CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

Informatics

Department of Information Engineering

Diploma Thesis

Web technologies for Internet services specification, implementation and utilisation

Author: David Jarchovský
Supervisor: Dr. Ing. Robert Pergl

Czech University of Life Sciences Prague | Faculty of Economics and Management

Department of Information Engineering

Academic year 2010/2011

DIPLOMA THESIS ASSIGNMENT

David Jarchovský

specialization of the study: Informatics

In accordance with the Study and Examination Regulations of the Czech University of Life Sciences Prague, Article 17, the Head of the Department assigns the following diploma thesis to

Thesis title: **Web technologies for internet services specification, implementation and utilisation**

The structure of the diploma thesis:

1. Úvod
2. Cíl práce a metodika
3. Sběr a analýza požadavků na moderní webové služby
4. Rešerše technologií webových služeb
5. Vymezení aplikační domény
6. Analýza a návrh řešení
7. Zhodnocení a diskuse
8. Závěr
9. Seznam použitých zdrojů
10. Přílohy

The proposed extent of the thesis: 50 - 60 pages

Bibliography:

ZIMMERMANN Olaf, TOMLINSON Mark, PEUSER Stefan. Perspectives on web services. Springer-Verlag Berlin Heidelberg 2003. ISBN 3-540-00914-0
ALONSO Gustavo, CASATI Fabio, KUNO Harumi, MACHIRAJU Vijaj. Web services. Springer-Verlag Berlin Heidelberg 2004. ISBN 3-540-44008-9
CARDOSO Jorge, SHETH Amit. Semantic web services and web process composition : first international workshop. Berlin: Springler 2005. ISBN 3-540-24328-3
CERAMI Ethan. Web Services essentials. Beijing : O'Reilly, 2002.ISBN 0-596-00224-6
W3C: Web Services Description Language (WSDL) 1.1.[on-line]Dostupné z: http://www.w3c.org/TR/wsdl
OASIS: UDDI Version 3.0.2. [on-line]Dostupné z: http://www.uddi.org/pubs/uddi_v3.htm

The Diploma Thesis Supervisor: **Dr. Ing. Robert Pergl**

Deadline of the diploma thesis submission: April 2011

.....
Head of the Department

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE
Provozně ekonomická fakulta

.....
Dean

In Prague: 30th March 2011

Declaration

I declare that I have worked on my diploma thesis titled "Web technologies for Internet services specification, implementation and utilisation" by myself and I have used only the sources mentioned at the end of the thesis.

In Prague on 31st March 2011

_____

David Jarchovský

Acknowledgement

I would like to thank name of the supervisor and my family for advice and support during my work on this Thesis.

Web technologies for Internet services specification, implementation and utilisation

Webové technologie pro specifikaci, implementaci a využívání služeb v prostředí Internetu

Souhrn

Diplomová práce se zabývá problematikou konzumování webových služeb v produktově orientovaném prostředí. V práci je navrženo strojově čitelné popisování produktů, které by dokázalo zachytit sémantické souvislosti nabízených webových služeb. Pro účely návrhu je v práci obsažen průzkum vztahů mezi firmami a jejich dodavatelskými subjekty.

Návrh je koncipován takovým způsobem, aby bylo možné implementovat systém s maximálním využitím existujících registrů webových služeb – UDDI. Samotný návrh definuje zúčastněné strany a návrh mechanismu popisování služeb. Popis je navržen tak, aby umožňoval plně automatizovanou interakci stroj-stroj, kde jsou dílčí kroky skryty koncovému uživateli. Popis webové služby umožňuje žadateli prohledávat i sémanticky podobné služby odpovídající jeho požadavku.

Klíčová slova

Webové služby, UDDI, WSDL, UML, analýza, produkt, SOAP, API.

Summary

The thesis deal with the issue of consumption of web services in product-oriented environment. The thesis depict the system of machine readable descriptions which describe semantic relations between offered web services. The research of the relationship between companies and their suppliers is included for purposes of the further design.

The design is conceived that the system could be implemented with the maximal utilization of current UDDI registries. The design himself defines actors and the services description mechanism. The description is designed to allow full automatic interaction machine-to-machine where the partial steps are hidden to an end-user of the system. The description allows a user to search semantically similar services according his demand.

Keywords

Web services, UDDI, WSDL, UML, analysis, product, SOAP, API.

Contents

1 Introduction

Cloud* computing is one of the newest trends in information system design. The implementation of IT technologies to inner processes of a company may lead, according to the research of GS1 Czech Republic**, to decrease of error occurrence and increase in effectiveness of processes. This type of architecture is considered by technology leaders of the IT industry as the architecture of the future. The architecture is based on services provided via Internet.

The concept of Web Services is one of the well known architecture of service oriented applications. Web Services introduced protocols to describe or communicate with a service such as SOAP or WSDL. The important part of service oriented architectures is a registry of services where services and vendors are described, the most known registry solution is UDDI.

Especially public service registries are interesting because they open the way to make services more accessible to users. The issue of the effective services filtering in reasonable time. The next problem of such registries is how to trust a provider and how to be sure companies sell what they declare. If companies could trust the description of a service then the automation of a process of ordering would arise instantly. The electronic universe where machines can understand meanings of words will be necessary.

This thesis will consider the issues of a machine-readable description of services provided via Internet.

---

* This thesis will work with the definition of cloud by Reuven Cohen, CTO of Enomaly Inc. : "The cloud computing is 'internet centric software.' This new cloud computing software model is a shift from the traditional single tenant approach to software development to that of a scalable, multi-tenant, multi-platform, multi-network, and global. This could be as simple as your web based email service or as complex as a globally distributed load balanced content delivery environment."
GEELEN Jeremy. SYS-CON Media. *Twenty-one Experts Define Cloud Computing*. [online] 24th January 2009. [q. 18th March 2011]. Available on WWW: <http://cloudcomputing.sys-con.com/node/612375>

** GS1 Czech Republic is a standardization organization which creates and implements global standards to increase effectiveness and transparency of logistic chains.

2 Objectives and methodology

The goal of this thesis is to design a description of services in a product-oriented cloud system. The focus is put on the maximal utilization of existing web standards. The description of a particular service must be readable and processable for a machine to provide human flexible instruments to search services.

The theoretical part will describe a technology of Web Services centred on descriptive standards, particularly UDDI. The goal is to create a solid base to understand the technical issues of the description of web services.

The companies requirements for their supply chain will be described in the research part of the thesis. The research part will contain a questionnaire and an interview with people from real business environment. The questionnaire is focused on general information about the relationship between a company and its suppliers. The information about customer's preferences over the suppliers qualities is obtained by a questionnaire which will be processed by the ANOVA method. The statistic calculation will be computed in SAS.

The interview is focused on the description of the specific issues of the business environment. The problem of the certification that arise in the interview will be further addressed.

The case study will design a services description model based upon information collected in previous chapters. The service description, in the context of the case study, means a precise information about a service effect. The system will be described by UML diagrams which will first define selected use-cases and then important service description components will be depicted with the use of class diagrams.

The final part of the thesis will bring a discussion about implementability of the designed system with current Web Services standards.

3 Technology of Web services

Before we will start to talk about *Web services* it is necessary to introduce the general paradigm of *Service Oriented Architecture (SOA)* briefly. A service oriented information system is a set of services which communicate with each other. It does not matter if we talk about a single communication between a requester and a service or if we talk about complex transaction where many services are involved and coordinated. The concept of distributed information system is nothing new, the first attempt to create unified interface of middle-ware was *CORBA (Common Object Request Broker)*.

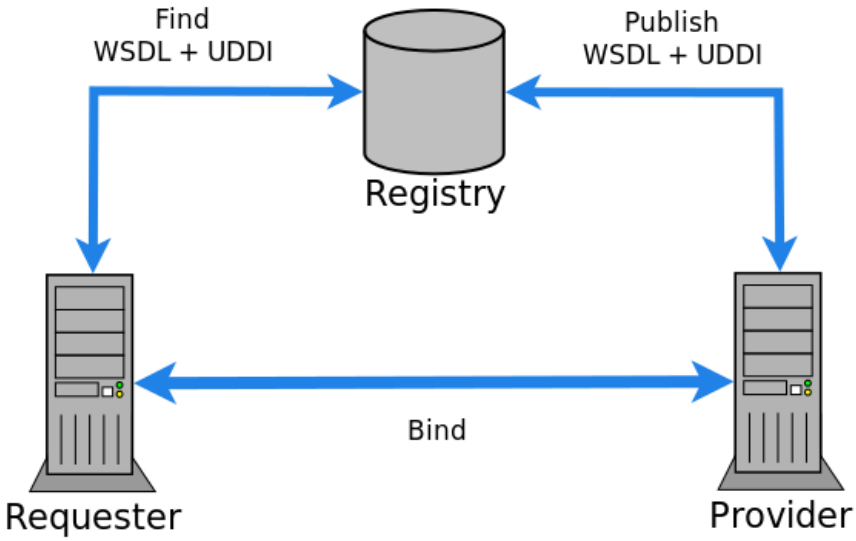

Figure 1: Three sides of consuming of a web service.

The architecture of *Web Services* is the realization of *SOA*. *W3C* uses following definition for *Web services*:

> *"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."* [1]

1 Web Service Architecture [online]. 2004 [q. 20. 10. 2010]. Available on WWW: <http://www.w3r.org/TR/ws-arch/#whatis>

The goal of *Web Services* is to provide tools for co-operation of machines where a platform does not matter. A web service is represented by a hardware or software agent; the agent receives or sends data from, respectively to, a consumer. The agent can change but the service remains the same. Standards of *Web Services* secure the same semanticist of the agents.

The specification describes consumption of a service as an interaction of three sides:

- *Requester* – an agent which requests a service.
- *Provider* – an agent which provides a service.
- *Registry* – a special service which provides description of services.

A provider publishes a service description in *a registry* where it is read by *a consumer* who communicates with *a provider* in the manner defined in the description. The following chapter will describe crucial standards and technologies of *Web Services*.

3.1 eXtended Markup Language (XML)

Backbone standards of *Web Services* technology is based on *XML* serialization. It is necessary to be familiar with *XML* to understand standards described further in this chapter.

This thesis will describe *XML 1.1* which is W3C recommendation since 4th February 2004. The definition was formulated by Netscape, Microsoft, Sun Microsystems Inc. and W3C.

The goal of *XML* is to create a machine-readable format which is easy to process. The standard is focused on the syntax rather than on semantics. The *XML* document contains one and only one rooted tree* in which non-paired tags and simple strings are leafs of the tree.

* The tree is a special kind of undirected graph in which any two vertices are connected with exactly one simple path.

3.1.1 Description of XML document

The *XML* document can contain following structures:

- *character data and markup*,
- *comments*,
- *process instruction* and
- *CDATA*.

The document is split into two parts. A prologue is a header of a document where it is described what should be expected inside the document. The main part of the document is the rooted tree which contains a carried data.

The most common structures are intermingled character data and markups. "Markup takes the form of start-tags, end-tags, empty-element tags, entity references, character references, comments, CDATA section delimiters, document type declarations, processing instructions, XML declarations, text declarations, and any white space that is at the top level of the document entity (that is, outside the document element and not inside any other markup)." [2]

Character data are defined as any string which does not contain the start-delimiter of any markup or CDATA-close-delimiter. These special characters could be replaced by special strings which avoid misunderstanding of a document.

The special kind of the structure is *a comment*. This structure is skipped when a document is processed; it generally contains notes for human readers which supports the document with explanations.

The process instructions contains orders for processing applications. The document has to contain the instruction in the prologue first line which defines the version of the *XML* used.

2 W3C. *Extended Markup Language (XML) 1.1*. [online] rev. 15th April 2004 [q. 19th December 2010] Available on WWW: <http://www.w3.org/TR/2004/REC-xml11-20040204/#syntax>

```
<?xml version="1.1"?>
<!DOCTYPE expression SYSTEM "binary.dtd">
<!-- Prolog is above this comment -->
<!-- this is a comment -->
<expressions>
<expression result="true">
<![CDATA[ 2 < 4 ]]><!-- The start-tag character is omitted. -->
</expression>
<expression result="false">
<![CDATA[ 2 > 4 ]]>
</expression>
</expressions>
```

Figure 2: XML Document with the prolog and main part.

CDATA is special structure where markup is not recognized. It is not necessary to use escape characters inside *CDATA* structure.

A prologue serves contains important instructions about a following document. We can see the prologue part of *XML* documents on the figure 2 highlighted on the very top of the document. The very first line of the prologue has to contain an *XML* document version. *Document Type Definition (DTD)* is the next important part of the prologue of each *XML* document.

DTD is a document where all allowed elements and attributes are declared. The author can define his own document structure. *DTD* is necessary for an *XML* document validation which is important for a proper document processing by a machine. The machine must check the data if they are correct. *DTDs* for W3C documents are available on the W3C's web side.

3.1.2 XML Schema

XML Schema definition (XSD) is a special *XML* document which refers to a particular *XML* document. *XSD* describes elements data types within a particular *XML* document. Data validators follow an *XSD* to check validity of data.

An *XSD* document is composed of the data type definition of *XML* elements and attributes. *XML Schema Definition* contains basic and derived datatypes and it also provides instruments to define own datatypes and structures. The *simpleType* is derived from existing datatypes with various restrictions applied.

The other custom datatype is a *complexType*. The *complexType* allows a user to define

very complicated structures. Such a structure could be defined as an instance of an object**. This structure gives developers a way how to carry an object inside an *XML* document.

We can summarize *XML* as the complex language for creation of machine-readable messages where a document structure definition fits particular purpose. Available document structures definition is described in *XSD* which allows developers to implement automatic data verification.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormat="qualified"
xmlns="http://www.w3c.org/2001/XMLSchema">
  <xs:element name="Person">
    <xs:complexType>
      <xs:element name="FullName">
        <xs:complexType name="FullName">
          <xs:sequence>
            <xs:element name="FirstName" type="xs:string"/>
            <xs:element name="Surname" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Age" type="xs:integer/>
      <xs:element name="education">
        <xs:simpleType>
          <xs:restriction>
            <xs:enumeration value="Elementary school"/>
            <xs:enumeration value="High School"/>
            <xs:enumeration value="University"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="monthOfBirth">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="12"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 3: The example of XML Schema Definition

3.2 Description of a service

The description of a service is divided into two parts. The first part is a description of a service interface and the second is a business background description. The service provider and a service itself is described in *UDDI (Universal Description Discovery and*

** Instances of classes, objects, contain particular values of properties of a class only. The methods are not part of an object.

Integration) in human readable form. *UDDI* contains a technical description address also.

The technical description is stored in a *WSDL (Web Services Description Language*) document. A *WSDL* document defines an exact manner how to communicate with the service.

3.2.1 UDDI

The goal of the definition of *UDDI* is to standardize a publication of services and description of their owners. *UDDI Version 3.0.1.*[*] is the latest full specification. *UDDI* is a special service, rather than a standard, however the way how to describe a service became widely accepted as an implementation of such a service. The specification is supported by 220 companies such as Dell, Fujitsu, IBM, Microsoft, Oracle, SAP etc. The specification is maintained by OASIS.

IBM, SAP and Microsoft brought an idea of registry which would help to establish relations between businesses without a human intervention – UDDI Business Registry (UBR), however, all nodes of UBR were disabled in 2006. The companies concluded that the business environment did not reach the point where companies could trust relations established without human presence. Even that the practical demonstration provided by the UBR helped in the ratification of UDDI specifications as OASIS standards and several software vendors now include UDDI support as a key feature in their software products. UDDI registries are being broadly deployed to solve application and service integration challenges. [3]

The structure of UDDI

The *UDDI* structure is expressed by *XML* (*eXtensible Markup Language*). The specification defines entities which contain the information necessary to reach and consume a particular service. The *UDDI* basic entities:

---

* The latest full specification is 3.0.1 but the draft 3.0.2. exists since 19th August 2004.

3 MICROSOFT. *UBR Shutdown FAQ.* [online] [q. 8th November 2010] Available on WWW <http://uddi.microsoft.com/about/FAQshutdown.htm>

- *businessEntity* describes a business or organization which provides the service.
- *businessService* describes web services related to a *businessEntity*.
- *bindingTemplate* contains an information necessary to consume a service.
- *tModel* contains a technical information about service.
- *publisherAssertion* describes one-sided relationship between *businessEntitites*.
- *subscription* describes a request to keep track of changes to the entities described in this element.

Data structures above are composed together as it is described on figure bellow.

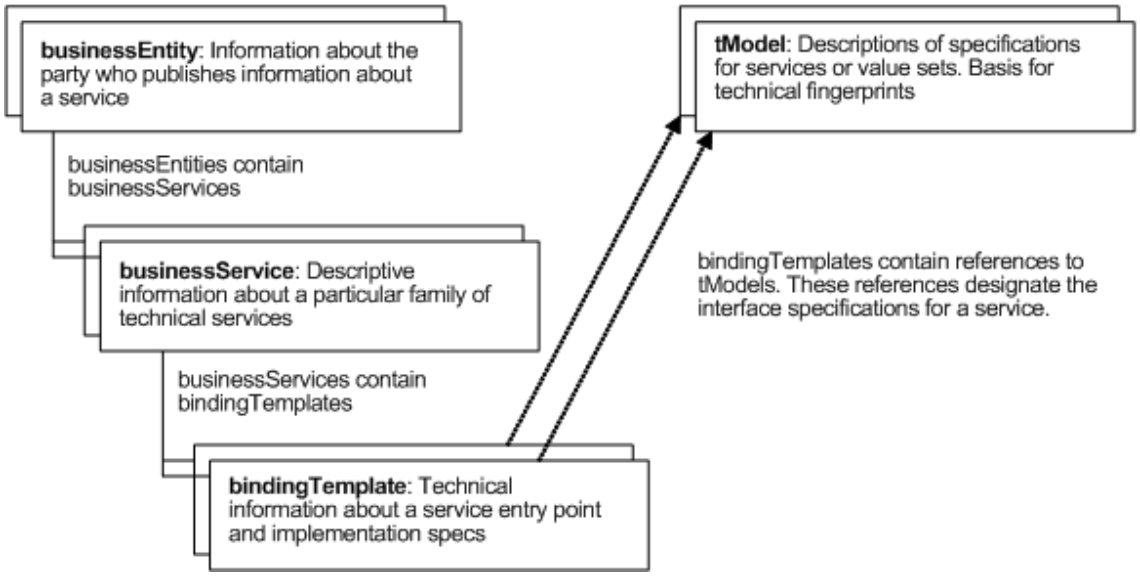

Figure 4: The data structure of UDDI. (Source: http://www.uddi.org/pubs/uddi_v3.htm)

We can divide the data stored in *UDDI* according the information they contain into three groups:

- *White pages* contains general information about companies.
- *Yellow pages* contains general classification of companies or services they offer. (industry, product, or geographic code based on standard taxonomies).
- *Green Pages* contains the technical information about services (a pointer to an external specification and an address for invoking the web service).

Technical specification UDDI

The data within a *UDDI* are kept in a relational data structure. The data are composed to containers and then into final structures. The most of attributes of an element define own implicit value which is, in most cases, a *NULL* value. If we set one of those attributes manually to its default value then we will get semantically the same service as the one without the value set manually but the fingerprint of those two services will be different.

The *UDDI* registry components could decomposed into three main components which should be secured to run a registry properly.

- *UDDI Data model is* an *XML* schema of business or service description.
- *UDDI API Sets* is SOA-based API for searching and publishing all the data structures of a registry.
- *UDDI cloud service* is the special synchronization service with other registries which runs on regular basis. *UDDI cloud services* are currently provided by two companies – IBM and Microsoft. The single private *UDDI* registries does not require *UDDI cloud service*.

UDDI API Sets

The *UDDI API Set* is defined as a service which allows the requester to do a single operation within the registry. They split the service description into different categories accessible by a particular API sets. Seven API Sets are defined within the *UDDI API Sets*.

"Having a common API for accessing the information in a SOA Registry is important when an organization is looking to implement design time governance, run time governance, management of a SOA in the data centre, and discovery in the software development life-cycle. By having a common API for accessing registry information you can purchase the tools that are right for you and be assured that they will work correctly with your SOA Registry."[4]

4 PIJANOWSKI Keith. *The UDDI API sets* [online]. 10th June 2008 [q. 20th March 2011] Available on WWW: <http://www.keithpij.com/Home/tabid/36/EntryID/16/Default.aspx>

Inquiry API Set

This API Set allows an external entity to access the information stored in a registry. The set contains two groups of operations which provides "searching" services and "getting" services.

The first group, searching services, allows requester to find services according defined criteria and the second group returns the information about a *UDDI* entity.

API	Description
find_xxx functions	
find_binding	Searches for bindings associated with specified service.
find_business	Searches for businesses that match specified criteria.
find_relatedBusinisess	Searches for related *businessEntities* which have complete relationship* with given entity.
find_service	Searches for services associated with special businesses.
find_tModel	Searches for *tModels* that match the specified criteria.
get_xxx functions	
get_bindingDetail	Retrieves binding detail.
get_businessDetail	Retrieves *businessEntity.*
get_operationalInfo	Retrieves entity level operational information.
get_serviceDetail	Retrieves *businessService.*
get_tModel	Retrieves *tModel*.

Table 1: Inquiry API Set functions.

Publication API Set

The API could be described as an API to publish, edit and delete core entities of a record and publisher assertions. Publisher assertions define visible relationships between *businessEntity* structures. The assertions are necessary to create *businessEntities* composed from other *businessEntities*, for example, a company and its subsidiaries.

* A complete relationship is bivalent relationship between two entities.

API	Description
add_xxx functions	
add_publisherAssertions	Adds one or more *publisherAssertions.*
delete_xxx functions	
delete_business	Removes one or more *businessEntity* and related content.
delete_binding	Removes one or more *bindingTemplate.*
delete_publisherAssertions	Removes a *publisherAssertions.*
delete_service	Removes one or more *businessService* and related *bindingTemplate.*
delete_tModel	Hides** the information about tModel.
get_xxx functions	
get_assertionStatusReport	Returns publisher assertions and the status information.
get_registeredInfo	Returns abbreviated list of *businessEntity and tModel* of particular publisher.
get_publisherAssertions	Returns list of active *publisherAssertions.*
save_xxx functions	
save_binding	Adds or updates *bindingTemplate.*
save_business	Adds or updates *businessEntity.*
save_service	Adds or updates a complete information about *businessService*
save_tModel	Adds or updates a complete information about *tModel.*
set_xx functions	
set_publisherAssertions	Updates a complete set of existing *publisherAssertions.*

Table 2: Publication API Set functions

Security API Set

Operations from *Security API* Set are necessary when the registry requires authentication; whether a requester needs to be authenticated is given by registry policy. *UDDI* uses a special data structure called *authToken* to recognize a requester.

The authentication token could be requested by other API from the API Sets. The token should be discarded on the end of the session. If a requester does not discard a token then the token will expire automatically. The expired or discarded token can not be used again.

** The *tModel* is still available to a direct access but it is invisible for search.

API	Description
discard_xxx functions	
discard_authToken	Passes authentication token.
get_xxx functions	
get_authToken	Requests authentication token.

Table 3: Security API Set functions.

Replication API Set

The *UDDI* is designed to support a distributed information systems. Therefore we can say that whole *UDDI* contains *UDDI nodes* which are, in fact, single *UDDI* registries. Then the regular maintenance of each registry is necessary to secure correct run of registry as a whole.

The *Replication API* provides instruments to secure that each node inside of the *UDDI* has a correct information. It means that the *Replication API* provides operations to propagate changes between the nodes. Each node keeps a *highWaterMark* to detect the changes of records. The *highWaterMark* contains *USN (Update Sequence Number)* of the most recent record change that have been successfully processed. The *USN* is a 63 bits number which never decreases. All data in the registry are tagged by own *USN* which shows their version.

API	Description
do_xxx functions	
do_ping	Checks existence and readiness of node.
get_xxx functions	
get_changeRecords	Returns changed records.
get_highWaterMarks	Returns latest *highWaterMark.*
notify_xxx functions	
notify_changedRecordsAvailable	Notifies about the new available record changes.

Table 4: Replication API Set functions.

Custody Transfer API Set

When the *bussinessEntity* or *tModel* is published it belongs to particular node and particular publisher. It actually means that only the creator has the right to change an entity on particular node where the original record is stored.

Custody Transfer API Set provides methods to change an ownership or carry those records from one node to another.

The transaction is a binary operation between a current publisher and a future publisher. The current publisher initiates transaction by calling *get_transferToken* operation. When *get_transferToken* operation is called the future publisher obtains a *transferToken.* The future publisher must finalize transaction with a valid *transferToken*.

API	Description
get_xxx functions	
get_transferToken	Initiates custody transfer from one node to another.
discard_xxx functions	
discard_transferToken	Discards obtained *transferToken*
transfer_xxx functions	
transfer_entities	Finalizes transfer of custody.
transfer_custody	Transfer custody of entities.

Table 5: Custody Transfer API Set functions.

Subscription API Set

This API can serve as a watchdog of existing records within a registry. The requester can apply for tracking a new, changed or deleted entities bellow:

- *businessEntity*
- *businessService*
- *bindingTemplate*
- *tModel*
- related *businessEntity*
- *publisher Assertion*

The implementation of each API of the *Subscription API Set* is optional, it depends on the node policy.

API	Description
delete_xxx functions	
delete_subscription	Cancels monitoring.
get_xxx functions	
get_subscriptionResults	Returns information about subscribed content within a time period.
get_subscriptions	Returns complete list of existing subscription of a subscriber.
save_xxx functions	
save_subscription	Requests to monitor a content.
notofy_xxx functions	
notify_subscriptionListener	Delivers information to subscribers.

Table 6: Subscription API Set functions.

Value Set API Set

Whenever the data must be saved, *tModels* should be validated by this API Set. The data to validate are determined by a policy of a particular registry. *UDDI* allows third parties to define their own value sets and to control their validation process. The validation could be performed either by *UDDI* or by a remote service.

The API Set supports a caching also. The requester can obtain all valid values of a particular *tModel* by calling *get_allValidValues* operation.

API	Description
get_xxx functions	
get_allValidValues	Returns a set of all valid values
validate_xxx functions	
validate_values	Checks if a value is valid.

Table 7: Value Set API Set functions.

3.2.2 WSDL

WSDL was developed to describe an interface of a particular service provided on Internet without any relation with transporting technologies. The standard was developed by Microsoft, IBM and Ariba and version 1.1 was submitted to W3C and accepted as a note. The note was accepted by another twenty-two companies and it has begun the most supported W3C submission ever. *WSDL* is a W3C Recommendation now.

The *WSDL* document is, in fact, an *XML* document with unified elements defined by *WSDL* definition. The *WSDL* definition describes a service as a set of end-points or ports

where a ports collection defines a service. It is important to have the same *XML Schema* accessible for both a requester and a provider. The requester must know which data in which manner should be passed to a provider and provider must be able to interpret a data.

Structure of WSDL document

WSDL document contains five essential parts:

1. *Types* – Data types contained in the document are defined in this part.
2. *Message* – This part contains carried data. The *message* contains logical part connected with *types*.
3. *Port Type* – The set of abstract operations. The element defines input and output parameters for each *portType* according its transmission primitive.*
4. *Binding* – Defines a particular protocol, data format and message of an appropriate port.
5. *Service* – This element represents set of ports which creates a service.

WSDL document allows composing single documents together with element *import*. Each imported document must be identified by absolute URI.

Element Types

The element *types* contain definitions of relevant data types to the *WSDL* document. *XSD (XML Schema Definition)* schema is prioritized in *WSDL* definition to keep maximum independence on a platform. *XSD* is a recommendation of *World Wide Web Consortium* how to formally describe the elements.

But it is roughly possible that *XSD* schema could cover all data types so *WSDL* allow publishers to define data types by their own. Each data type defined in *types* element must be identified by the unique *name* parameter.

---

* We recognize four transmission primitives: one-way, request-response, solicit-response, notification.

Element Message

This element contains one or more relevant elements to a defined types described in the *types* elements. The set of possible attributes of each element is not finite but three attributes are strictly defined in accordance with *XSD*.

- *name* – unique identifier of an element
- *type* – pointer to a appropriate data type.
- *element* – pointer to a defined element.

Element portTypes

The *portType* describe messages within an operation. The operations are defined in the *portType* element and each operation can have an input, an output and a fault message defined. Each *portType* element must be identified by a unique *name.*

WSDL standard defines four transmission primitives:

- *one-way* – The port of this type receives a message but it does not send a response. It means that such a *portType* has defined only an input *message*.
- *request-response* – This port receives a message and it sends a correlated message back. This *portType* has defined input, output *message* and it can have fault *message* defined optionally. The response does not have to come immediately within the same interaction, like in HTTP protocol, but the response could be sent later.
- *solicit-response* – This port an inverse *request-response* transmission. The port, in fact, asks for a *message*. To this definition is added fault message in the same manner as within the *request-response* primitive.
- *Notification* – Another unidirectional port. It sends messages without any input message.

Element binding

When we have an interface described we need to know how to reach particular

operation. This information is stored inside the element *binding*. Each *portType* has a format of data and a communication protocol described.

The element *binding* contains two mandatory attributes: a *name* and a *type*. An attribute *name* is a unique identifier of the element and a *token* refers to a particular *portType*. The grammar of input, output and error message is defined inside this element as well as information about the flow of the interaction and operations which take place after interaction.

Element service

The *service* element is a container for *port* elements. Each element *service* is identified by the attribute *name*. The element has to be placed on the very end of the *WSDL* document. The relations between ports is described within the *service* element. Two ports could rely in the following manner:

- The ports have no relationship.
- The ports are semantically same. This occurs when a couple of *ports* share the same *portType*
- The ports are in functional relationship.

Each element *port* within an element *service* has to be identified with the attribute *name* and it has to contain attribute *binding* also. The attribute *binding* refers to a particular *binding* definition inside of the *WSDL* document. The element *port* must specify exactly one connection address of an end-point and must not contain any other information about connection except the address.

WSDL and UDDI

> *"Since the service interface represents a reusable definition of a service, it is published in a UDDI registry as a tModel. The service implementation describes instances of a service. Each instance is defined using a WSDL service element. Each service element in a service implementation document is used to publish a UDDI businessService. "* [5]

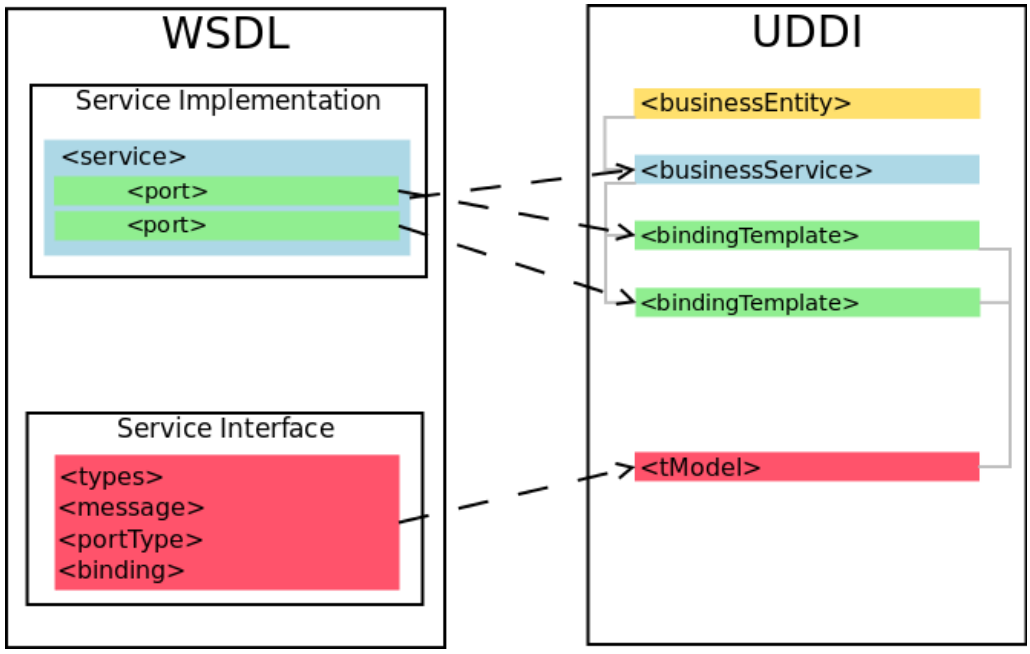

Figure 5: The mapping od WSDL document on UDDI document

WSDL document could be mapped on *UDDI* elements (see the figure 5). The mapping is a whole process where elements and attributes of a *WSDL* document are mapped on elements of *UDDI* document. The important is to check stack overflow in the case of element *wsdl:documentation* because elements are mapped on *uddi:*:description* element which allows to save 256 characters at maximum.

Mapping of *WSDL* elements on *tModel* is the next interesting issue. The definition of a *tModel* does not contain necessary structures to map the *WSDL* literally on *UDDI*. The solution is to pass URI of the information stored in *WSDL* document to a requester. The requester must complete the description by exploring passed URI.

---

5 IBM.com. BRITTENHAM Peter, CUBERA Francisco, EHNEBUSKE Dave, GRAHAM Steve. *Understanding WSDL in a UDDI registry, Part 1.* [online][q. 12th December 2010] Available on WWW: <http://www.ibm.com/developerworks/webservices/library/ws-wsdl/>

The complete mapping is out of the frame of the thesis but the whole mapping of a *WSDL* on *UDDI* is on "http://www.ibm.com/developerworks/webservices/library/ws-wsdl/".

It is important to realize that the good human readability of an *UDDI* document decreases if the document is merged with *WSDL* description. It has to be secured that all the relevant documents are searched because the final document is a composition of the original UDDI and *WSDL* documents.

3.2.3 Description alternatives

The main part of the thesis will is focused on *UDDI* and *WSDL* standards but there are another description instruments which can more or less describe the service as well. The next chapter will describe these minor description languages.

WADL

WADL(Web Application Description Language) is a specialized description language for purposes of *RESTful*[*] applications. The *WADL* definition is currently a submission of *W3C* submitted by Sun Microsystems, Inc., currently maintained by ORACLE. The abstract of a *WADL* definition says:

> *"An increasing number of Web-based enterprises (Google, Yahoo, Amazon, Flickr to name but a few) are developing HTTP-based applications that provide programatic access to their internal data. Typically these applications are described using textual documentation that is sometimes supplemented with more formal specifications such as XML schema for XML-based data formats. WADL is designed to provide a machine process-able description of such HTTP-based Web applications."* [6]

WADL answers the current trend in a HTTP-Based applications development. Such a kind of application does not need as robust description as *WSDL* provides. It is not necessary to define binding precisely because it is sure that the other protocol then HTTP is not expected. The robustness is a feature of *WSDL* which is discussed a lot because it makes a *WSDL* document more difficult to understand either for a human or a machine.

---

* REST is a architecture of distributed applications.

6 W3C. *Web Application Description Service.*[online][q. 18th December 2010] Available on WWW: <http://www.w3.org/Submission/wadl/>

The *WADL* definition covers three use-cases:

- Application Modelling and Visualisation
- Code Generation
- Configuration

The *WADL* document uses *XML* syntax as well as *WSDL* documents. The root element is *application* which contains definition of namespaces and a grammar definition.

The resources of the application are depicted in *resources* element. A resource is, in fact, an operation of a described *application*. Each operation is described by HTTP method.

The comparison between *WSDL* and *WADL* shows that *WADL* is a simplified description of a service for the needs of HTTP-oriented applications. It provides lightweight description which is easily understandable against a robust description of *WSDL*.

OpenSearch

The *OpenSearch*, as previous standards, uses *XML Schema* to describe a particular service. This definition is even more simple then *WADL*. The services described by *OpenSearch* must be reachable by HTTP, as in case of *WADL*, and they must be reachable only by HTTP GET method. The GET method is required because the description of a binding is a URL template.

The root element of an *OpenSearch* document is *OpenSearchDescription*. The description of a binding is reduced to a single tag *Url* which contains two mandatory attributes *type* and *template*. The first one contains a HTTP Content-Type header and the second one contains a URL particular template.

The template of URL is given by *OpenSearch URL Template Sytnax.* Parameters inside of a URL template could be either optional or mandatory.

The optional parameters are marked by the question mark on the end of the name of the parameter. If the parameter is from the extended namespace then the namespace is defined

```
<Url type="application/rss+xml"
xmlns:example="http://site.com/opensearchextensions/1.0/"
template="http://site.com?
query={searchTerms}&amp;amp;color={example:color?}"/>
```

Figure 6: OpeanSearch Url element with both mandatory and optional parameters.

before the name of the parameter (see figure 6). The additional namespaces are defined inside of the *Url* element.

3.3 Transportation

The previous chapters defined the way how to describe a service. The next goal to accomplish is to send data from the one place to another within a network. Several approaches to send a data evolved among service oriented applications. Each approach follows the purpose of an application. *Web services* prefer *Simple Object Access Protocol (SOAP)* to transport data from one place to another. *SOAP* lays on the *Application layer* of the OSI model therefore *SOAP* is independent of transportation protocol.

3.3.1 SOAP

Simple Object Access Protocol (SOAP) is the universal standard for transmission a data from one object to another over a network. This protocol is the main protocol for *Web services;* other protocols consider a *SOAP* message as a preferred incoming message format.* A *SOAP* document is defined by the *XSD* published on W3C website. The actual version of *SOAP* 1.2 was published on 27th April 2007.

The definition of *SOAP* is designed to use various protocols as a carrier of a message. The *SOAP* is independent of any existing transportation protocol. The most common standards used as carriers are HTTP, SMTP, MQSeries and IIOP. *SOAP* is also independent of *XML* versions therefore we can exclude invalidity of *SOAP* message caused by different *XML* version.

* See the quotation of W3C's definition of *Web services* on page 10

A *SOAP* document is divided into two parts: the header and the body. The header contains information for intermediary nodes between a sender and receiver. The message body contains information for both the end-point. The definition describes three different interaction patterns between two systems:

- *one-way*
- *request-response*
- *peer-to-peer conversation*

SOAP message model applies to the one message only, it does not holds for coordination or correlation of group of messages. The advantage of *SOAP* protocol is differentiation of nodes which a message goes through. That means the message can contain instructions for nodes visited during the transmission. This kind of instruction is read in the particular node and then they are not sent any further. Instructions for intermediary nodes have to be put in the head because the body carries an information for ultimate receiver only.

SOAP defines three kind of nodes which a message could goes through (see table 8).

Short name	Full name	Description
next	http://www.w3.org/2003/05/soap-envelope/role/next	Each ultimate and intermediary node of a SOAP conversation must act in this role.
none	http://www.w3.org/2003/05/soap-envelope/role/none	No SOAP node must act in this role.
ultimateReciever	http://www.w3.org/2003/05/soap-envelope/role/ultimateReciever	The ultimate receiver must act in this role.

Table 8: Three SOAP nodes roles.

The roles are identified by URI on the W3C server. Each node has a particular role assigned which must not change during transmission. The goal of roles differentiation is to recognize nodes whose purpose is neither the routing nor creation of semantics of a transmission of a message.

3.3.2 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."* [7]

```
{
"name" : "John Doe",
"age" : 33,
"address" : {
       "street" : "Wall Street",
       "city" : "New York City"
       }
"pets" : [ "dog", "cat", "parrot"]
}
```

Figure 7: JSON collection with a list.

JSON is a text format independent of any protocol or programming language. *JSON* is designed to carry the data between two nodes. A *JSON* message carries nothing else then the data. This is the big difference between *JSON* and *SOAP* message.* *JSON* does not supports a proper types definition. The data validation logic lays on the service it self.

A *JSON* message contains two structures:

- *A collection* of name and value pairs. The collection is implemented to different programming languages as an object, a record, a dictionary, a hash table, an associative array, etc.
- *A list* is a sequence of values. The sequence is implemented to programming languages as an array, a sequence, a vector, etc.

JSON format is the most used data transmission format in *Rich Internet Applications* (*RIA*). The format is highly effective in a code of a JavaScript application because a parsing is not necessary; the message evaluation creates an JavaScript object.

Even if *JSON* is preferred format in JavaScript applications, it can serve as a format for general interaction with any service as well. The main advantage of *JSON* is simplicity. It is easy to read by either human or machine. A *JSON* agent could be light-weighted

7 JSON.org. *Introducing JSON.* [online][q. 13th January 2011] Available on WWW: <http://www.json.org/>

* *SOAP* message can carry either information for intermediary nodes. *JSON* message can not carry such an information.

alternative against a robust *SOAP* agent.

3.4 Related protocols

The core protocols of *Web services* were not enough to fulfil complex needs of business-to-business applications. A several related protocols were designed to provide standard approach to create robust business applications. This chapter will describe protocols to extended addressing, to implement a policy, a security and transactions.

3.4.1 WS-Addressing

The *WS-Addressing* protocol adds extra addressing instruments which are not contained in *SOAP*. *SOAP* provides instruments to execute a simple communication only. This patterns are not enough to identification actors in complex communication which includes many end-points.

The protocol is based on *XML* as other protocols of *Web Services*. It provides complex actors description which is put directly into a message. The protocol supports transactions within a network which contains many different nodes like *endpoint managers*, *firewalls*, *gateways*, etc. The *WS-Addressing* protocol provides two different structures, *EndpointReference* and *MessageInformationHeaders*, which can describe nodes within a network.

Endpoint references

This construction allows endpoints to be used in several different manners. The description of an endpoint is made not only by *URI* but even by parameters and the policy of a particular end-point. The policy is described by *WS-Policy* protocol.

```
<wsa:EndpointReference>
       <wsa:Address>http://site.com/addressing/</wsa:Address>
       <wsa:ReferenceProperties>
              <site:userID>123</site:userID>
       </wsa:ReferenceProperties>
       <wsa:ReferenceParameters>
              <site:userNationality>CZ</site:userNationality>
       </wsa:ReferenceParameters>
       <wsa:PortType>site:UserPortType</wsa:PortType>
</wsa:EndpointReference>
```

Figure 8: WS-Addressing Endpoint reference example.

Message Information Headers

The *message information headers* is another way how to describe an endpoint involved in a message transmission. The header, then, defines endpoints to a family of messages. These endpoint contains information where to send a response, fault message, etc.

```
<S:Enveloper xmlns:S=http://www.w3.org/2003/05/soap-envelope
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
<S:Header>
       <wsa:MessageID>http://site.com/message/12345/</wsa:MessageID>
       <wsa:RelatesTo RelationshipType="...">
       http://site.com/relations/1/
       </wsa:RelatesTo>
       <wsa:To>http://site2.com/reciever/</wsa:To>
       <wsa:Action>http://site.com/reciever/</wsa:Action>
       <wsa:From>endpoint-reference</wsa:From>
       <wsa:ReplyTo>endpoint-reference</wsa:ReplyTo>
       <wsa:FaultTo>endpoint-reference</wsa:FaultTo>
</S:Header>
<S:Body>
...
</S:Body>
</S:Enveloper>
```

Figure 9: WS-Addressing Message Information Headers example.

3.4.2 WS-Policy

"*WS-Policy defines an extensible framework for Web services constraints and conditions on an interaction between multiple Web service endpoints. It is intended to provide a definition for the services to annotate their interface definitions, to describe the policy, ensure their service qualities, and specific policy requirement by a machine-readable expression form containing combinations of individual assertions.* " [8]

WS-Policy extends *WSDL* protocol by transmission reliability and additional security. The *WSDL* provides service function description but that is not enough in business environment. The semantics and composing specification must be described too. However the service policy definition is not supported by *WSDL* specification.

WS-Policy uses *WS-PolicyAttachment* protocol to attach a policy to a particular service. *WS-Policy* describes two different kinds of a metadata:

- *capabilities* – the rights of a service.
- *constraints* – the necessary conditions which must be fulfil to use a service.

The protocol is focused on several policies which are described by other protocols such as: *WS-Security, WS-Transaction,* etc. *WS-Policy* describes how to represent and describe those policies.

8 WANG Yue. *Web Services for a Software Development Platform: Master degree project*. Kalmar / Väjxö: Linnæus University, 17th August 2008., p 14.

3.4.3 WS-Security

The security is a big issue nowadays. Companies must resist attacks targeted to a specific enterprise more often. *WS-Security* creates a secure environment to run *Web Services*. *WS-Security* is based on existing security models - *Kerberos* and *X509*. Aside of them *Web Services* could use even security instruments which lays on the transport layer like *HTTPS* or *BASIC-Auth.* However these secure transportation protocols are not capable to provide a complex security which is necessary during a communication between multiple participants.

WS-Security is a family of standards which provides necessary models to perform a secure complex transmission. The family contains following standards:

- *WS-Security* is used to secure a content of *SOAP* message. The protocol supports authentication of origin, checking integrity and encryption of a message.
- *WS-Trust* supports a *Security Token Service* to verify an identity of a participant of communication.
- *WS-SecureConversation* serves to participants to agree with specific session keys.
- *WS-Federation* uses *WS-Trus* and *WS-SecureConversation* to make a common security definition for *Web Services* conversation with multiple participants.

- *WS-Privacy* declares a privacy policy of a company and it also guarantees that a requester agrees with the policy.
- *WS-Authentication* allows system administrators to declare rights of a particular requester.

3.4.4 WS-Transaction

Transactions are crucial part of a complicated business activities which *Web Services* can perform. The protocol is a composition of three other *WS-** protocols:

- *WS-Coordination*
- *WS-BusinessActivity*
- *WS-AtomicTransaction*

The *WS-Transaction* provides necessary features which any transaction architecture must provides:

- *Atomicity* guarantees that all steps within a transaction are done or no step is done.
- *Consistency* ensures that all participants obtain consistent result.
- *Isolation* makes a transaction undividable; all steps must be done without any interruption or modification from outside.
- *Durability* – An outcome stays maintained after successful transaction is performed.

3.5 Summary

For the description of *Web services* we can see that the term is very fuzzy. The main protocols and technologies are *UDDI, WSDL* and *SOAP.* It was described that these three protocols provides instruments to built a robust solution independent on the transportation protocols, programming languages or platforms. But this robustness seems to be the greatest weakness of the architecture. It was shown that numerous protocols for either description or transportation protocols exist. It is caused that developers very often does

not need to use various protocols to realize a transition so they do not need to create a very complex *SOAP* message.

The main advantage is robustness paradoxically. On the one hand it demands a good knowledge and skill of developers but on the other hand it provides a huge set of instruments how to realize communication between a requester and a server which begins by the searching of a service until the end of communication.

3.6 REST

Aside of the *Web services* architecture is growing another *Service Oriented Architecture (SOA).* The *REST (Representational State Transfer)* is an architecture commonly used by web sites. It is a concept of *SOA* which fits the purposes of web pages more then other robust architectures.

3.6.1 Description

The numerous instruments of *SOAP* standard are not necessary for the purpose of communication between web page and a client. The concept of *REST* simplified the communication to two actors only. *REST* recognize the client, or requester, and an end-point. The client is moving between end-point using *URL*s (*Uniform Resource Locator*).

A key concept in REST architecture is that of the resource. A resource is any object that can be named. A resource may be a document, blog, or search result. [19] The *REST* architecture then distinguishes a resource from its representation. The one resource could be represented either in more then one format.

The example of *RESTful* application is *ATOM Publiching Protocol.* The protocol follows principles of *RESTful* applications. The protocol provides way to create, edit and delete a resource such as newsfeed.

The *WADL* (see Chapter) protocol was designed for purposes of *RESTful* applications. The format of *WADL* provides the easy way to generate *URL* to connect to a service.

The *REST* uses *HTTP* protocol as a transportation channel only. On the other hand it Tabulka3uses all methods provided. It would not be possible to utilize *HTTP* methods as effectively as *REST* do in case of the *Web services*. *REST* does not use any specific

protocol to encode the message which must be delivered to the service, it implements so called *CRUD (Create, Read, Update, Delete)* operations. These operations are based on *HTTP* methods (see figure 10).

Operation	HTTP method
Create	POST
Read	GET
Update	PUT
Delete	DELETE

Figure 10: CRUD operations and their HTTP methods.

REST leaves an open space to add other HTTP methods* but it is preferred to use the four methods mentioned on the figure 10.

3.6.2 Summary

REST is the alternative to the robust architecture of *web services*. It is designed to use the *HTTP* protocol very effectively without necessity of creating rich messages to transport data from a requester to a server.

The architecture is based resources identified by *URL* which are separated from their representation. To distinguish between various queries send to a service *REST* uses *HTTP* methods POST, GET, PUT and DELETE. *REST* defines *CRUD* operations as a core operations which should be built in a *RESTful* application.

The existence of REST underlines the current situation on the Internet where web applications are as important as they can have its own SOA.

* These *HTTP* methods can be: HEAD, TRACE, OPTIONS, CONNECT, PATCH.

4 Collecting of data and analysis of requirements

The goal of the analysis will be to describe the relationship between a supplier and customer. It is necessary to understand its special qualities to design a good working architecture which will be trustworthy to the customers and profitable for suppliers. When we design a system when the supplier is not known literally to a customer it is necessary to understand which guarantees must be given to a customer. On the other hand there is a supplier which does not know how many pieces would be ordered.

4.1 Questionnaire

The questionnaire is aimed to get a general view on the relationship between a company and a supplier. The aim is to find out how big part plays the personal contact between both sides. The results of will be used to realize general threats which are present in the interaction between businesses. The questionnaire was sent by email to approximately three thousands of emails and 234 companies answered.

The result will be extended by the results of interviews to describe a potential environment of business-to-business applications deeply.

4.1.1 Questions

1. Do you have an IT department in your company? (multiple choice)
2. Does your company use a business-to-business applications? (multiple choice)
3. How important to gain a supplier is:
    1. Personal contact
    2. References
    3. Speed of delivery
    4. Geographical location
    5. Flexibility of delivered amount.
4. Would you trust a supplier which you don't know personally?

5. (Why the personal contact is so important?)*
6. (Would you trust to a supplier if the authority vouch for him?)*
7. Number of employees?
8. Branch of business?
9. Do you have interests abroad?
10. Additional information.

4.1.2 Results

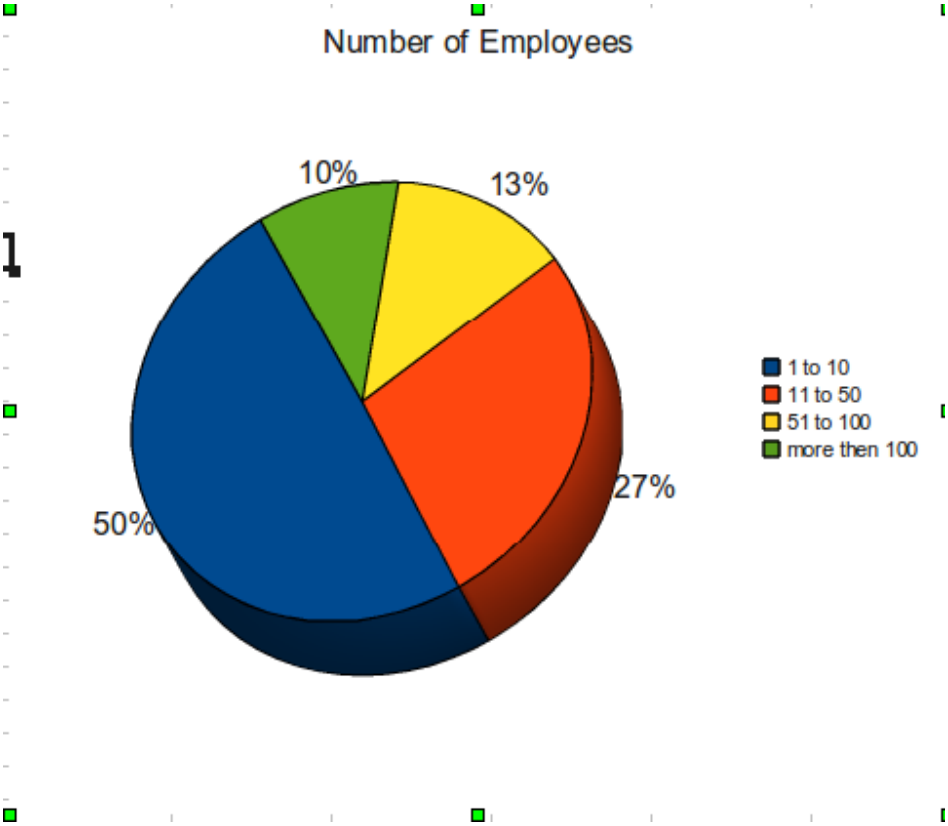

Figure 11: Proportion of the sample according number of people employed in a company.

The research will work with the sample containing 234 items. The proportion of the sample according number of people employed (Figure 11) shows the majority of small companies with at most ten employees.

* The question is answered only if the question number 4 is negative.
* The question is answered only if the question number 4 is negative.

The interesting is fact that even if the sample contains a lot of small companies, the companies exist on both domestic and foreign market. This could be the reason why the choice of supplier does not depend on the personal contact often. The 84% of respondents answered they do trust to supplier who would not be contacted personally. The last part of a research is to compare qualities of supplier between each other according the preference of the companies.

The whole analysis will consider the probability of the type-1 error 5% ($\alpha = 0.05$). The method used is non-parametric ANOVA because the sample of preferences does not have a normal distribution. The p-value of Shapiro-Wilk is much lower then α so the null hypothesis is declined and we accepted that the sample does not have normal distribution (see figure 12).

Tests for Normality				
Test	Statistic		p Value	
Shapiro-Wilk	W	0.844276	Pr < W	<0.0001
Kolmogorov-Smirnov	D	0.216652	Pr > D	<0.0100
Cramer-von Mises	W-Sq	9.508567	Pr > W-Sq	<0.0050
Anderson-Darling	A-Sq	64.26059	Pr > A-Sq	<0.0050

Figure 12: The test of normality of the sample of preferences of the companies.

The results of ANOVA would bring us the information about preferences of the companies. The analysis of variances proved that there is significant difference between qualities of suppliers (see figure 13). We must explore the qualities deeply to consider the preferences in following design of the architecture.

Wilcoxon Scores (Rank Sums) for Variable Importance Classified by Variable Question					
Question	N	Sum of Scores	Expected Under H0	Std Dev Under H0	Mean Score
PersonalContact	224	112443.00	125888.0	4165.29186	501.977679
References	227	135592.00	127574.0	4186.08956	597.321586
SpeedOfDelivery	232	167063.50	130384.0	4220.11637	720.101293
Locality	211	72658.50	118582.0	4071.74179	344.353081
Flexibility	229	143369.00	128698.0	4199.79491	626.065502
Average scores were used for ties.					

Kruskal-Wallis Test	
Chi-Square	184.2254
DF	4
Pr > Chi-Square	<.0001

Figure 13: Non-parametric ANOVA - Wilcoxon Scores of Quality

The Tukey's HSD test is suitable to research differences between classes of ANOVA test. The score of the each class we can see on the figure 14. The test proved following order of qualities according preferences of the companies:

1. Speed of delivery,
2. references, flexibility,
3. personal contact,
4. locality.

The questionnaire includes opened questions also. A company could note additional information or explanation of the relation between them and their suppliers. The respondents pointed out several issues as the legal issues of relationship between a company a supplier.

The next benefit of the personal contact is the co-operation of a supplier on the business solution. The supplier can be included in the project planning and design. It could remove the future problems with compatibility of several parts.

The next remarked problem is the normalization problem. The companies could have their own specific norms for machines or material they use. This could be governed by internal decision or by additional certification. These standards could be even more strict then national norms or standards*. The description of such norms is very complicated to exclude the personal contact form the business relationship.

* For example ISO or ČSN

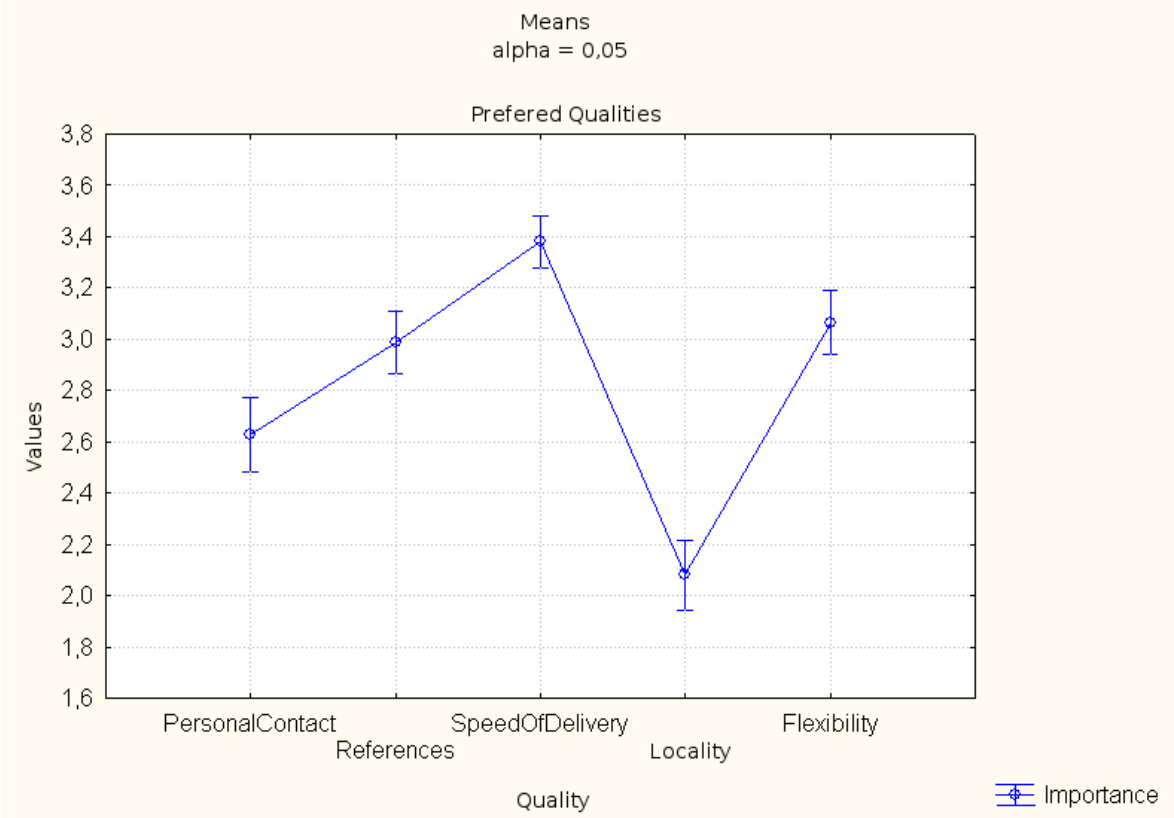

Figure 14: Tukey's HSD test.

4.2 Interview

The main question:

"Is it possible to a company to buy a material or goods from unknown supplier?"

4.2.1 ČEZ a. s.

ČEZ a. s. is the Czech company in the energetics with 32.000 employees. The company run almost all kinds of power plants within a Central and Eastern Europe.

The interviewed employee works as: *Director of Quality and Management Systems.*

The energetics business and especially nuclear energy is the branch regulated by a government a lot because accidents could have fatal consequences as we saw in Chernobyl in 1986. This is the reason why, for example, the management of quality is demanded by a law to all companies which run nuclear plant in the Czech Republic. The nuclear energy has the very similar quality demands as air-planes or in astronautics.

All crucial suppliers of ČEZ a. s. must follow its quality policy. The material is split to

several categories which differs by the conditions which must be fulfil. The restrictions and proves of quality increases according the level of threat which could occur if the part was broken. The hardest restrictions must obey suppliers of parts used to maintain and run nuclear plants. The quality of parts must be proven by a certificates and precisely documented manufacturing process. The supplier must be well known and guaranteed by auditors therefore there is no space for anonymous suppliers. The company could consider only office stuff which could supplied by an anonymous distributors.

4.3 Conclusion

The analysis brought an information which supports an idea of a electronic distributed system. The companies, a side of price which we consider implicitly as the most important factor, prefer the speed of delivery and flexibility as the most important qualities of suppliers. An information system would provide them the most fast delivery possible.

On the other side a weakness of a system are references. An architecture of such an information system can hardly provide references. The possible solution is to establish specialized services which attach references to services.

The special requirements of different industries are the next weakness of the electronic relationship between a company and its suppliers. Rules given by a law which could be hardly possible to fulfil by an impersonal interaction. The fatal barriers are present in dangerous businesses such as nuclear plants where conditions of relationship between a company and suppliers is given by a law.

5 Application domain

The following chapters will describe an architecture of *Open Distributed Ordering System (ODOS)* which will be able to describe meta data of services and it should provide necessary data to connect and consume service automatically.

Such an architecture must be able to provide all other information to run a regular Web Service. The research showed the luck of trust between companies and anonymous service providers. The vital to an architecture is to introduce mechanism to get a trust of customers.

5.1 General purpose of the system

The main goal of a thesis is to describe how to search and recognize web services according the goods or service they provide. The system assumes a consumer which demands a product and a service supplier. The first kind of users, consumers, is decided to consume a service but the provider does not matter. Then the system should search services to find appropriate set of relevant results according a requester's demand . Supplier are the second kind of users which can provide whatever they want. The necessary is to describe a product precisely to be searched properly.

5.1.1 Problem of a consumer

The first problem of a consumer is the uncertainty where he should send a request to obtain asked result. This issue is common for ordinary Internet browsing. The problem of services is very similar. The services must create an universe where a machine could orient properly and where the machine is able to interpret the information.

The second problem is to provide guarantees to a consumer that the service he consumes gives him the result he expects. This is the main difference between the ordinary web searching and the searching of services because the common Internet browsing does not provide guarantees about the content of a web page.

5.1.2 Problem of a supplier

The supplier offers a product. His main goal is to advertise and sell product to a consumer. He should be focused to produce his product and to be just requested to sell the product. The main problem is how to identify the product and how to declare certainty about the consumer that he will get what he asked for.

5.1.3 Problem of tasks

This is important part of the application domain. What does the system may generally serve for? The system is considered to provide goods and services which must be in the right time on the right place. In fact this will lead to composing services to complex transactions which must be accomplished to consume the final service.

6 Analysis and design of the solution

The chapter will be split to the two parts. The first part will analyse the description of *UDDI* as the place to deploy necessary description. The important issue is going to be the semantic correctness of deployment. The next part of the analysis is the effective description of goods. Then threats of the searching algorithms will be described.

The second part of the chapter is design of the solution. The design will contain the concrete grammar of the description and the algorithm of searching.

6.1 Analysis

The analysis first consider the use cases of *ODOS.* The system includes three types of actors. The first two are obvious from the purpose of the system. *Supplier* and *Consumer* are those who the system serves to. They request an interoperability which is reached by the proper service description. They also need to find what they want in reasonable time.

6.1.1 The services filtering

A Consumer solves a described *problem of a consumer* (see the chapter 5.1.1). *A Consumer* must search the set of products description. It is crucial for *a consumer* to distinguish similar services. The products must be classified properly but the similarities between products must be preserved. The assumption holds that one product must be declared in the same way across numerous suppliers. *Consumer* then will be able to find the best transaction for his purpose.

6.1.2 The services publishing

The suppliers would be able to publish every service that they want. They must describe an interface and effect of a service precisely. The interface description has to be machine-readable so the whole transaction will be able to be performed without the human intervention.

The service, as it was mentioned in previous chapter, has an impact in the real world. This impact is local or global. It depends whether it can be consumed all around the globe

or if the product is available only on a specific place in the moment of the order. The order of goods has a local impact because in the moment of purchase the good is available in a warehouse and, on the other hand, the cashless payment has a global impact because it can be consumed everywhere on the planet.

The process of publishing does not lay on the side of a *supplier* only because the effect of service must be guaranteed. The whole system must have a supervisor or supervisors who could vouch for published services. The guarantees are requested by the *consumers*.

6.1.3 The service consumption

The service consumption is not as simple as it looks like. The services could be consumed only in the particular area. The goods should be shipped somewhere else from the warehouse. The whole order could be composed of more services. The first step could be to send an order to a warehouse but then the physical goods must be moved from one place to another. This could be realized by a transportation company, in fact another service.

The consumption of a service is then the process of composition of services to complex transactions.

6.2 Design

The next chapter will design a products recognition system. It assumes that system could contain unlimited number of products and it has to keep the product filtering as effective as possible. The time complexity of searching algorithms must be explored to allow searching system to get a result in reasonable time. The memory complexity is not relevant because the unlimited number of items need unlimited memory regardless the complexity.

6.2.1 Use cases

The design is based on requirements formulated in the conclusion of the research. The system will contain three actors whose are described bellow.

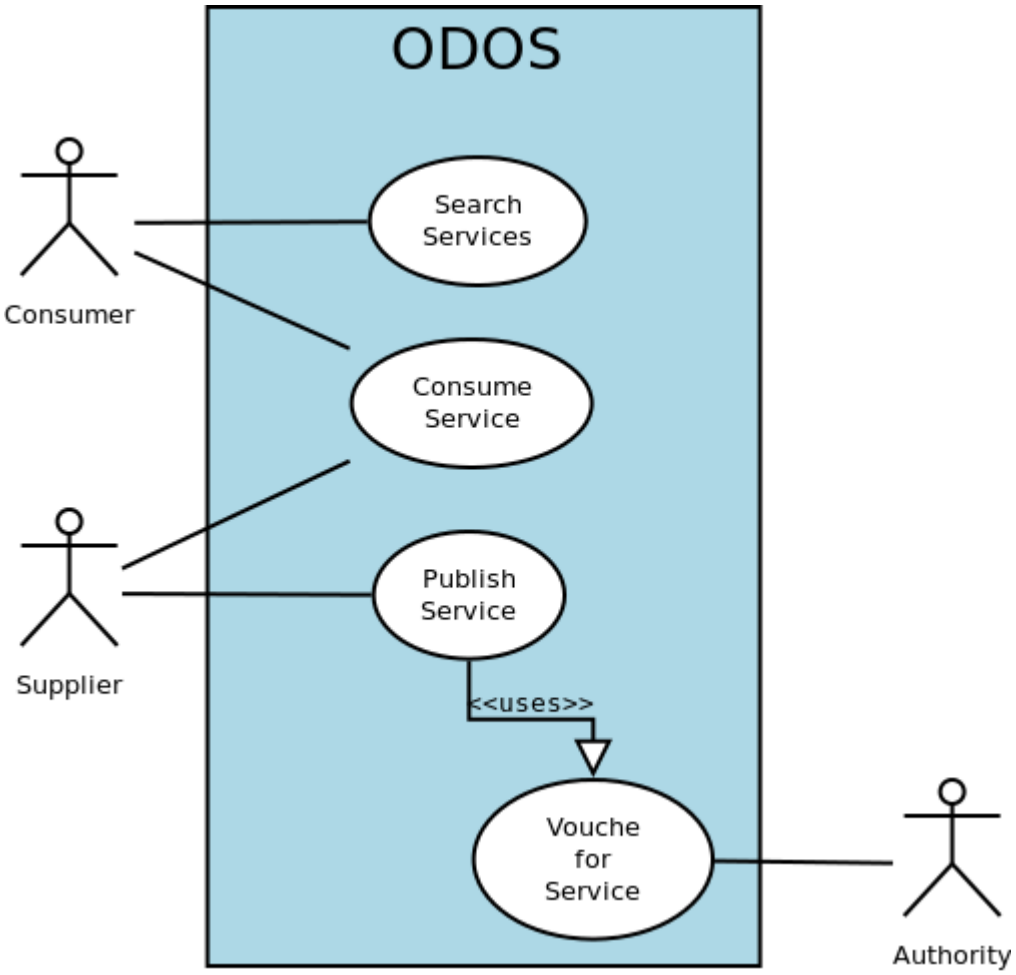

Figure 15: The use case diagram of Open Distributed Ordering System (ODOS)

Actor	Description
Consumer	A consumer is the person or machine with unsatisfied need. He acts as an initiator of a transaction.
Supplier	A supplier offers a service to the public. He waits for the demand and then performs a service if he is capable.
Authority	An authority declares that the service could be trusted.

Figure 16 The description of actors.

The search-engines in *ODOS* does not exists. The only place to search services are registries which provide descriptions of services. The system consider the correct result as each service which fulfils the conditions with the lowest price. The conditions are:

1. right authority
2. right service
3. right location

The electronic relations between a company and its suppliers, according the research (see chapter 4), struggles a lot by the lack of trust. The authority is an external which guarantees that an effect of a service is the same as a supplier declares. The authority must satisfy the natural scepticism of business entities by a transparent service certification. The system allows to provide different levels of certifications. A service without a guarantee given by an authority must not occur in the system. The consumers, then, could choose trusted authorities and services without guarantees given by trusted authorities will be ignored.

6.2.2 The description of services

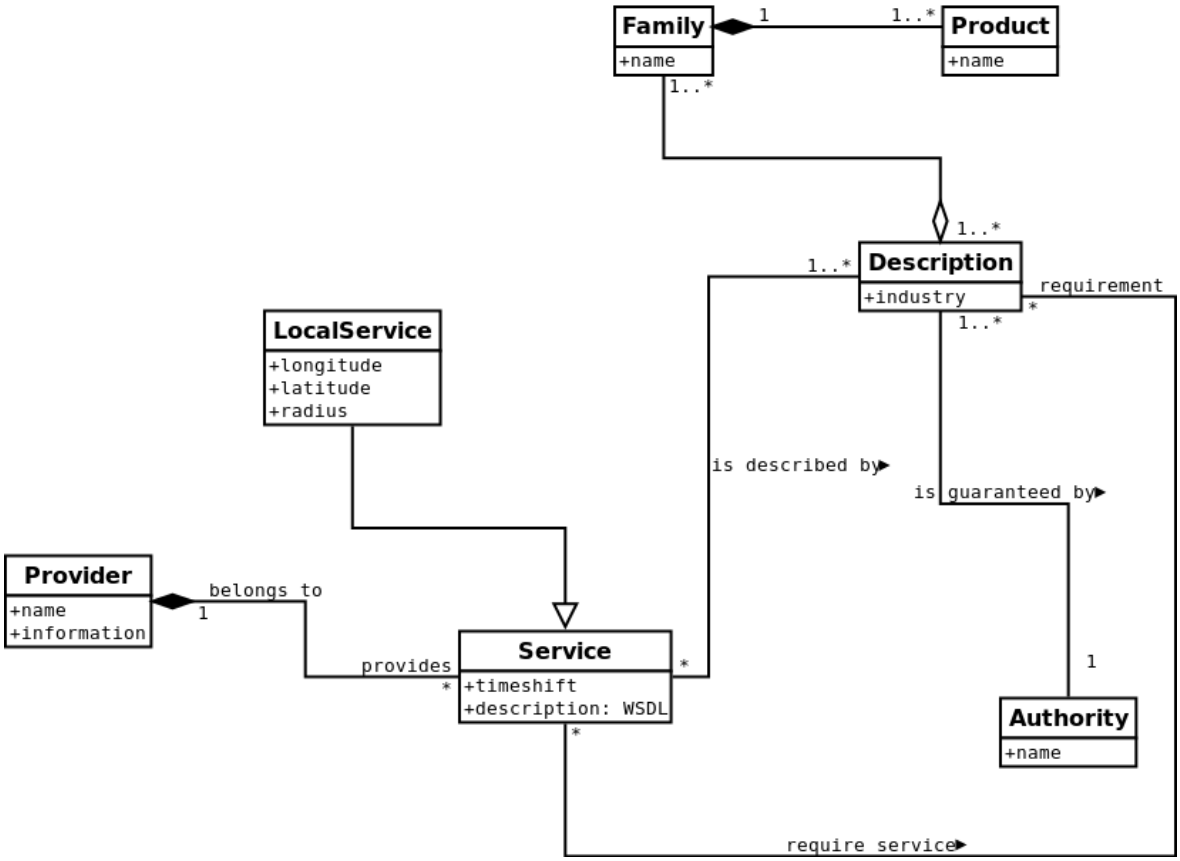

Figure 17: Description of a service.

The services description within *ODOS* is composed from several information

containers. We use an *UML* class diagram to describe the relationships between the service description containers (see figure 17). All of these elements' properties must be accessible by a *consumer*. We consider classes without methods because the diagram reflects the description of a service only.

Provider

The provider is a business entity which is responsible for a service. He provides warranties for a final product according a law. The provider must be recognizable for a *consumer* and must provide all legal information necessary to consume a service.

Provider	
Attribute	**Description**
name	The name is a name of business entity. The name must be unique to recognize providers.
information	The information contains all the relevant data necessary to perform transaction. It could be identification number, address, postcode, etc.

Authority

The authority is a trusted sign given to a service by an external entity. The authority is the part of a service description which is trusted by a *consumer*.

Authortity	
Attribute	**Description**
name	A unique name of a authority.

Description

The description class is a description of a service result. A *consumer* search services according a products they offer as first step and then other filters could be applied. The structure of a description must be searchable effectively, that means the number of steps to explore the entire system must be as low as possible.

The product description has to be guaranteed by an authority. A product with no authority associated must not occur within a system. A one product could have more then one description; the description could be approved by more then one authority.

Description	
Attribute	**Description**
industry	The description of industry which a product belongs to.

Family

The family is second level searched when the particular product is demanded. It represents a set of products with common purpose. The family could belong to one or more industries and therefore they could implement additional logic to a description; the products from different industries could share the same family.

Family	
Attribute	**Description**
Name	A name of a family of products. The name is unique within an industry.

Product

Final part of a description. The product identifies precisely a good or a service within a family.

Product	
Attribute	**Description**
Name	A unique name within a family.

Service

The general service. The *consumer* do not have to take care about position. The effect is global. The effect of a service is described by a *description*. The service himself is technically described by *WSDL* which is out of the frame of this thesis. It is mentioned to create a complete image about a service.

The description of an effect of a *service* is excluded from the technical description. Each service have to be owned by a *provider*. The service performing could depend on another services. The example could be a purchase of a good which definitively relies on a payment which could be represented by another service.

Service	
Attribute	**Description**
timeshift	The time shift between a request and a service realization. Services could be

Service	
Attribute	**Description**
	prompt or delayed.
description	The description represents a technical description of a service. The system consider a WSDL description but it is not obligatory.

6.3 The product description

The product description is decomposed to the three main parts. This decomposition is is only the general overview of classification of product. The important is the inner logic of the parts. The description will generate the connected graph where the nodes of the graph are products and edges are relations.

John Wilkins showed how to create such a classification in 17th century in his work *An Essay towards a Real Character and a Philosophical Language* where he designed a language where the words are written according their meaning. The words were divided from the more general meaning to the very concrete meaning. Such a structure allow users to search a particular product or similar products which he could be interested in.

The three classes described before could be seen as containers for other information. The figure18 shows that each class could be decomposed to other logical groups. The decomposition could be infinite and it depends only on the level of recognition of a particular authority which publishes and approves a description.

The relevance of the search results is given by a consumer who could define maximal distance *d* from the searched entity. If *d = 0* then only particular product is returned therefore the higher the *d* is the wider area of products is returned. The maximal *d* depends on the recognition level of a description.*

* If we take figure 18 as an example of a code and we say that we search a particular basketball shoes then if *d* = 1 we obtain as a result all services where we can buy various basketball shoes, if *d* = 2 we all the indoor shoes, etc.

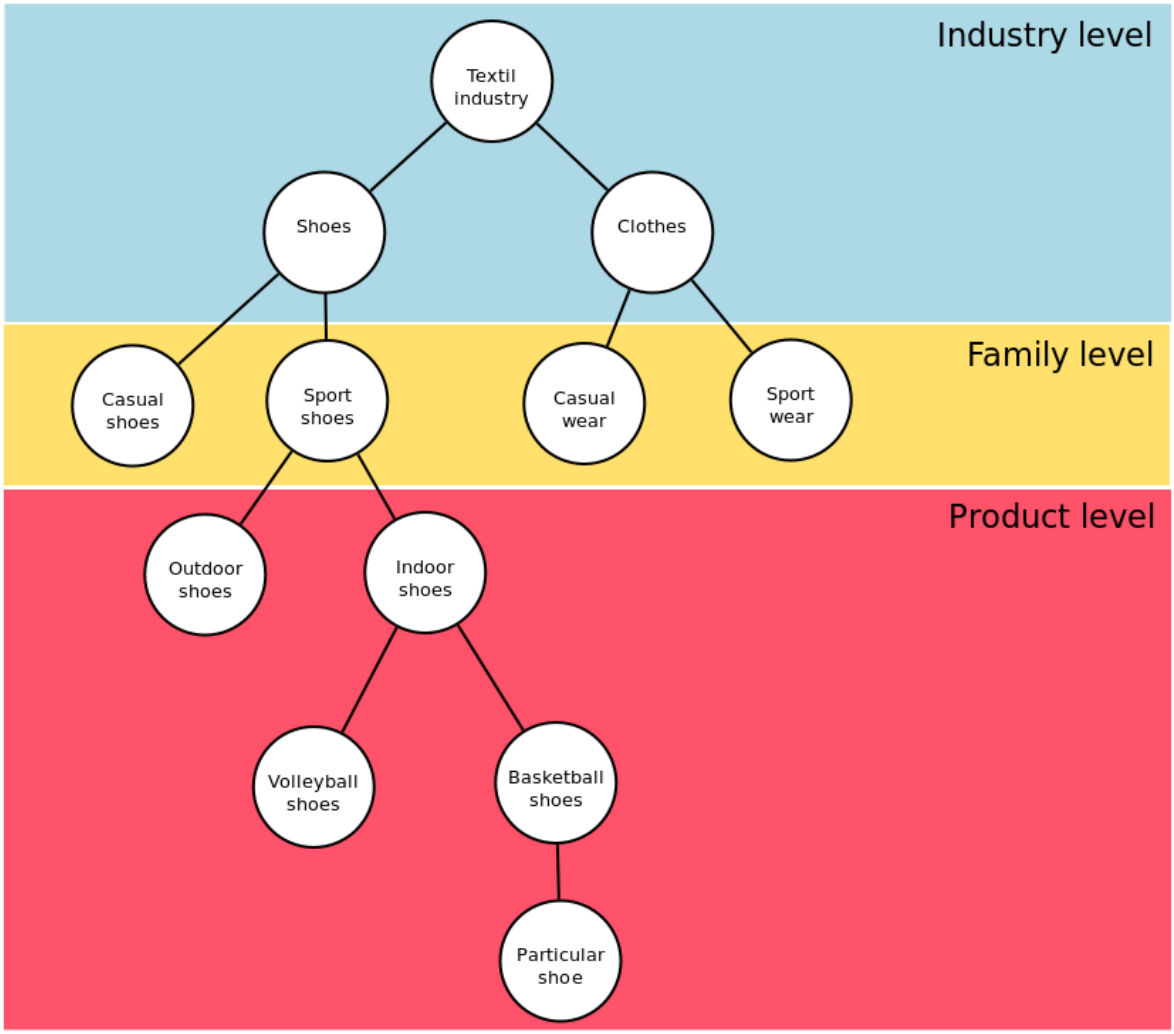

Figure 18: A product description

6.4 The mapping of ODOS to UDDI

The easy way to implement the system is to use existing standards or systems to map *ODOS* to *UDDI.* it fits semantically because the description of a service is not technical except the description of a service interface. However the technical description is separated properly from the other information so we could split it out of *UDDI*.

The goal is to project the *ODOS* architecture semantically to *UDDI* standard to keep the usability of *UDDI* and contain information from *ODOS*.

The information stored in the *UDDI* structure will be available via *UDDI API Sets*. These communication methods are defined in the description of *UDDI* therefore no additional API has to be defined.

The information necessary to describe a service will be mapped to a *businessEntity* and

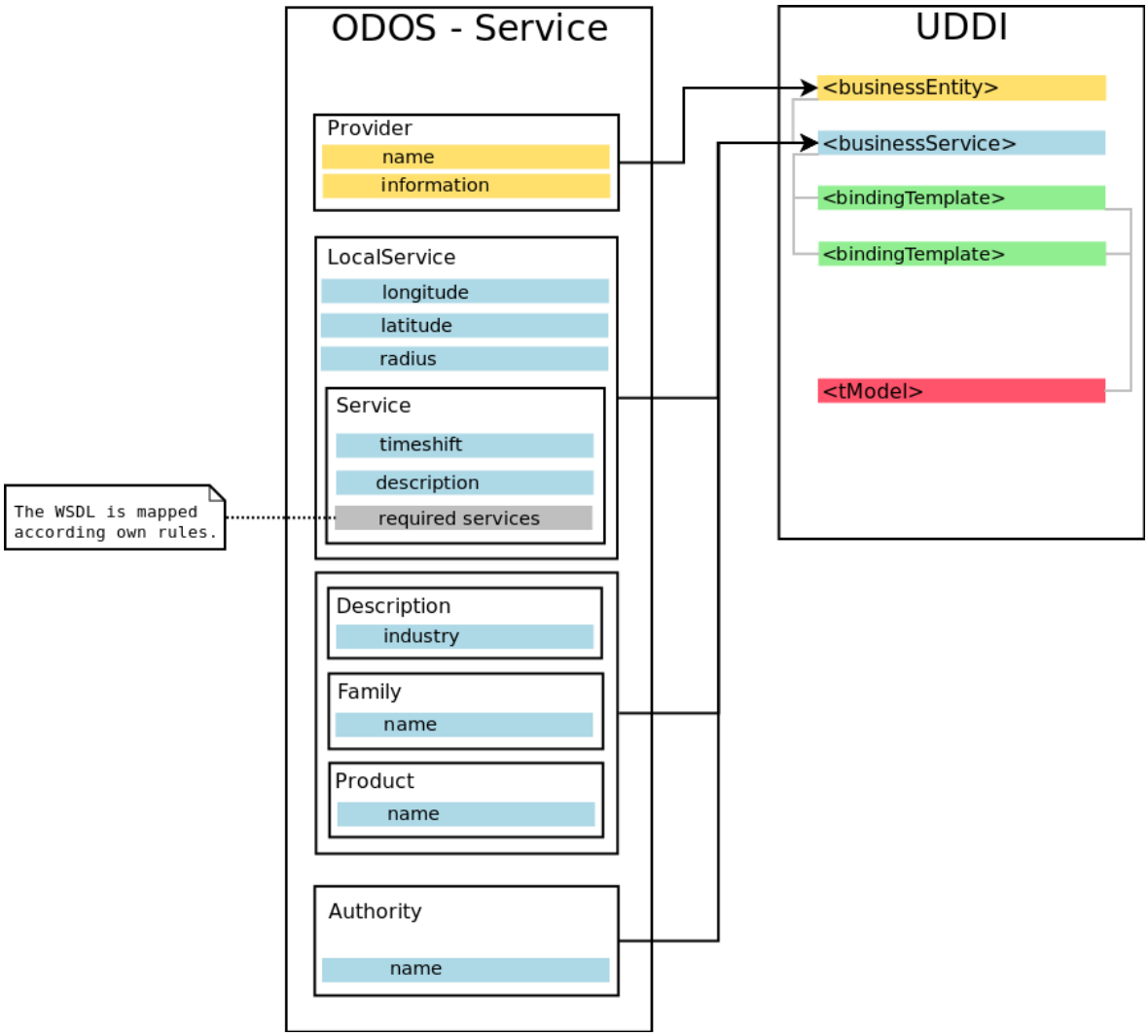

Figure 19: ODOS mapping on UDDI.

to a *businessService* of a particular *businessEntity*. We will use existing element *categoryBag* which is optional for both *businessService* and *businessEntity*. To ensure that current data will not be overwrote we will use special *keyedReferenceGroup*. The category bags allow requesters to filter services according the criteria contained in a *categoryBag*.

6.5 Mapping on businessEntity

The business entity is a top level of a service. The information about a provider will be mapped here. The information will be obtained after the call of *get_businessDetail* from *Inquiry API Sets.* The customer then will be able to recognize all the information about a company which the company provides.

The information demanded by *ODOS* will be mapped to the current *IDDI* structures. There is no necessity to define a new data-structures within a current standard.

6.6 Mapping on businessService

The searching of products requires effective access to *businessService* without the knowledge of the related *businessEntity* which is required nowadays. The problem of the searching of services is definitely the complexity. We can assume millions of services within one registry. The composite description that the thesis described before splits the description to the levels what decreases the number of steps to filter products.

The set of filtered services could be infinite so the additional logic must be added to return a reasonable number of results. This logic should not be unified so different registries could provide different logic which will determine filtered results. This differentiation will lead to the diversity of results which may lead a consumer to chose a registry according the logic it provides. For example, the search results could be ordered by the frequency of purchases, the limit of filtered services is set up to twenty. When services fulfil all conditions required by a consumer, then only twenty the most frequently used services are returned.

The *categoryBag* structure will be used to store the service description. Three *keyedReferenceGroup* will be defined within a *categoryBag*. The fact that *categoryBags* could be searched by *find_service* operation from the *Inquiry API Sets* is important.

- *LocalService* – the group defines all attributes of the *localService* whose are described above. This *keyReferencedGroup* is optional. It depends on the service's character.
- *Service* – the group is obligatory for all services in the *ODOS* system. It contains common information about a particular service.
- *Authority* – at least one authority group is obligatory for each service. The group is identified by the name of the authority which guarantees the description. The group defines all parts of a description of a product.

```
<categoryBag>
<keyedReferenceGroup tModelKey=”uddi:odos:localService”>
       <keyedReference
              tModelKey=”uddi:odos:localService:longitude”
              keyName=”Longitude”
              keyValue=”+48.009”/>
       <keyedReference
              tModelKey=”uddi:odos:localService:latitude”
              keyName=”Latitude”
              keyValue=”+8.0002”/>
       <keyReference
              tModelKey=”uddi:odos:localService:radius”
              keyName=”Radius”
              keyValue=”100m”/><!-- meters —->
</keyedReferenceGroup>
<keyedReferenceGroup tModelKey=”uddi:odos:service”>
       <keyedReference
              tModelKey=”uddi:odos:service:timeshift”
              keyName=”Timeshift”
              keyValue=”20d”/> <!-- in days -->
</keyedReferenceGroup>
<keyedReferenceGroup tModelKey=”uddi:odos:authority:ISO”>
       <keyedReference
              tModelKey=”uddi:odos:authority:ISO:industry”
              keyName=”Industry”
              keyValye=”044-DADDC893123-9099”/>
       <keyedReference
              tModelKey=”uddi:odos:authority:ISO:family”
              keyName=”Family”
              keyValye=”0001-FA988778088-237892AF”/>
       <keyReference
              tModelKey=”uddi:odos:authority:ISO:product”
              keyName=”Product”
              keyValue=”FA67-0011056FCDD-245679A0”/>
</keyedReferenceGroup>
<keyedReferenceGroup tModelKey=”uddi:odos:authority:EU>
       <keyedReference
              tModelKey=”uddi:odos:authority:EU:industry”
              keyName=”Industry”
              keyValye=”DAS678K-MNIO2213K0-DMAL99”/>
       <keyedReference
              tModelKey=”uddi:odos:authority:EU:family”
              keyName=”Family”
              keyValye=”MMIOJ99-LMLMO8978-MKLOOO0”/>
       <keyReference
              tModelKey=”uddi:odos:authority:EU:product”
              keyName=”Product”
              keyValue=”NUUO88J81-KP00I012MM-0LKKLN1”/>
</keyedReferenceGroup>
</categoryBag>
```

Figure 20: Example of the categoryBag of a service

7 Summary and discussion

The data collected by this research described a supplier of a company as a flexible partner. The necessity of personal contact is not of major importance for the inquired businesses. Speed of delivery* is considered by the contacted companies as the most important attribute.

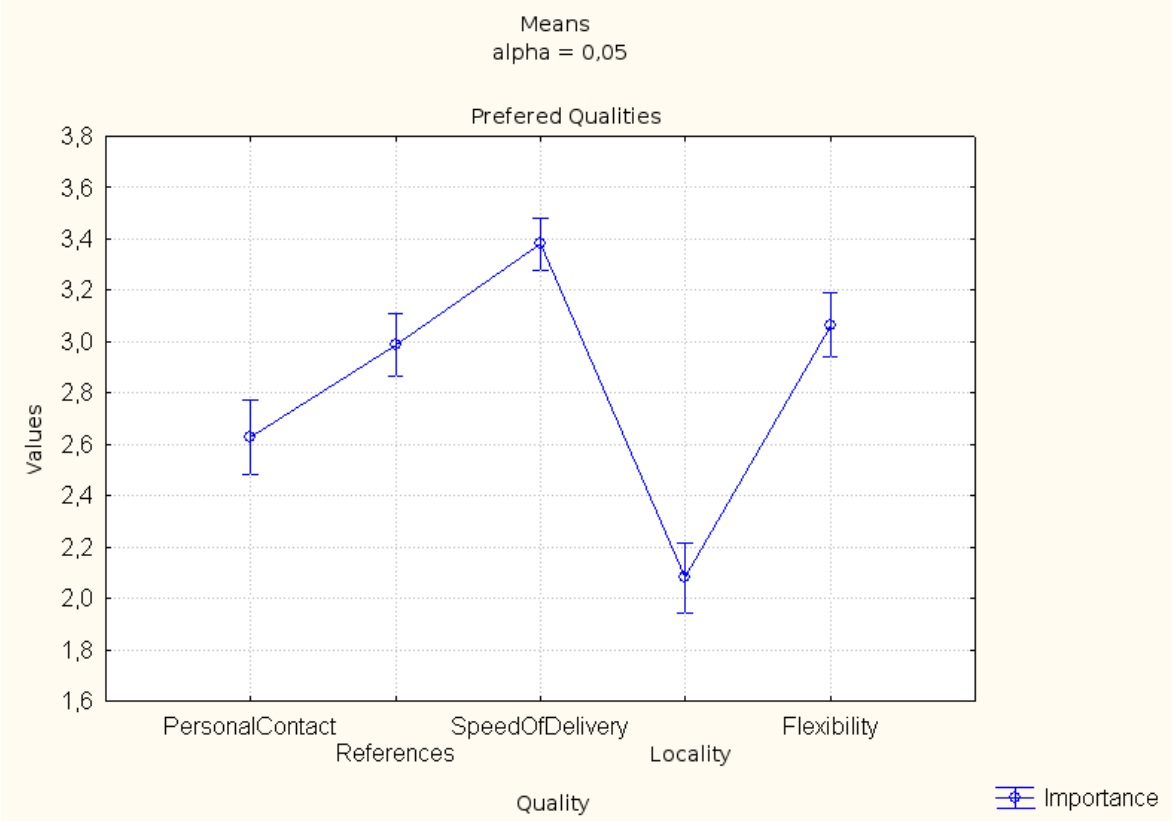

Figure 21: Tukey's HSD test.

Information systems could be the way how to satisfy the company's demand for flexible and quick delivery of materials or goods. The system will be able to find and order the goods effectively all around the globe. The service's description brought by the designed architecture provides sufficient information to search and compose services for more complex transactions where goods could be found, paid and shipped. All the processes are done by providers who fit the best the given criteria. The providers do not have to implement their own payment and logistic mechanism as they can rely on services

* See chapter 4.1.2

provided by third parties. The composition of different services will be automatic.

The descriptions of a product will create a universe where all services are grouped to families and industries according common qualities. Each part of the description could introduce its own logic to create a richer environment with a more precise product recognition level.

The current Web Services architecture provides all the necessary instruments to implement the designed architecture. The services would be searched via *find_service* operation from *Inquiry API Sets*. The service description structure design allows descriptions brokers to keep all the products in easily searchable trees.

The design introduces third party certification authorities which provide accurate descriptions and certify their authenticity. The authority will decide about the level of recognition which is implemented into three main parts described in this thesis. The services are grouped together by the authorities; this leads to the differentiation of services according to authorities which guarantees for the service description. Each service can be guaranteed by as many authorities as is needed in accordance with its purpose. The certification will help companies to impose particular search criteria according to their service quality requirements and apply for trusted services.

The issue itself is the business motivation why to access the system. The problem lies with the fact that the proposed architecture does not allow the prioritisation of services by standard marketing and advertisement mechanisms. The system appears because of this as a perfectly competitive market which is very effective for a demander but there is narrow margin of profitability for the suppliers. On the other hand, the system provides a big opportunity for oligopolistic markets. Since *ODOS* does not consider brand and current market share as objective criteria for its search algorithms, it can help small local entities to reach their customers thanks to the affinity of their proposed services with the applied customer demand.

8 Conclusion

The goals of the thesis were achieved. The product description architecture in the Web Services environment was designed.

The described architecture defines a structure where the consumer can find the provider of a demanded service. Services can be composed together to make a service chain which covers all the necessary activities from the order of a good to its delivery, everything is hidden in the cloud where only the effect is known and the execution is not important. This cloud system generates an electronic market which behaves nearly as a perfect competition.

The product description is structured so to make filtering and grouping of similar products easier. The thesis suggests to split a description into the three main classes:

- *Industry*
- *Family*
- *Product*

The differentiation of these three parts within the description of a product creates an universe where similar words are semantic neighbours. The differentiation within the classes is unlimited.

The research showed that the speed of delivery or the flexibility of a service is a priority for the analysed enterprises. Information systems can help a company to find the cheapest or fastest delivery for the demanded goods. This helps the market, generated by suggested cloud, then, becomes more open and effective for the companies. The implementation of IT technologies decreases error occurrence also*.

The system brings benefits to the suppliers too. The dominance of big companies is an obstacle for competitors to access existing markets. The described architecture creates a market which does not make difference between big and small companies, it is all the times impartial and is very close to a *perfect competition* where the price are the only

* According the research of GS1 Czech Republic.

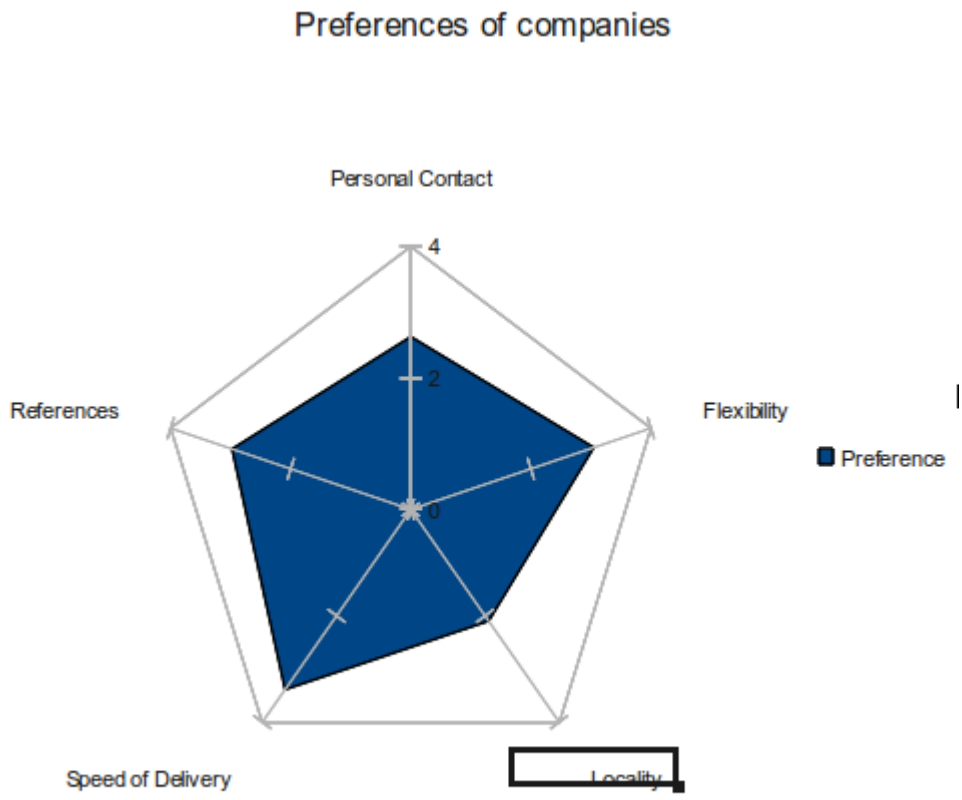

Figure 22: Important factors in the relationship between a company and its suppliers.

competitive advantage.

The research also brought up the trust issue. The system must guarantee that the services are described properly and that they have the effect they declare. The proposed architecture implements a system of authorities that guarantees the description of a service. The authorities system also allows to introduce different quality groups of products. The consumer can define which authorities he trusts and can consider only search results approved by the selected authorities.

The ODOS architecture defines the system of description of *Web Services* which is platform independent, trustful and effective.

9 Sources

[1]: ZIMMERMANN Olaf, TOMLINSON Mark, PEUSER Stefan. Perspectives on web services. Springer-Verlag Berlin Heidelberg, 2003. ISBN: 3-540-00914-0

[2]: ALONSO Gustavo, CASATI Fabio, KUNO Harumi, MACHIRAJU Vijaj. Web Services. Springer-Verlag Berlin Heidelberg, 2004. ISBN 3-540-44008-9

[3]: CARDOSO Jorge, SHETH Amit. Semantic web services and web process composition: first international workshop. Berlin: Springler, 2005. ISBN 3-540-24328-3.

[4]: CERAMI Ethan. Web services essentials. Beijing: O' Reilly, 2002. ISBN 0-596-00224-6.

[5]: W3C. *Web Services Description Service (WSDL) 1.1*. [online] Available on WWW: <HTTP://www.w3c.org/TR/wsdl>

[6]: OASIS. UDDI Version 3.0.2. [online] Available on WWW: <http://www.uddi.org/pubs/uddi_v3.htm>

[7]: PIJANOWSKI Keith. *The UDDI API sets* [online]. 10th June 2008. Available on WWW: <http://www.keithpij.com/Home/tabid/36/EntryID/16/Default.aspx>

[8]: JAVABEAT. *UDDI Publish API.* [online] Available on WWW: <http://www.javabeat.net/tutorials/183-uddi-publish-api.html>

[9]: OASIS. *UDDI Version 2.03 Replication Specification*. [online] Available on WWW: <http://www.uddi.org/pubs/Replication-V2.03-Published-20020719.pdf>

[10]: SAP NEW DESK. *SEO World Magazine.* [online]. 18th December 2005. Available on WWW: <http://soa.sys-con.com/node/164624>

[11]: MICROSOFT. *UDDI Shutdown FAQ*. [online] Available on WWW: <http://uddi.microsoft.com/about/FAQshutdown.htm>

[12]: NEWCOMER Eric. Understanding Web Services. Boston: Pearson Education,

Inc., 2002. ISBN 0-201-75081-3.

[13]: IBM.com. BRITTENHAM Peter, CUBERA Francisco, EHNEBUSKE Dave, GRAHAM Steve. *Understanding WSDL in a UDDI registry, Part 1.* [online] Available on WWW: <http://www.ibm.com/developerworks/webservices/library/ws-wsdl/>

[14]: W3C. *Web Application Description Language.* [online] Available on WWW: <http://www.w3.org/Submission/wadl/>

[15]: OPEANSEARCH.ORG. *OpenSearch 1.1 Draft 4.* [online] Available on WWW: <http://www.opensearch.org/Specifications/OpenSearch/1.1>

[16]: W3C. *Extensible Markup Language (XML) 1.1.* [online] Availabel on WWW: <http://www.w3.org/TR/2004/REC-xml11-20040204/>

[17]: W3C. *XML Schema Part 2: Datatypes Second Edition.* [online] 28th October 2004. Available on WWW: <http://www.w3.org/TR/xmlschema-2>

[18]: JSON.org. *Introducing JSON*. [online] Available on WWW: <http://www.json.org/>

[19]: MedBiquitous.com. *MedBiquitous REST Web Service, Design Guidelines.* [online] 28th October 2009. Available on WWW: <http://www.medbiq.org/std_specs/techguidelines/RESTguidelines.pdf>.

[20]: BARRY K. Douglas. *Service-oriented architecture (SOA) definition.* [online] Available on WWW: <http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html>

[21]: WANG Yue. *Web Services for a Software Development Platform: Master degree project*. Kalmar / Väjxö: Linnæus University, 17th August 2008. 78 p.

[22]: W3C. *Web Services Addressing (WS-Addressing).* [online] 10th August 2004. Available on WWW: < http://www.w3.org/Submission/ws-addressing/>