains the security and integrity of the transmission channel by using encryption, authentication and message authentication codes. XML is human and application readable programming language. Therefore, it is critically important for companies to secure their XML and Web Services traffic. Companies and organizations must to focus on their security infrastructure with centralized XML and Web Services security policy definition and control. The minimal security is firewall for HTTP. It could not ensure the content of SOAP message envelopes. There established today many security toolkits. The first one is SOAP security Extension: Signature (SOAP-DSIG). DSIG adopts public key cryptography to digitally sign SOAP messages. DSIG has been submitted to W3C and you can find more explicitly at `http://www.w3.org/TR/SOAP-dsig/`.

Here is a good explicit documentation focus on SOAP Message Security. (By OASIS) `http://www.oasis-open.org/committees/download.php/3281/WSS-SOAPMessageSecurity-17-082703-merged.pdf`

OSIS consortium develops the Security Assertions Markup Language (SAML) is an XML-based framework for Web Services that enables the exchange of authentication and authorization information among business partners. SAML is designed to deliver much-needed interoperability between compliant Web access management and security products. SAML addresses the need to have a unified framework that is able to convey security information for users who communicate with a provider so they can seamlessly interact with another but he does not address privacy policies, however. Rather, partner sites are responsible or developing mutual requirements for user authentication and data protection. You can find here more about SALM. (`http://www.oasis-open.org/committees/download.php/4865/sstc-saml-core-2.0-draft-02.pdf`)

Microsoft has released a toolkit, called Web Services Enhancements (WSE). You can find about this tools from: `http://www.msdn.microsoft.com/webservices/building/wse/default.aspx`.

There are several companies have to put for onward the XML Key Management services (XKMS). XKMS is designed to simplify the integration of PKI and digital certificates (which are used for securing Internet transactions) with all kinds of applications. The specification is available online at `http://www.w3.org/TR/xrms/`.

3.7 Web Services problems

There are drawbacks to Web Services because the technology is still very new and partly still in the development. Hence, standards are changing the versions quickly and might make older versions incompatible very quickly. Web Services could not guarantee security 100In the general occur following problem formulations, which are still enough unsolved partly:

How can be guaranteed the security, availability and performance?

How behave cooperating services?

How can two partner protocols communicate?

How the services can be localized?

How can be realized effective, reliable and cost friendly services?

How practicable are the topical Web Services architectures?

From the consumer side: How does one find the most suitable service?

There is a difficulty for the developer in practice. For writing a Web Services in Java you can get a RPC problem to analyse the variables from server, i.e. Web Services are very restricted, so couldn't run most of the classes and return permission errors. Thus, the Web Services develops should observe for deploying of Web Services.

3.8 B2B

B2B stands for Business-to-Business and refers to Electronic Commerce between businesses rather than between a business and a consumer (referred as B2C E-Commerce). Businesses can often deal with thousands of other businesses, either as customers or suppliers. There are obvious advantages for conducting these transactions electronically over traditional methods. It's faster, cheaper and more convenient. Electronic transactions have been around for a while in the form of EDI (Electronic Data Interchange). Web Services found their main use in the business world in which e-commerce is increasingly inevitable regarding competition. Many companies have got success in e-commerce by using Web Services technology and engaged in business-to business e-Commerce. For instance: T-Mobile (a division of Deutsche Telekom), Amazon etc. UDDI plays main role in B2B world. Any company can access the registry on the Internet enter the description of its business, reach to a UDDI site and search through all the business services listed in the UDDI registry. There is no cost to access information in the registry.

Relevant literatures are:
Arthur Sculley and William Woods, „B2B Exchanges: The Killer Application in the Business-to-Business Internet Revolution“
Michael J. Cunningham, „B2B: How to Build a Profitable E-Commerce Strategy“
Martha Rogers, Don Peppers „The One to One B2B“
Matthew Friedman „Understanding B2B“
Erik Brynjolfsson, Glen Urban „Strategies for E-Business Success“
Barry Silverstein, Jeffrey P. Papows „Business-to-Business Internet Marketing“

4 Summary

The purpose of this writing was to introduce how can be realized a Web Services and how write Web Services. A big advantage for a business is that data can be transferred without detailed knowledge of the other's IT system. We can access into entire company Infrastructure using only WSDL file. Web Services orientation was human to application centric. Web Services make the application-to-application programming interface available. It is extremely helpful. For examples include shopping, language translation credit card verification and more. Web Services offer also integration of heterogeneous systems. The purposes of Web Services are providing a flexible global integration between applications, improvement of business processes and saving of costs. The use of Web Services accelerates the business processes supporting widely used standard protocols.

4.1 Web Services today

Web Services made global e-commerce revolution. They allow companies to easily integrate their strategic applications with those of their partners, both internally and over the Internet. They can create cost-effective, flexible methods for conducting B2B transactions. First of all, the development of the Web Services technologies powered by Microsoft and IBM. Microsoft has released Ṅet platform that oriented completely Web Services technology. The Web Services are based on various new standards. But XML, SOAP, WSDL and UDDI are still leading the technology. These standards are supported by most of OpenSource products so that a very strong competition to the commercial products is emerged. Many development tools programming languages supports this technology. A lot of companies have success by using Web Services in business environment. Web

Services also offer the promise of automated Web. The best real world examples of Web Services are Amazon's shop system and google search system.

4.2 Future/potential of Web Services

Web Services essentially make the Internet itself the basis of a new operating system, and this has software vendors excited about the future. Most businesses are now connected to the Internet, and the long-term vision for Web Services goes beyond simply integrating existing programs to delivering plug-and-play software applications on demand over the Internet[7]. In the long term, Web Services also offer the promise of the automated Web. If Web Services easily discoverable, self-describing, and stick to common standards, it is possible to automate application integration[1]. Users will have no problems upgrading or troubleshooting applications. Web Services build the reliable system for an open, flexible, cooperative and low-cost environment for business-to-business e-commerce.

Appendix The example of Web Service greating, which used in section 3-1. (written in PHP)

```
<?php
//
require('inc/nusoap.php');
// create server
$soapServer = new soap_server();
// wsdl generation
$soapServer->debug_flag=false;
$soapServer->configureWSDL('Distance', 'http://openmn.org/WS');
$soapServer->wsdl->schemaTargetNamespace = 'http://openmn.org/WS';
// add complex type
$soapServer->wsdl->addComplexType(
   'DistanceData',
   'complexType',
   'struct',
   'all',
   '',
   array('dist' => array('name'=>'dist', 'type'=>'xsd:string'))
);

// register method
$soapServer->register('getDistance', array('
fromcity' => 'xsd:string', 'tocity' => 'xsd:string'),
array('return'=>'tns:DistanceData'),
'http://openmn.org/WS');

// method code (get DB result)
function getDistance ($Start, $Target) {
if (is_string($Start)) {
$DBlink   = @mysql_connect('localhost', 'root', '');
```

```
$DBresult = @mysql_db_query('city', 'SELECT dist
FROM distance WHERE fromcity = LCASE("' .
mysql_escape_string((string)$Start) .'")
&& tocity = LCASE("' . mysql_escape_string((string)
$Target) . '") LIMIT 1');

// simple error checking
if (!$DBresult) {
return new soap_fault('Server', '', 'Internal server error.');
}

// no data avaible for x fromcity
if (!mysql_num_rows($DBresult)) {
 return new soap_fault('Server', '', 'service contains data
 only for a few cities.');
}
mysql_close($DBlink);
//$temp = mysql_fetch_array($DBresult);
// return data
 return mysql_fetch_array($DBresult, MYSQL_ASSOC);
}

// we accept only a string
else {
return new soap_fault('Client', '', 'service requires a
string parameter.');
 }
}

// pass incoming (posted) data
$soapServer->service($HTTP_RAW_POST_DATA);
?>
```

Here is the database table.

Id	fromcity	tocity	dist
1	Aachen	Koeln	60
2	Dresden	Aachen	644
3	Frankfurt	Aachen	460
4	Berlin	Frankfurt	400

Tabelle 2: Distances between two cities.

Here goes the complete source code of Web Service consuming.

```
<?php
/*************************************************************
*  Description:
*  Creates a simple SOAP Client using WSDL (client_wsdl.php).
*************************************************************/
// use form data
if ((string)$_GET['action'] == 'get_data') {

  // includes nusoap classes
  require('inc/nusoap.php');
```

```
  // set parameters and create client
 $soapClient = new soapclient('http://127.0.0.1/distance.wsdl', 'wsdl');
 $soapProxy  = $soapClient->getProxy();

   // call a webmethod (getDistance)
 $Result = $soapProxy->getDistance((string)$_POST['fromcity'],
 (string)$_POST['tocity']);

  // check for errors
  if (!$soapClient->getError()) {


 // print results
print '<h1>>Current data for: '    . (string)$_POST['fromcity'] .'-->'
. (string)$_POST['tocity'] . ':</h1><ul><li>dist: '   . $Result['dist']
. '</li></ul>';
  }
  // print error description
  else {
    echo '<h1>Error: ' . $soapClient->getError() . '</h1>';
  }
}

// output search fclientorm
print '
  <form name="input" action="'.$_SERVER['PHP_SELF'].'?action=get_data"
   method="POST">
    Your fromcity: <input type="text" name="fromcity">
    Your targetcity: <input type="text" name="tocity">
    <input type="submit" value="Search">
  </form>
';
?>
```

Literatur

[1] CERAMI, ETHAN (Herausgeber): *Web Services Essential: Distributed applications with XML - RPC, SOAP, UDDI and WSDL*, Band 1. O'Reilly, 2002.

[2] COMPUTERZEITUNG, 2000-2003. `http://www.computer-zeitung.de/`. Computerzeitung.

[3] DOUG TIDWELL, JAMES SNELL, PAVEL KULCHENKO: *Programming Web Services with XML-RPC*. O'Reilly, Dec. 2001.

[4] FISHER, M.: *Introduction to Web services*. New Riders, Aug. 2002. `htpp://java.sun.com/webservices/docs/1.0/tutorial/`.

[5] GUSTAVO ALONSO, FABIO CASATI, HARUMI KUNO und VIJAY MACHIRAJU (Herausgeber): *Web Services: Concepts, Architectures and Applications*, Band 1. Springer, 2003.

[6] UDDI CONSORTIUM: *UDDI executive White Paper*, November 2001. `http://uddi.org/pubs/UDDI_Executive_White_Paper.pdf`.

[7] W3C CONSORTIUM: *Web Services Architecture Requirements*, 2002. `http://www.w3.org/TR/wsa-reqs/`. W3C Recommendation.