Service Matching with Contextualised Ontologies

Dissertation

Schriftliche Arbeit zur Erlangung
des akademischen Grades eines
Doktors der Naturwissenschaften

Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn

Ulf Rerrer-Brusch

Paderborn, Oktober 2006

To Margit and Clara

Acknowledgements

I would like to gratefully and sincerely thank my advisor Odej Kao for his expert guidance, understanding, patience, and most importantly, his friendship during my work at Paderborn University. His academic research and mentorship are a powerful example for me. I am not sure many PhDs are given the opportunity to develop their own individuality and self-sufficiency by being allowed to work with such independence. For everything you have done for me, I thank you.

I am also grateful to Thorsten Hampel who agreed to review this thesis. Furthermore, I would like to thank Irene Roger who provided the so much needed humor and entertainment in what could have otherwise been a somewhat stressful work environment. My dear colleagues as members of our little doctoral group Bernd Eßmann, Frank Götz, Tomasz Mistrzyk, and Simon Oberthür for their constructive thoughts and ideas. And of course all members of PC^2 who became wonderful colleagues in recent time.

I am not able to mention all students who supported me during this thesis, but I want to name a few who helped me to realize this work. All members of the project group "Location-based Services" contributed their generous help in the experimental phase of this research, thanks a lot! Also thank you to Georg Birkenheuer, Bernd Eßmann, Frank Götz, Wilke Hagelweide, Ludger Lecke and Florian Pepping for proofreading parts of this dissertation and their support. I have appreciated all the support of the Department of Computer Science at the University of Paderborn during the years of my PhD study.

Finally, I would like to thank my lovely wife Margit for her everlasting love and continuous support. Even in hard times she was a great source of encouragement and had a lot of patience for my work. Often, she and Clara helped me to forget about work for a little while. Without them, this work could not have been completed.

Abstract

In this thesis, we look at the problem of how the quality of service discovery can be increased. Three major issues are addressed to solve this problem: service *annotation*, service *composition*, and service *contextualization*.

Service annotation is the process of describing a service with semantic meta-data. In traditional service discovery a request fails instantly if keywords do not match the descriptions. With semantically annotation the service's meaning is machine-processable and automatic service matching is possible. Unfortunately, creating these annotations is mostly handwork.

Service composition allows increasing the set of possible matches. If the function of a service does not match a request exactly, it will fail. Composeable services can combine basic functionality from different services to meet the individual request precisely. The immense amount of semantic information needs to be handled efficiently here.

Service contextualization integrates context information into the matching process. User's and service's contexts contain facts of their individual environments and preferences. It can be used to improve the discovery quality by eliminating mismatches.

We present approaches that contribute to the previously described issues. The goal of this thesis is to improve the quality of service matching with an *automatic semantic service annotation framework, manageable service composition consideration and a service contextualization approach*.

The main elements of the thesis are a WSDL-to-OWL-S algorithm, which creates automatically semantic annotations to new developed Web services without extensive manual description work, a matching-cut procedure reducing the available semantic information in the service composition process which makes compositions considerable in the service discovery, and a service contextualization technique integrating context information from different domains into the service matching process completes the contribution.

We have implemented the approaches prototypically. The matchmaking components were evaluated with a large set of real semantic Web services to show the benefits of our approaches.

Contents

1 Introduction

We're not lost. We're locationally challenged.

John M. Ford

The emerging world of mobility is characterized by a multiplicity of exciting new technologies, applications, and services. The narrower sense of mobile computing is data processing on small and portable devices like notebooks, personal digital assistants (PDAs) or mobile phones [133]. Due to the mobility and location independence of the participating users new challenges in technology arise. Wireless communication and ad hoc network organization are only two aspects evolved in the mobile device context. The air, for example as shared communication medium, is interference-sensitive, has limited bandwidth, and holds fragile privacy properties [124]. However, these challenges are overcome.

But the mobility also enables new ways in application and service employment. Amongst the most promising ones is the ability to identify the exact geographical location of a mobile user at any time. This ability opens the door to a new world of innovative services, which are commonly referred to as location-based services [12]. Applications for emergency services, car navigation systems, or intelligent "yellow maps" use the location information to improve the service's value and convenience for instance in useability.

Regarding more environmental issues than only the location of a user leads to the concept of contexts. Context-awareness in mobile computing is awareness of the physical environment surrounding a user and their mobile device. With advances in sensor technology, awareness of physical conditions can now be embedded in small mobile devices at low cost. Even without further sensing capabilities, user activities, preferences and states generate a complex structure of personalized context information. Context is a key concept in natural language processing and more generally in human-computer interaction [141]. For instance applications can use contexts to adapt graphical user interfaces or other application features to context information such as user preferences and interaction status.

When dealing with services e.g. Web services, the application area is not only human-computer interaction, but also computer-computer interaction. The difficulty in this area is to make the content more easily machine-processable and to provide intelligent techniques to make the meaning of data machine-understandable. The Semantic Web [17] is supposed to make data located anywhere on the Web accessible and understandable, especially to ma-

chines. For instance, ontologies are a form of knowledge representation about a specific domain of the world and can be used to reason about objects in that domain and relations between them. Ontologies make information machine-processable and due to their relations between different concepts also understandable.

The rapid progress of the "information society" in the past decade has been made possible by the removal of many technical barriers. Producing, storing and transporting information in large quantities are no longer significant problems. Once created and stored it is also possible to move the information around in almost unlimited fashion. Global connectivity to the Internet has solved the transportation issues bringing not only offices, but also households on-line with high bandwidth. The most urgent remaining problems lie in the field of information finding and information integration. The more information is available, the harder it is to locate the particular piece one is looking for. Even when it is possible to find any particular piece of information, it is very hard to combine it with any other information one may already possess. Information is only meaningful in its context, but most techniques available for publishing, locating and retrieving information deal with single, isolated pieces of information without the source environment.

In the case of services, a service discovery is difficult, matching user's request satisfactory. While many similar services are available e.g. in a service registry, which service should a client use? Which service meets the requirements best? Is my interpretation of the service description the right one? Three issues improve the quality of the service selection.

Firstly, context information is useful in service selection. For example, when Alice wants to print her presentation slides, the closest printer to her current position is selected instead of her preferred printer in the office. If she wants to print photos, a color printer is selected automatically.

Secondly, service compositions result in more and better alternatives in service selection. When Bob wants to print for example a ".doc" document, but all network printers are only capable to print postscript documents a service composition can be useful. One service converts ".doc" documents in postscript documents and another service prints postscript documents on network printers. A composition of both services is able to print ".doc" documents.

Thirdly, semantic descriptions help to discover services by meaning instead of syntactic descriptions. If Charlie for example is looking for a service to print his thesis, a semantic service registry would offer a *university printing service*, because the semantic description matches the user request. Also a *copy shop service* would be suggested as alternative because its meaning is related to "printing" even if the term does not emerge. A *building service* presenting blueprints will be ignored although the term "print" appears in its description.

This thesis focuses on the automatic annotation of Web services, the service composeability, and the contextualization of ontologies in order to improve the quality of semantic service matching.

1.1 Problem Definition

Service matching is the process of comparing service provider advertisements with user service requests. The searching process for suitable matchings is occasionally also called dis-

covery. Providers and users should be interpreted here in a very generic manner. Anyone providing a service, from one to some thousand services, is called *provider*. Anyone requesting a service is called *requestor*. This might be a human user e.g. searching for a specific Web service or another application or system gathering information provided by other services.

A *service* in this context is a software system designed to support interoperable machine-to-machine or human-to-machine interaction over a network [1, 22]. Services can be anything [165, 166], from network services over travel and banking services to business services. An often-cited example of a service is that of a *stock quote service*, in which the requestor asks for the current price of a specified stock, and the response gives the stock price [151]. This is one of the simplest forms of a Web service in that the request is filled almost immediately. Service matching is not restricted to Web services, but in this thesis we have mostly Web services in mind when talking about services.

1.1.1 Service Annotation

In order to enable a high quality service matching fulfilling the needs of different users, we need detailed described user requests and service advertisement. These descriptions contain *domain specific background knowledge*. The background knowledge for the services is already available at the stage of the service development, but usually not specified immediately and directly by the programmer. Manual semantic annotation after finishing the development is a very complex, time-consuming, and error-prone work. Therefore, a framework for automated service annotation is needed, supporting the programmer during the development process annotating a service investing only slightly more work. A thoroughly semantic description can capture the meaning of the service which can be used to improve the quality of the service matching. For instance, *synonyms* like “buy” and “purchase” are completely different words, but have the same meaning. *Homonyms* like “order” have completely different meanings in different contexts (“order” in the sense of proper arrangements and “order” in the sense of a commercial document to a request). Both confuse machines without the help of semantics.

For traditional service registries informal service descriptions are provided describing the service’s function and purpose superficial. A *semantic annotation* to a service is much more precise, specifying inputs, outputs, preconditions, and effects using concepts from commonly agreed ontologies. These ontologies provide a shared understanding of certain terms and concepts in a specific domain.

When creating a new Web service, e.g. a *book selling* service, the developer usually has access to domain specific knowledge specified in company-wide knowledge-bases. Here, terms like “book”, “credit card”, or “customer address” and the relations between them are defined e.g. in an ontology concerning books and an ontology describing business process concepts. So it is easy for the programmer to associate the appropriate ontologies in the domain of the new Web service. Now an algorithm can associate the correct concepts to the service’s inputs like the requested book and the number of a valid credit card, and the customer’s delivery address to the service’s output likewise. This process can be automatized using the generated interface description of a Web service. This interface description contains all essential information concerning the Web service. The specified inputs and outputs use previously defined simple data types such as a credit card number, or complex data types like a book composed of

author, ISBN number, publisher, etc. This data types can be matched to the commonly known concepts in a domain specific knowledge-base.

1.1.2 Semantic Service Composition

In a traditional service discovery a request fails if keywords in the request do not match the manual service description. With semantically annotated services the meaning of the service's inputs and outputs can be understood. Unfortunately, if the function of a service still does not fit the request exactly, the request fails also. E.g. if a user is looking for a print service to print his/her ".doc" file, the request will fail, even if a print service is available, but only capable of printing postscript files. If the service matching concerns *composeable services*, a composition of a "doc2ps" translation service and a "printing" service will successfully match the request. A service composition is therefore able to improve the matching quality significantly.

With the help of semantic annotated services the meaning of the request and available services can be understand to successfully match the request with a composition of multiple basic services. All elements affecting the matching originate from ontologies, syntax and semantic descriptions. After gathering all information a reduction to the relevant elements for the matching needs to be performed in order to reduce the matching complexity. This reduction enables us to combine services from different domains identifying the shared concepts of different knowledge-base descriptions to match the created composition to the request.

This thesis focuses on the issues directly involved with the matching of semantic Web services. Many interesting problems exist in the area of service composition, which are not considered here such as transaction safety, security issues, or privacy concerns [1, 65, 67, 98].

1.1.3 Semantic Service Contextualization

In larger service registries the number of matching services and possible matching service compositions increases significantly. A user might seek for any service or service composition matching his/her requirements. Huge result sets may be generated to rather generic queries. Often, it is difficult for the user himself to specify any restrictive constraints to reduce the result set. However, the user's and service's *contexts* contain descriptions of their individual environments. A service request depends highly on the current situation of the requestor. Furthermore, the services offered by providers depend on their current environment. On both sides context information helps to conclude certain facts or preferences useful for the matching process. It *reduces the result set* to a query.

Service contextualization describes the process of integrating context information into the service description. This procedure comes along with a second advantage. By eliminating mismatching results concluded from the current circumstances *improves the result quality* significantly. Applying this approach not only to syntactical context descriptions like in mobile or pervasive computing [71, 72], but using semantic description tools to understand the meaning of the environmental properties enables this possibility. Using a shared understanding through common terms and concepts enables to compare contextualized services from different domains using different context descriptions.

The already mentioned example of a printing service is also suited to illustrate the contextu-

alization benefits and will be repeatedly considered throughout this thesis. For example users select a *printing service* to print his/her documents. The request is automatically enhanced with additional context information specifying the location of the requestor and the contextualized service adapts to this information by printing the document to the closest printer to the user. When the activity of the user changes from e.g. printing business emails to printing private photos the context information reduces the matching services capable of printing to those able to print colored documents.

Summarizing, this thesis considers the aspects of automatic service annotation during the development process of Web services. The resulting semantic service descriptions improve the quality of the matching compared to traditional service matching approaches. The semantic annotations are further used to evaluate different service compositions resulting in better adapted and more alternative services to user requests. It is important to stress out that even services from different domains are composeable due to the ontology-based descriptions. At last, context information is integrated to reduce the service matching result set to only relevant services. Furthermore, the contextualization improves the matching quality significantly.

1.2 Contribution

In this thesis we propose solutions to the different steps in the service matching process defined above to increase the quality of the matching result. In order to constitute realistic experiments for a contextualized semantic service matching we started to developed different services and descriptions to apply our approaches. From the huge variety of context issues Web services can contain, we choose the location property as an easy but not trivial to generate and describable context property. We developed a novel *positioning technique for wireless* LANs (WLANs) to determine the location of WLAN devices within buildings using the regular wireless network infrastructure. We achieved an adequate positioning accuracy to apply several *location-based services*, a specialized form of context-aware services. Our *enhanced fingerprinting algorithm* matches current signal strength measurements with previously collected reference vectors calculating the device's position. The signal measurements are taken from multiple access points, directly at the mobile device.

In the area of annotating semantically Web services we developed a novel framework to support the service developer in the service creation process. Our *WSDL-to-OWL-S algorithm* enables the developer to annotate the new service automatically without investing extensive additional work like in the common manual annotation process. A WSDL interface description can be generated automatically in many programming languages [4, 6]. Our approach uses the W3C *Web Ontology Language for Services* (OWL-S) to describe the semantics of a Web service. The data types defined in the WSDL document and used in the interface description are associated automatically with concepts from domain specific ontologies selected by the service developer. These associations are adequate to *generate a complete OWL-S service description.*

As we match semantically described services to user requests, an exact match is often unsuccessful. A *service composition* of basic services generates much more alternatives for a

successful service matching. We propose a procedure to check the composeability of services comparing their semantic OWL-S description. If inputs, outputs, and preconditions match, the composed service is encapsulated and functions as a new alternative in a user's service discovery. Our approach combines even semantic Web services from completely different domains. The associated ontologies are compared and shared concepts are identified. Our new *matching-cut* procedure reduces the available semantic information to the relevant entities in order to match the preconditions and outputs of one service to the inputs and preconditions to a second service. This allows a much higher matching quality than in other semantic matchmaking approaches which usually only consider the service profiles.

Context information is useful to adapt applications and services to the users needs. In service matching, context information helps to identify requestor preferences and providers application areas. We use context information to enhance the semantic service matching process. Our strategy for a *service contextualization* integrates semantically described context information into the service descriptions modeled in ontologies. To compare and match services using different context interpretation – or more abstract context providers – we developed an *upper context ontology*. This upper ontology enables context ontologies from different sources to associate to one common knowledge-base on a high level. Therefore, shared concepts and interpretations can be identified and the contextualized services can be matched to user requests. If requestors also use context information in their queries, our applied methods use this information to reduce the matching sets and eliminate mismatching services.

Finally, we *present our prototype system* for the location-based services and evaluate our service matching of contextualized semantic Web services. A test set of over 500 semantically annotated Web services from different domains including numerous sample queries is used to evaluate the matchmaker performance and illustrate the benefits of out approaches.

1.3 Synopsis

In the following chapter 2 we introduce context-awareness. Context related terms are defined and modeling approaches are discussed. Furthermore, location-based services are classified and characterized, including an overview of positioning techniques and positioning technologies in order to automatically sense location information as one context property. The chapter closes with a short review of computer-supported cooperative work (CSCW) as one application scenario for useful location-based services.

The next chapter 3 presents the vision of the Semantic Web and discusses the standards and technologies used to bring this vision into being. The major building blocks are WSDL, RDF, and OWL-S which form the foundation of the idea presented in this thesis.

Chapter 4 is devoted to the automation of the semantic description generation process. Here, the framework supporting a Web service developer in the semantic annotation process is described. The annotation is used in chapter 5 to check for service composeability in order to create better service matching alternatives. The inputs, outputs, and precondition descriptions are matched after a significantly reduction of the relevant semantic information. Chapter 6 presents the contextualization approach as third and last step to improve the semantic

service matching.

Chapter 7 shows an evaluation and discusses the experiences gained and the lessons learned while experimenting with the presented approaches. In chapter 8, we conclude the work in this thesis and we give an outlook on future research directions.

2 Context-Aware Services

> I've done some things I wish I could erase. ...
> I invented mistakes. But the mistakes must be seen in context, and they must be weighed along with the positives.
>
> Sammy Davis, Jr.

"Let not the mobile phone ring at the theater or as my mother-in-law serves dinner". The motivation of context-aware computing springs from the natural desire to preserve ourselves from its intrusion into social situations. It is not coincidental that the notion of context-awareness has become increasingly popular as technology has become ever more pervasive.

Context is a key issue in the integration between humans and computers, describing the surrounding facts that add meaning to something. Humans are quite successful at reacting appropriately in everyday situations understanding its meaning. The richness of the language they share, the common understanding of how the world works, and an implicit understanding of information are only a few of many factors making this behavior possible. This implicit information, or context, can be used to enhance the traditional ways of providing input to a computer and to design more powerful and useful computational services.

The concept of context and context-aware computing increasingly gained importance in the area of distributed systems since the 90's. Since then, it seemed to be a promising solution for a lot of problems which have been implied by the usage of mobile devices in ever-changing environments. In the ubiquitous computing paradigm sensing a person's environment plays an important role. Therefore, the classic contents of contexts in the early days were location, time, and temperature or network bandwidth. Exotic hardware sensors were embedded to realize numerous prototypes in the field of mobile computing devices. Context computing, which includes the collection, transformation, interpretation, provision and delivery of context information, is regarded as the origin to develop smart environments and applications.

The basic idea introducing context in the field of computing is to gain more information about the surroundings as mentioned before. Casually, using this additional information makes it possible to reduce the information overload in computing systems. The magnitude of information today's services and systems are dealing with is much too high and far to complex for intelligent human-computer-interaction. Real-time and other interactive systems require

mechanisms like taxonomies, classifications or filters provided by contexts to realize adequate results. Context-awareness facilitates simplified, convenient and efficient systems, but what exactly is context?

2.1 What is Context?

Most people understand what context is, but they find it hard to define it precisely. The word *context* is defined for example as "the interrelated conditions in which something exists or occurs" in the Merriam-Webster Dictionary [1]. The use of the word tends to be vague because everything in the world happens in a certain context.

The study of contexts spans over a lot of disciplines from Philosophy over Communication, Computer Science, Linguistic, to Industrial Engineering. The term has also been used in many ways and different areas of computer science e.g. Artificial Intelligence or Mobile Communication.

Previous definitions of context are done by enumeration of *examples*, by choosing *synonyms* for context, or by *categorizing* the environmental aspects. Context, relevant in the field of computing services, was firstly defined by researchers of pervasive computing listing features of environments. SCHILIT and THEIMER [137] refer to context as location, identities of nearby people and objects, as well as changes to those objects. RYAN ET AL. [135] define context as the user's location, environment, identity and time. BROWN ET AL. [26] specify the location, identities of people around the user, time, season, temperature etc. These definitions are difficult to apply. They enumerate several attributes to a context. It is not clear, whether a specific type of information is listed in the definition of a context, or not.

Other definitions provide simply *synonyms* for context. For example FRANKLIN and FLASCHBART [51] view context as the situation of the user. WARD ET AL. [160] see context as the state of the applications surroundings and RODDEN ET AL. [131] define it to be the application's settings. As with definitions by example, definitions by synonyms for context are difficult to apply because some consider context to be the user's environment while others consider the application's environment.

With lessons learned operational and practical context definitions were finally provided. By *categorizing* environment aspects a formal concept suitable for broad application scenarios was developed. SCHILIT ET AL. [138] claims that the important aspects can be qualified by where you are, who you are with, and what resources are nearby. Thus, a computing, user, and physical environment cover the constantly changing execution aspects of a context. DEY ET AL. [40] define the user's physical, social, emotional and informational state. Finally, PASCOE [116] defines context to be of physical and conceptual states of interest to a particular entity. In these definitions context is all about the whole situation relevant to an application and its users. In some cases, the physical environment may be important, while in others it may be completely immaterial. The most popular and for this work most adequate definition of context is given by DEY and ABOWED [39].

[1] `http://www.m-w.com`

Definition 2.1 (context)
Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

This definition makes it easier to enumerate the context for a given application scenario. If a piece of information can be used to characterize the situation of a participant in an interaction, then that information is part of the context. For example the user's location can be used to characterize the user's situation. If the user is located in the United States, the sum of the weights will be presented in terms of pounds and ounces. If the user is located in Germany, these values will be presented in terms of kilograms.

2.1.1 Context-Awareness

Contexts, which follow our definition 2.1, can still cover different pieces of information to build a context. This information can be categorized in context *types* to help to understand the intention and the designated usage of a particular context. Context-aware applications, which are defined below, look at the who, where, when and what of entities and use this information to determine why the situation is occurring. There are certain types of contexts that are, in practice, more important than others. These are *location*, *identity*, *time* and *activity*. These *primary* context types characterize the situation of a particular entity and can be distinguished from *secondary* context types. Primary types act as indices into other sources of contextual information. For example, given a person's identity, related information such as phone numbers or email addresses can be acquired to set up a detailed context. This two-tiered approach is proposed in [39], but similar type definitions occur in [135, 138].

When applications begin to *use* context or in a finer expression *adapting* their behavior to context we come to the definition of *context-awareness*. It is commonly agreed that the first research investigation of context-aware computing was Olivetti's *Active Badge* system [159]. This system locates users within a building using infrared badges attached to each user and routes calls for each user to the closest available telephone. Since then, there have been many attempts to define context-aware computing. HULL ET AL. [77] and PASCOE [116] use context by giving computing devices the ability to detect and sense, interpret and respond to aspects of user's local environment and the computing devices themselves. SALBER ET AL. [136] define context-aware to be the ability to provide maximum flexibility of a computational service based on knowledge of the user's context. Getting more specific applications changing their behavior dynamically based on context is referred to as adapting to context. RYAN ET AL. [135] describes applications that monitor input from environmental sensors and allow users to select from a range of physical and logical contexts according to their current interests and activities. Combining all these aspects leads us to the common definition of context-aware computing or context-awareness also by DEY and ABOWED [39].

Definition 2.2 (context-awareness)
A system or application is context-aware if it uses context to provide relevant information to the user and adapts its behavior according to the context, where relevancy depends on the user's task.

This definition is more general than some of the above mentioned definitions, but other specific definitions do not fit. For example, an application that simply displays the context of the user's environment to the user is not modifying its behavior, but it is context-aware.

Context-aware systems, which follow our definition 2.2, can still have a broad variety of features. A further attempt helping to define the field of context-aware applications more closely are a categorization or classification of features into an *taxonomy*. DEY and ABOWED [39] suggest three categories for feature classification of context-aware applications. The first is *presentation* which contains all features for presenting information and services to a user. This interaction technique presents a list of objects and places, where items relevant to the user's context are emphasized to ease the user's choice. Applications in the category of *automatic execution* of a services list services which are executed automatically when the right combination of context exists. Usually these are actions based on simple if-then rules. The last element in the taxonomy is *tagging* of context to information for later retrieval. Here the ability to associate data with the user's context is captured. Information like e.g. virtual notes can be associated to objects and places to provide details on these items when other users come close-by.

The previously discussed definitions of context and context-awareness are kept in a more general notion, but most of them do not consider the concept of services in all its facilities. The definitions are restricted to user/device or user/service interactions. Context-aware systems, firstly used in the ubiquitous computing paradigm, usually have a user-centered view of the applications and do not consider automatic service-to-service interaction. Therefore, we define the term *context-aware service* to emphasize the important role of services in today's systems.

Definition 2.3 (context-aware service)
A service is context-aware if its output or behavior is influenced or enhanced by relevant context information. Without context a context-aware service is still operational.

To have an even clearer distinction between regular services and services using context we will work with another type of services, defined as *context-based services*. A context-based service computes its output based on a context. The main difference between both definitions arises when context information is not available to services. A context-based service does not fulfill any task in this situation while a context-aware service can operate furthermore in a different way.

Definition 2.4 (context-based service)
A service is context-based if its output or behavior can only be applied if relevant context information is provided to the service. Without context a context-based service has no significant function.

An open question still is how to model contexts properly. This is considered in the next section.

2.1.2 Context Modeling Approaches

In previous works, both informal and formal context models have been proposed. Informal context models are often based on proprietary representation schemes which have no facilities to ease shared understanding about context between different systems. The CONTEXT TOOLKIT [38] for example represents context in form of attribute-value tuples, and COOLTOWN [91] proposed a Web-based model of context in which each object has a corresponding Web description. Formal context models commonly employ formal modeling approaches to manipulate context. ER[2] models or first-order predicates are only some examples [72, 122]. In the following, different modeling approaches are introduced and evaluated.

Realizing context-aware systems or services requires thoroughly modeled contexts. Many different context modeling approaches exist. In [148] these approaches are classified by the scheme of data structures which are used to exchange contextual information in the respective systems. A well designed context modeling and retrieval approach is a key accessor to the context in any context-aware system. The categories for context modeling approaches are key-values pairs, markup schemes, graphical models, logic-based models, object-oriented models and ontology-based models.

The model of *key-value-pairs* is the most simple data structure for modeling contextual information introduced by SCHILIT ET AL. [138]. In many frameworks information is stored in e.g. environment variables as a list of attributes in a key-value manner. The discovery procedure operates as an exact matching algorithm on exactly these attributes. Key-value pairs are easy to use, but lack of sophisticated structuring and the imminent danger of unrecognized inconsistency because no inheritance can be applied. This makes efficient context retrieval algorithms impossible.

Markup scheme models provide a hierarchical data structure of tags with attributes and content. Typical representatives of this kind are profiles usually based upon a derivative of the popular markup language XML[3]. For example, HELD ET AL. [71] propose a profile approach supporting the full flexibility of RDF[4] to express natural structures of profile information as required for contextual information. Context models in this category oftentimes are either proprietary or limited to a small set of contextual aspects, or both. Strengths are often the applicability to existing markup-centric infrastructures of a dynamically computing environment like Web services.

UML[5] is a very well known general purpose modeling instrument with a strong graphical component. *Graphical models* are, due to their generic structure, also an appropriate model to contextual information. BAUER [14] for instance models context aspects as UML extensions. In general, graphical models are in particular useful for structuring, but are usually not used on instance level.

When context is realized through facts, expressions and rules a *logic-based modeling* technique is used. Logic defines the condition on which a concluding expression or fact may be derived from a set of other expressions or facts. MCCARTHY ET AL. [101] introduced contexts

[2] Entity Relationship Models
[3] Extensible Markup Language
[4] Resource Description Language
[5] Unified Modeling Language

as abstract mathematical entities with properties useful in artificial intelligence. Other approaches like GRAY ET AL. [58] use first-order predicate logic as a formal representation of contextual propositions and relations. The level of formality is extremely high in logic-based context systems. They are usually composed distributed, but partial validation is difficult to maintain. Without partial validation the specification of contextual information is very error-prone.

The intention behind all *object oriented modeling* approaches for contextual information is encapsulation and reuseability, as always in the object-oriented paradigm. The main benefit of an object-oriented approach is helping to cover parts of the problem arising from the dynamics of contexts in changing environments. The details of context processing are encapsulated on object level and hence hidden to other system parts. Access to contextual information is provided through specified interfaces only. SCHMIDT ET AL. [141] uses so-called cues collecting values of a single physical or logical sensor. The context is modeled as an abstraction level on top of the available cues. Therefore, the cue objects provide contextual information through their interfaces, hiding details of determining the exact values.

Ontologies are a good instrument to specify concepts and interrelations. Using ontologies provides an uniform way for specifying a core concept as well as an arbitrary amount of subconcepts and facts. Altogether it allows a sharing and reuse of contextual knowledge in an highly dynamic computing system. The knowledge is evaluated using ontology reasoners. WANG ET AL. [158] for instance, created an upper ontology which captures general features of basic contextual entities and a collection of domain specific ontologies and their features in each subdomain. This allows contextual reasoning using inference engines developed for description languages. The *CoBrA* system by CHEN ET AL. [32] uses a broker-centric agent architecture to provide runtime support for context-aware systems. Ontological concepts characterize here entities such as persons and places to describe their contexts.

For instance, to provide a *printing service* with a context the environment and properties of a printer can be described via an ontology. Figure 2.1 shows a small part of such an ontology. Here, a postscript printer (`PSPrinter`) is a subtype of a `Printer` and has next to its usual properties like `Name` also environmental properties. E.g. the location of the printer is specified with the `InRoom` property. This allows together with further information about the building map to conclude which printer is nearest to a user. This feature is used for example by location-based services described in the next section.

Strong similarities between the modeling instruments of ontologies and objects-based manners are obvious. Concepts and facts in ontologies are analogue to classes and instances in object oriented modeling approaches. Both have a distributed nature, but can partial validate contextual knowledge which is highly desirable is contextual systems. The quality of information provided in contexts usually varies over time. A working set of information is typically incomplete or ambiguous. This should be covered by the model. At a level of formality an ontology modeling approach is more precise and traceable than modeled by objects. A precise definition of terms and a sharing of the same interpretation of the data in each participating party of an interaction is indispensable. Therefore, the shared “understanding” of the underlying meaning in contextual information can only be achieved with an ontology-based modeling approach.

```
<owl:Class rdf:ID="Printer"/>
...
  <owl:Class rdf:ID="PSPrinter">
    <rdfs:subClassOf rdf:resource="#Printer"/>
  </owl:Class>
  <owl:DatatypeProperty rdf:ID="Name">
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="InRoom">
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
  <owl:Class rdf:ID="PagesLeft"/>
  <owl:DatatypeProperty rdf:ID="Pages">
    <rdfs:domain rdf:resource="#AmountPages"/>
    <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
  </owl:DatatypeProperty>
...
```

Figure 2.1: *Example of a context modeled with ontologies*

2.2 Location-based Services

Filling context descriptions with content can be a time-consuming job. Despite the risk of incompleteness and ambiguity already mentioned in the previous section, the up-to-dateness and reliability of the provided information has a considerable value. Therefore, the desire to automatically sense different parameters especially for mobile users emerged very soon. The most important pieces of information are the primary context contents identity, time, location and activity. To determine the current time and identity of a user respectively his/her device is not very complex. Almost any mobile communication device is registered to a service provider when active and speaking of personal mobile devices a user can usually be identified by its device. To specify the precise location and activity of a mobile device is much more difficult. Different mobile devices require miscellaneous positioning techniques often separated from the applied communication technique. If the behavior of an application or the result of a service is a function of its location it is referred to as *location-aware*.

The term *location-based service* [12] is a concept that denotes applications integrating geographic location with the general notion of services. Many examples exist - mainly in the field of mobile phone networks - including applications or emergency services, car navigation or "yellow maps" as a combination of yellow pages and maps. LBSs[6] have on the one hand a long tradition (since the 70s the U.S. Department of Defence operates the GPS[7] and is on the other hand relatively new due to the newly developed technologies in mobile communication. Location-based services present many challenges in terms of research, industrial and commercial concerns. Approximately 20% of current mobile network operators income [140] is coming from SMS[8]. To fulfill the growing needs of the business, operators invested in new

[6] Location-based Services
[7] Global Positioning System
[8] Short Message Services

technologies like mobile messaging via MMS[9] and mobile Internet via UMTS[10]. LBSs are the new challenge over the next years. The user's location is an important dimension in this new data-service-oriented world.

Famous, and two of the first examples for location-based services in mobile phone networks, are the FINDFRIEND application from AT&T Wireless [9, 149] as well as E-911 service from the U.S. government [125] respectively the FCC[11].

The visionary and forward-thinking FINDFRIEND application delivers relevant user information about the location of a friend's cell phone position. Together with a GIS[12] database the application can transform the latitude/longitude coordinates pinpointing the location of a user's mobile phone into an address, city and state by reverse geocoding. The goal was to provide enhanced solutions for people to stay in touch with their friends and family, to find one another and to get directions to nearby locations. AT&T Wireless created the first-ever find friend application delivered over mMode (a WAP[13] derivate) to provide these services. It also obtains POI[14] information in proximity to a mobile phone position similar to the Yellow Pages. For example, it helps to find the nearest cinema, toy shop, gas station or airport. A user interface with driving directions to the once selected destination is then provided.

The second service concerns the U.S. public emergency system. The traditional emergency-calling service 911 automatically routes calls to the nearest 911 call center (PSAP[15]) and delivers a caller's location to the appropriate public safety entity, based on the phone's fixed address. It associates between the caller's line and the to this line registered address. This is often used in times of fire, break-ins or kidnapping where communicating one's location is difficult. With the increasing amount of cell phones there was needed to locate mobile callers. The E-911[16] service uses GPS[17] and other technologies like radio location from cellular networks to detect the caller.

2.2.1 Classification and Characteristics

Both applications, AT&T's FINDFRIEND as well as E-911, are visionary and early developments in the field of LBSs in mobile phone networks. Meanwhile, the market shows many interesting services from different areas which are more sophisticated than experimental toys. To get a baseline understanding of the potential the services and applications using location information can be classified into *information, emergency, navigation, tracking, billing* and *assistance* services.

[9] Multimedia Messaging Service
[10] Universal Mobile Telecommunications System
[11] Federal Communications Commission
[12] Geographic Information System
[13] Wireless Application Protocol
[14] Point-of-Interest
[15] Public Saftey Answering Point
[16] Enhanced 911
[17] Global Positioning System

Category	Push/Pull	Active/Passive	Example
navigation	pull	active	vehicle, tourist
tracking	push	passive	children, goods, fleetmanagement
information	pull	active	maps, schedules, weatherforcast, points of interests
emergency	pull	active	emergency, breakdown
billing	push	passive	home-zones, toll
assistance	push	active/passive	shopping, office

Table 2.1: *LBS application classification*

Providing information to a user depending on his/her current position is the nature of LBS and can be placed in the category of *information* services when realized without additional features. POIs are defined for travel services providing information about local sightseeing objects, Yellow Pages services notify about special local institutions like public swimming pools or tourist centers in the surrounding area. Infotainment services announce local events or location-specific multimedia content to interested users.

Services concerning *emergency* issues are the most obvious and plausible propose when thinking of location related or location-based services. Individuals - unaware of their exact location - in a case of an emergency (injury, criminal attack, etc.) use their mobile device to call for help. With life-threatening injuries a call for assistance can be used to reveal automatically the exact position of the caller extremely fast. Alarmed emergency forces are directed immediately to the right location using the positioning capabilities of the mobile device. In times of break-ins or kidnapping any public safety entity can be informed even without speaking with the victim. Services like the E-911, as mentioned in the paragraph above, are at the moment applied in the North America or E-112 respectively in the European Union.

Entering a foreign city or area the traveler's need for directions within the current geographical location can be satisfied with applications from the *navigation* category. These services allow finding places (shops, hotels, gas stations, etc.) depending on the traveler's location together with detailed maps and route descriptions (positions, directions, traffic conditions, point-of-interest information, etc.) transferred to a mobile device. Well established car navigation systems already provide these services. Starting with basic transformation from coordinates into street addresses and vice versa (geocoding and reverse geocoding) lead to ideas for more dynamic and complex services. The potential of more general LBS in this category lies in combined indoor and outdoor navigation connected with the power of highly interactive and user-friendly mobile devices like handhelds. Possible scenarios are finding special areas in malls or the presentation of a company on huge fairs.

Location-matters are of course important if objects or persons are lost. *Tracking* services are LBSs that help to find things or control the track of an object. Examples are applied for the discovery of lost children, stolen cars or elderly people who got lost in huge places like shopping malls. Applications allowing companies to locate their personnel (salesman or maintenance crews) as well as monitoring the state and position of part for supply chain management are highly demanded services.

Different location-sensitive billing systems can be applied depending on the current state and location of a customer. These *billing* services usually control which customer is charged at which rate for a specific service. For instance, a network operator defines a small zone around the customer's home including the garden, neighbor houses, etc where the customer usually accomplishes private calls. In this "home zone" calls from the mobile phone are cheaper than outside. Also possible are location-sensitive advertisements as another example of location-based billing systems. A message like "enter the coffee shop to your left and get 10% discount on a coffee" advertises nearby products to a different price.

The last category for LBSs completing this listing can be named as *assistance services*. These services work on location-based as well as context-based information. Besides just making location-specific information available to a user, an assistant LBS goes a step further and interprets the users habits and behavior at certain place and makes decisions for the user. For example, a shopping assistance service in a mall or supermarket can add the customer's favorite desert to his/her shopping list, if it is on sale and the supply at home is diminished.

The six described categories information, emergency, navigation, tracking, billing and assistance for location-based services are a recommendation proposed in this work. Although the term LBS is well known for several years, no commonly accepted classification exists [54, 139, 140]. More or less categories as well as different terms to group certain services are imaginable. Future services may also need additional categories, but the universal nature of this classification should be sufficient enough to classify any service.

Besides from mobile phone networks, location-based services originate from many domains. The above classification gives an idea of the diversity, potential and richness of useful LBSs. The resulting heterogeneity reflects in the commonly used terms concerning these features in different communities. The expressions *position* and *location* are used frequently in a similar manner. Apart from it varying representation, a position can be defined as follows.

Definition 2.5 (position)
A position is an exact, unique place based on a geographic coordinate system.

The term location has a more relative meaning than position and can be defined as subsequently described.

Definition 2.6 (location)
A location is a position in physical space expressed relative to the position of another point or object.

Now, with a precise definition of a location, the term *location-awareness* can be defined analog to the previous definition of context-awareness (definition 2.2). The LBS community has not produced a consistent definition of location-awareness yet [16, 37, 69, 86, 123]. A lot of synonyms like position-awareness or situation-awareness exist similar to the irregular definitions of context. Location-awareness is defined here as follows.

Definition 2.7 (location-awareness)
A system or application is location-aware if it uses location information to provide relevant information to the user, were relevancy depends on the user's task. The system output or behavior is a function of the location.

This definition comes as expected. Location information is a part of the more general context-information and therefore location-aware systems are a subset of context-aware systems. The systems behavior or output is reproducible at the same location. Therefore, the reaction of the system can be referred to as a function of the location. Like context-aware and context-based services, location-aware and location-based services can be defined in the same manner.

Definition 2.8 (location-aware service)
A service is location-aware if its output or behavior is influenced or enhanced by relevant location information. Without location information a location-aware service is still operational.

Definition 2.9 (location-based service)
A service is location-based if its output or behavior can only be applied if relevant location information is provided to the service. Without location information a location-based service has no significant function.

Location-based services do have a lot of important properties and interesting characteristics. Apart from numerous features advertised by commercial service providers, some technical properties are comprehensible. The design of location-based applications or services can be split into *person-oriented* and *device-oriented* approaches. Person-oriented LBSs often focus on the location of a person or use the position of a person to enhance a service. Usually the person to be located is aware of his/her locating-request and can control the attempt. Applications for communication, commerce and entertainment usually use person-oriented services, because user-service-interaction is required. Applications providing device-oriented services are not personalized. Instead a device or the state of a device is determined unnoticed. Therefore, an object or a group of people can be located, which are usually not in control or aware of the locating procedure. Tracking services usually work device-oriented.

A second characteristic for LBSs concerns the information flow between user and service. Services can be distinguished between *push* and *pull* services. If a services is invoked actively by user information is pulled from this service. A push service, in contrast, means that a user receives information as a result of his or her surroundings without actively requesting it. A user usually previously subscribes for this kind of service.

Another aspect can be introduced with third-party interaction. A *active* service is initiated by the location request of a user itself. If a third-party - another user, or a different service - locates an individual the service is *passive*. Typical passive services originate from the tracking category whereas active services are usually information services.

2.2.2 Positioning Techniques and Technologies

Communicating without wires is not a new concept. Broadcasting radio and television are two common examples of wireless communication; others include satellites, cordless and cellular phones, or even remote controls and car door locks. Most of these examples employ communications via radio waves or the use of infrared light. Next to the ability to communicate without wires these technologies have the power to determine a receivers location. A receiver can identify a signal, determine the angle of the received signal, or measure the signal strength of the received radio waves and process its origin.

The underlying technology and position technique is essential for a location-based service system. The positioning itself is based on sophisticated physics and mathematics which "sense" the location of an object or user. LBSs require location information as input. This can be sensed automatically using the right technology and positioning technique.

Positioning Techniques

Some basic techniques appear again and again in combination with different technologies in the context of determining positions. *Cell-based, time-based, attenuation-based*, and *proximity-based* solutions are frequently used. They have in common that usually wireless signals are used. Depending on the application area other techniques like *scene-based* or *probability-based* approaches are also applicable. Figure 2.2 shows an illustration of these techniques.

COO[18] [30, 73, 133, 153, 164] is the most popular technique when a **cell-based** solution is chosen. In a wireless network a stationary base station is used to propagate signals to a specific area. The position of each station is known and if a device is within communication range a connection is usually established. Identifying a cell in a cellular-based network makes is possible to draw conclusions from the position of a receiver. Usually a device connects to the strongest signal in range and the position of the base station is decisive. Substituting a cell in several different *sectors* can improve the accuracy which otherwise depends mainly on the cell-size.

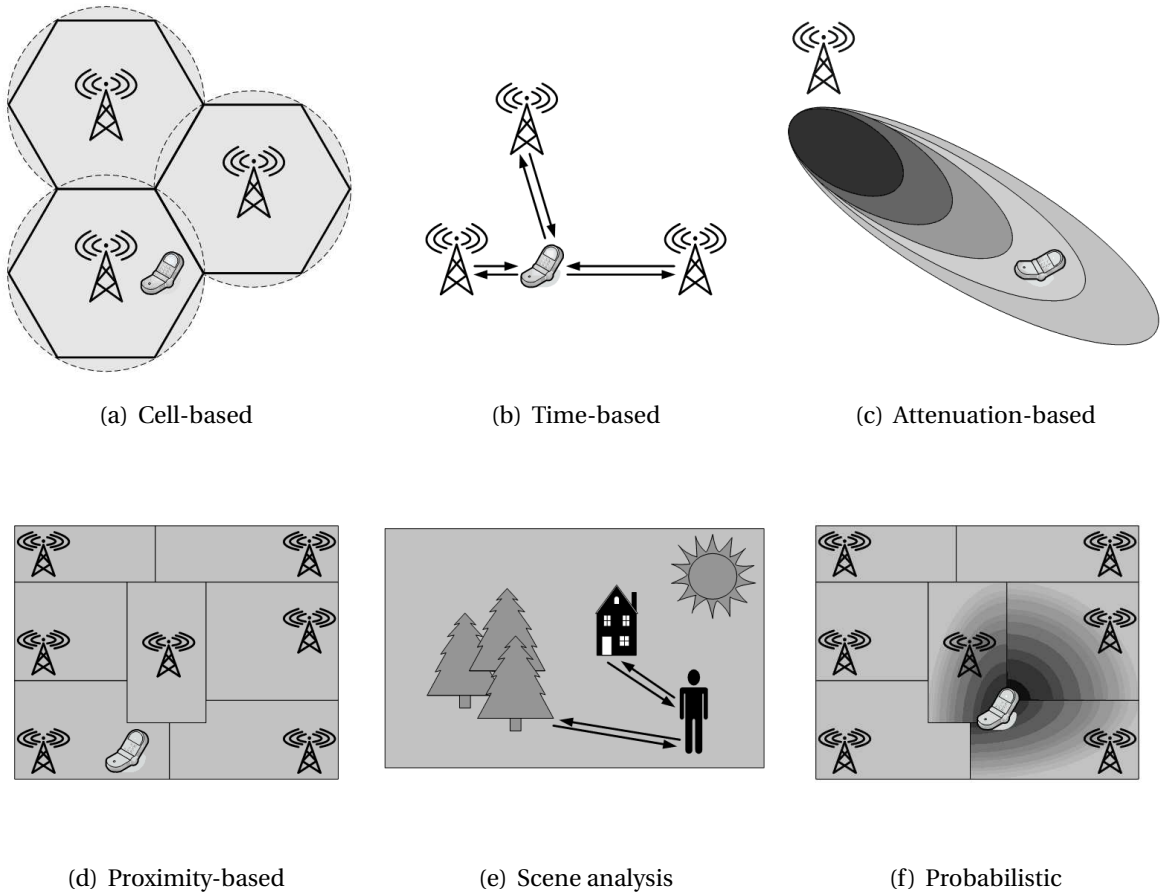
Triangulation [73, 133, 164] computes the location with geometric properties of triangles using either distances or angles and allows a more accurate positioning. In order to determine the distance between transmitter and receiver the signal propagation delay (time) or the received signal strength (attenuation) can be measured. Both, time-based and attenuation-based approaches use *lateration* to measure the distance to a number of multiple reference points (base stations). For *angulation* approaches, e.g. AOA[19] the angle between receiver and multiple transmitters is used to calculate the position of the receiver.

Mobile phone networks traditionally work with **time-based** techniques [73, 133, 163, 164] to determine the location of mobile phones if an accuracy higher than cell of origin is required. In TOF[20] also known as TOA[21] the time it takes a signal from transmitter to receiver is measured. Knowing the propagation time of the signals and the exact position of the base station allows determining the position of the mobile phone using triangulation. This requires high precision instruments and clocks on both ends of the transmission due to the propagation speed of radio waves. The E-OTD[22] also know as TDOA[23] is a variation of TOF where all base stations emit a synchronous timing signal. Calculating the time differences from several base stations on the receiving device allows determining its position. Especially on the mobile device the current mode, battery status and signal strength have a major influence on the result

[18] Cell of Origin
[19] Angle of Arrival
[20] Time of Flight
[21] Time of Arrival
[22] Enhanced Observed Time Difference
[23] Time Difference of Arrival

in both approaches. Signal interferences and reflection have a medium effect on these measurements (see section 2.2.3). To avoid these disadvantages the RTOF[24] technique demands the receivers to send an acknowledgement back to the transmitter immediately. The transmitter receives the acknowledgement message back on the same way the original signal was send. Measuring this *roundtrip* time only requires precise instruments and clocks on the transmitter and is therefore much easier to apply in mobile phone networks.

A characteristic of electromagnetic waves is the straight-forward propagation (line-of-sight) together with a proportional decrease of the signal strength at increasing distance. This **attenuation** [10,49,73,115] can be used to calculate the distance between transmitter and receivers knowing the output signal strength at the transmitter. RSS[25] [124] measures the received signal strength from the transmitter at the receiver and determines in triangulation similar to TOF the position of the receiver. Many influences like atmospheric properties, for example humidity, affect the signal propagation and have a major impact on the positioning accuracy in this technique.



(a) Cell-based (b) Time-based (c) Attenuation-based

(d) Proximity-based (e) Scene analysis (f) Probabilistic

Figure 2.2: *Overview of positioning techniques*

[24] Roundtrip Time of Flight
[25] Received Signal Strength

Proximity-based positioning techniques [8, 73, 109, 159] determine if a user is "near" to a know location. The user's presence is sensed through *physical contact, monitoring*, or *observing*. Pressure sensors, touch sensors, or capacitive field detectors sense physical contact. Monitoring is done by determining when e.g. a device is in range of a wireless cellular access point (see COO above). Observing means for example if a user uses her/his credit card at a point-of-sale terminal, which can be used to infer the location of the user.

In a **scene analysis** [73, 121, 126] users are observed by a reference point in order to draw conclusions about the position of the observer or the user in the scene. Usually the observed scenes are simplified to obtain the features that are easy to represent and compare. The location of objects and users can be inferred using passive observation and features that do not correspond to geometric characteristics. Measuring geometric quantities often requires motion or the emission of signals. A static analysis usually looks up recognized features in a database with a preset of objects already mapped to locations. A more dynamic approach tracks differences between successive scenes in order to estimate a location.

The use of **probabilistic** techniques [21, 29, 50, 73, 132, 162] provides a natural way to handle uncertainty and errors e.g. in signal power measurements. For the location estimation with a probabilistic-based technique machine learning is usually applied. A positioning system is initiated with training data and in a test-phase randomly determined test points are chosen, computed and compared to the correct coordinates.

Positioning Technologies

The previous section describes the principle way positioning can be accomplished. Only in combination with a technology using specific hardware leads to a helpful positioning system. The upcoming trend for mobile computing in the 90's created visionary, but also unordinary approaches testing also wireless communication technologies not intended for positioning originally. In the following paragraph *satellite-, network-, infrared-, RF-, ultrasound-*, and *video-based* technologies are shortly introduced together with established examples implementing these technologies for a positioning propose.

The idea for **satellite-based** navigation is very old, because is has multiple powerful advantages. The positioning itself is available around the globe even in abandoned regions, environmental conditions like the weather only have a small effect on the result, and a high accuracy can be achieved. The famous GPS[26] [57] is the only working system at the moment. Vehicle (car, ship, plane, etc.) navigation, logistic coordination, fleet management, and cargo tracking are only a few applications based on GPS today. Once designed for military use from the U.S. Department of Defence, a life without a civil usage of the positioning system in our modern life is unthinkable.

A user, who wants to determine his/her position via satellite, needs the exact position of the satellites as well as the distance to them. If three satellites are in visual range, the exact location can be calculated using the lateration technique of the triangulation method mentioned above. The GPS system, operational since 1995, consists of 24 satellites in six orbits. In most places on earth five satellites are available, which have four atomic clocks each sending syn-

[26] Global Positioning System

chronized the current time and the satellite's position to its receivers. The - today very small - receivers calculate the current position with this data.

Due to economy reasons, GPS receivers are not equipped with atomic clocks. Therefore, a forth satellite signal acquires correction data and allows the high accuracy [133]. GPS works in two modes. The SPS[27] guaranties an accuracy of 100 meters in 95% of all cases. The bad performance is caused by an artificial error called "selective availability". This mode was deactivated in May 2000 and the more accurate and encrypted PPS[28] was disposed for civil usage delivering a 22 meter accuracy in 95% of all cases or a 10 meter accuracy in 90% of all cases. Auxiliary terrestrial transmitters enhance the GPS signal to an exactness of one to three meters in a method called D-GPS[29]. The United States have reservations to activate the SPS mode in times of crises.

The EU and the ESA[30] are building GALILEO [52], which's release date is in 2008. GALILEO intends a higher precision than GPS up to ten meters without and one meter with additional auxiliary terrestrial transmitters. The 27 satellite system will have several commercial services with different accuracies, which allows indoor positioning (with auxiliary transmitters) and no fallback in times of a crisis. The Soviet pendant GLONASS [55] to GPS was developed in the cold war. Initial satellites were released in 1982, but the full operating mode with 24 satellites was never reached, due to the short lifetime (three years) and insufficient funding of its satellite program.

The satellite technology is ideal for the intention of location-based services due to the huge coverage, the easy access, and high accuracy of the positioning data. Unfortunately, complex location-based applications require a lot of communication and interaction with other networks, services, applications, and users. This is not possible with a read-only satellite system and without a universal data network. Furthermore, satellite signals cannot be received indoors and in signal shadow areas without immense additional efforts.

Next in the line of positioning systems are **network-based** approaches. Each network provider has a unique radio frequency band. Mobile phones and the network they operate under vary significantly from provider to provider, and even from nation to nation. However, all of them communicate through electromagnetic radio waves with a cell site base station, the antennas of which are usually mounted on a tower, pole, or building. Triangulation of time-based techniques was applied early to determine the position of a mobile device besides the cell-based information.

GSM[31], GPRS[32] and UMTS[33] are the most favorite and established technologies in the evolution of network standards. GSM builds the second generation (2G) of mobile phones. In addition to voice transmission small amounts of data are transferred between base station and mobile device. SMS messages are one example, but also cell information is shared which can be used for LBS like the "home zone" scenario (see section 2.2.1). New third genera-

[27] Standard Positioning Service
[28] Precise Positioning Service
[29] Differential GPS
[30] European Space Agency
[31] Global System for Mobile Communications
[32] General Packet Radio Service
[33] Universal Mobile Telecommunications System

tion (3G) phones in GPRS and UMTS networks have much more computational power. Better displays and higher transmission rates enable more complex mobile services, especially location-based services. Sadly, the data communication in those networks is strictly proprietarily. Besides the limited capabilities of human interaction with the mobile device, the creation of individual location-aware applications is not possible.

Infrared (IR) wireless communication was one of the first technologies for short-range wireless communication. This technology is used in devices such as remote controls, PDAs[34], notebooks, etc. Infrared makes use of the invisible light spectrum just beyond the red wavelength in the visible spectrum. In particular, one standard method for infrared communication is specified by the IrDA[35] [80]. It is designed for short-range, low-power wireless communication.

Advantages of the technology are the very low priced hardware. Simple LEDs[36] can be used as transmitter, simple photodiodes as receiver. Infrared requires typically a line-of-sight path between transmitter and receiver while other technologies can penetrate solid objects. Its usage necessitates no license. Unfortunately, infrared communication has a relatively small bandwidth. In addition, the full bandwidth can only be achieved in very short ranges, with no obstacles in the line-of-sight, and with no jamming entities like heat sources nearby. The limited range of an IR network for positioning is a handicap in providing ubiquitous coverage. Although it is often deployed for solely location determination, it does not provide traditional data networks services. Therefore, it is not suitable for complex LBSs.

Positioning systems using infrared technology normally employ transmitters as beacons combined with proximity- or cell-based techniques. The ACTIVEBADGE system [159] introduced by WANT ET AL. was an early and significant contribution to the field of location-aware systems. Sensors placed at known positions in the ceiling within a building receive the unique identifier infrared signal from a badge worn by a person to be located. The beacon signal emits every ten seconds and relays this to a location manager software, which provides the location information for further disposal. Unfortunately, the ACTIVEBADGE system suffers from several drawbacks. First, it scales poorly due to the limited range or IR. Secondly, it cause significant installation and maintenance costs and finally the system performs badly in direct sunlight environments.

Radio Frequency (RF) technologies employ transmitters and receivers tuned to produce and consume electromagnetic waves of a given frequency spectrum. Via modulation individually information can be encoded and interpreted. Bluetooth, WLAN[37], UWB[38], and other wireless communications operate with RF. It can penetrate many solid obstacles such as clothing, human bodies, walls, doors, and the alike. This means there is no need for a undisturbed line-of-sight between transmitter and receiver.

To overcome the problem of limited range of IR technology the *RFID*[39] [48] was developed building a **beacon-based** (or tag-based) location system. Unfortunately, these RFID systems,

[34] Personal Digital Assistants
[35] Infrared Data Association
[36] Light Emitting Diode
[37] Wireless Location Area Network
[38] Ultra-Wideband
[39] Radio Frequency Identification

like their IR counterparts, are often build for the solely propose of determining locations and do not provide any data network capability. A basic RFID consists of three components; the antenna, a transceiver with a decoder, and a RF-tag electronically programmed to emit unique signatures. Antennas are the link between tag and transceiver, which controls the system's data acquisition and communication. It sends radio signals to activate the tag - coming in a wide variety of shapes and sizes - and read and write data to it. *Active* RFIDs are powered by an internal battery and are typically read/write. *Passive* RFIDs operate without a separate power source and obtain power generated from the reader by magnetic induction. They reflect the RF transmission and add information by modulating the signal. Passive tags are much more lighter than active tags, less expensive, and offer a virtually unlimited operational life [48]. The trade-off is that they have shorter read ranges and require a high-powered reader. RFID systems use either low frequencies (30-500 KHz) or high frequencies (2.4 GHz) offering read ranges up to 50 meters (special antennas increase this to 300 meters) and response times less than 100 milliseconds.

LANDMARC [109] is a reliable and cost-effective location-sensing system developed by NI ET AL. that uses active RFIDs for locating objects inside buildings. The reader determines the tag's location based on the received signal strength from the tag (one of eight discrete power levels). As the reader only reports whether a tag is in its detection range or not, the system uses a proximity-based technique. In contrast of having more readers in place to increase the accuracy, the LANDMARC system employs the idea of having extra reference tags as reference points like landmarks in our daily life. These reference tags at known locations can also be located by the system on a scene-based manner to eliminate environmental effects that contribute to the variations off the detection range. The system spends one minute each time to scan the eight discrete power levels to estimate the signal strength of all tags in range. A problem is the changing behavior due to different chips and circuits as well as batteries even from the same RFID vendor. This causes a poor latency and accuracy of the overall system.

In search of other technologies to realize positioning systems **ultrasound-based** systems achieve a considerably high precision. Because the speed of sound (330 m/s) is low compared to radio signals, it is easier to get an exact measurement without high technical effort. Disadvantageous is the need for specialized and expensive hardware. Usually a TOF lateration or a proximity-based technique with beacons is applied.

The ultrasound positioning system ACTIVEBAT [66] proposed by HARTER ET AL. obtains a high precision of ten centimeters using an ultrasound time-of-flight approach. A user carries a so-called *Bat*, which is a device that sends a short ultrasound impulse upon a request. Stationary devices at know locations, that are assembled at the ceiling, receive the ultrasound signal and immediately pass his information to a location server via a wired network. The CRICKET [118] system from PRIYANTHA ET AL. switches the roles of transmitter and receiver compared to the ACTIVEBAT system. Fixed installed beacons send ultrasound signals received by mobile badges. CRICKET and its successor CRICKETCOMPASS [119] rely on beacons transmitting a RF signal combined with an ultrasound wave. A receiver estimates its position by listening to the beacon emissions and finds out which one is the nearest.

Realizing a truly scene-based positioning technique **video-based** technologies are the preferred choice. Video data is collected an evaluated in intensive computations. Visual tags such as colored badges have patterns that can be recognized easily and with fixed cameras at

known positions and orientations a user or object can be positioned.

In the CYBERCODE [126] system by REKIMOTO ET AL. the mobile user is equipped with a small camera, mounted for example on the user's head. Visual tags are attached on walls inside a building. In this case, the visual tags have fixed positions and the camera is mobile. If the camera detects two or more tags it can determine its own positions. CYBERCODE like other technologies in this area lack of low accuracy and high computational power to process positioning results.

Radio Frequency-based technologies offer a more sophisticated way of positioning when used in a beacon-like manner (RFID) implementing a proximity-based technique. The popularity of RF-based wireless communication and the wide variety of technologies available allow exploiting other techniques such as attenuation. Bluetooth and WLAN operate in the license-free 2.4 GHz ISM[40] spectrum. The signal propagation and the resulting received signal strength decreases with growing distance and is therefore a function of the distance to the source.

Bluetooth [20, 105] gained huge popularity within a short time and chipsets working with Bluetooth are implanted everywhere (phones, cameras, printers, accessories, computing devices, etc.). It is a low power and low cost technology building a multi-functional communication standard providing IP and even voice services. Bluetooth was not made originally for location determination. However, devices in a Bluetooth environment need a mechanism to identify, synchronize and finally connect to their neighbors. Such mechanisms are built into Bluetooth and can be used to obtain an accurate positioning system. The specification permits any device to assume either role - *master* or *slave* - for one communication link. The role of master does not imply special privileges or authority; instead it governs the synchronization of the FHSS[41] [124] communication between devices. The master-slave relationship is necessary in Bluetooth's low level communication, but usually devices operate as equivalent peers. The master determines the frequency hopping pattern (based upon its address) and the phase for the hopping sequence (based upon its clock). Then all (maximal seven) active slaves plus a single master in a *piconet* hop together in unison. The first of two Bluetooth positioning approaches, which is a proximity detection, finds the nearest access point from which the location can be inferred. The second uses the *get-link-quality* and a *read-received-signal-strength-interface* operation in the Bluetooth specification [20]. Once a connection is established can process the attenuation information and compute the relative distance to each other and hence the relative distance to a known position. Location systems using Bluetooth technology suffer from a high latency of about 30 seconds and high implementation costs due to the short range, but a universal data network.

The BLUEPOINT [30] system introduced by CHAN ET AL. is an architecture that realizes Bluetooth locating infrastructures and a middleware shielding the distribution and heterogeneity of the underlying communication network. The system only works cell-based, where access points help to identify the location of mobile units. Several attributes of access points such as the ID, physical location, location aliases, and remarks are provided to the system. Since the active range of a Bluetooth device is just tem meters it is still offers a relatively high accu-

[40] License-free RF Spectrum for Industrial, Scientific and Medical Usage
[41] Frequency Hopping Spread Spectrum

racy, but needs many access points. In a future version of the system, an accessability of the received signal strength information is planned.

Even more familiar to normal users than Bluetooth is *Wireless LAN*, also WLAN or WiFi specified in the IEEE 802.11 standard [35,78]. The extensive commercial promotion created a hype in recent years. Nearly every purchasable notebook is equipped with a WLAN device. Vendors offering numerous affordable wireless gadgets, business efforts like HotSpots in public places, and easy-to-use software created a widespread acceptance for WLAN technology. Many characteristics of WLAN make this technology ideal for location-based services: *mobility*, *affordability*, *flexibility*, and *acceptance*.

Working in the 2.4 GHz ISM band, the radio frequency technology WLAN offers 13 partly superposed channels. Different national regulatories in Europe, America and Asia enforced slightly dissenting specifications, e.g. 11 channels in the U.S. High data bandwidth (up to 54 Mbit/s in IEEE 802.11a) and moderate indoor ranges (30-80 meters) underline the popularity. A mature security concept and the extensibility through bridges and repeaters complete the feature list of a successful technology.

WLAN as a RF technology can penetrate solid to a certain degree and is therefore suitable for positioning in combination with a attenuation technique. In the infrastructure mode access points also structure the wireless network in cells which allows to apply a cell-based positioning approach. Both attributes make the technology most suitable for positioning systems.

One of the first systems was RADAR [10,11] developed by BAHL ET AL. at Microsoft Research in the year 2000. The user location and tracking system uses an attenuation technique processing signal strength information from multiple overlapping access points. It estimates the position with comparison of online measurements with previously collected empirical measurements. This stored *radio map* delivers several ranked best matches which are triangulated to the resulting location estimate. RADAR combines in this *fingerprinting* approach attenuation and scene analysis techniques. The implemented NNSS[42] algorithm performed with an 3-4 meter accuracy in 50% of all cases. This result was achieved in an artificial environment with a high access point density. Enhancements such as a signal propagation model or tracking issues to improve the accuracy were introduced in [11].

CASTRO ET AL. introduced another system named NIBBLE [29] implementing a positioning system using WLAN technology. It combines scene analysis, attenuation-based, and probabilistic techniques. It uses a Bayesian network with an applied probabilistic modeling which can learn the room locations of a building. It infers the location of a wireless client from signal quality measures and has a modular structure providing a high flexibility when base stations are added, removed or spatially reconfigured. The system relies on a fusion service. One component is an evidential reasoning model that aggregates and interprets sensor information. The second component is a resource optimization model minimizing the costs of the gathered data. NIBBLE provides several interfaces that are available to application designers to implement LBSs freely. It achieved an accuracy about 97% for one experimental site.

[42] Nearest Neighbor in Signal Space

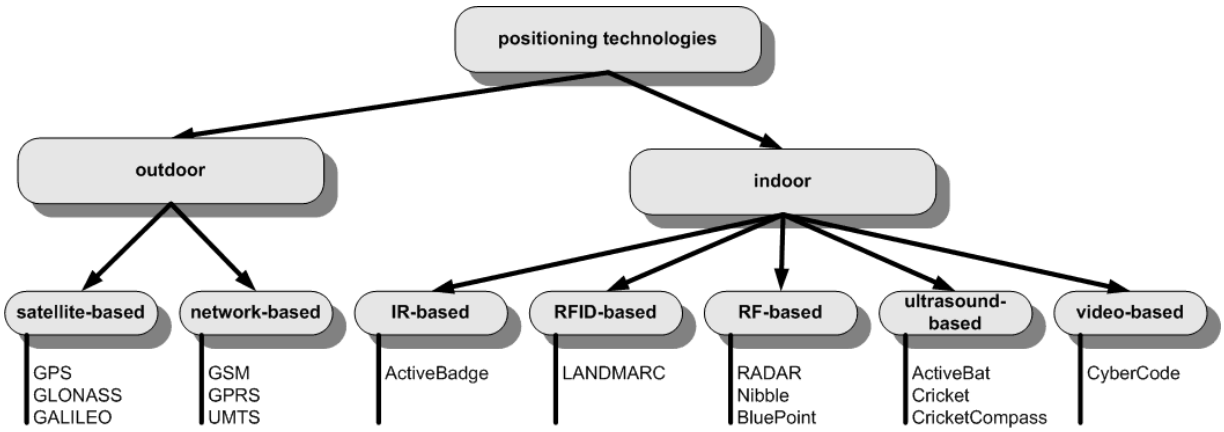


Figure 2.3: *Overview of positioning technologies*

Apart from the already mentioned and in the community for mobile computing matured systems – Figure 2.3 gives an overview over the systems – a lot of additional and partially exotic technologies exist. For example, MOTIONSTAR [8] uses magnetic field sensors for location determination. The PAL[43] [49] system was developed to assess the capability of ultra-wideband (UWB) geolocation. UWB is a developing communication technology that delivers very high-speed wireless data exchange across relatively short distances with a low-power pulse transmission.

Positioning Systems

Together a positioning technique and a positioning technology result in a powerful positioning system. This system can determine the location of a user or device in a certain area. Despite the different characteristics of the underlying technique and technology a positioning system does have its own features. In a way the positioning system has to deal with diverse characteristics originate from the used methods. The other way around applications does have different requirements to a positioning system.

A *geographic coordinate system* expresses every location on Earth by two coordinates. Latitude is the angle between any point and the equator. Longitude is the angle east or west of an arbitrary point on Earth. By combining these two angles, the horizontal position of any location on Earth can be specified. The *Cartesian coordinate system* is used to uniquely determine each point through three numbers. Each number lies on one of the x-, y-, and z-axis with its origin is the Earth centum. Whatever coordinate system is used, a global location information needs an *absolute* coordinate system. Sometimes a *relative* position declaration is more useful, because it represents the location of a user in reference to an observer. Thus it is not unique and not the same or all possible observers.

Indoor as well as *outdoor* positioning systems have besides the intention of location determination only a few things in common. Therefore, a distinction between both types is useful. In most instances completely different hardware and different approaches are used. Outdoor

[43] Precision Asset Location

systems aspire at wide coverage like with satellite technology. Indoor positioning systems usually intend or a high accuracy in difficult environments.

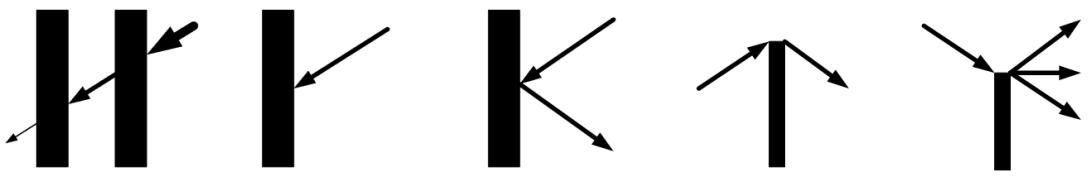
The location information returned by a positioning system can either be *physical* or *symbolic* [54,73]. Physical location can be expressed as mathematically-based coordinates like for example a person is positioned at $42^0 23' 49'' N$ by $23^0 27' 76'' W$ at a 12-meter elevation. Symbolic position information describes abstract ideas of an objets, e.g. "in room F2.104 at Fuerstenallee 11".

The last but most important characteristic is the *accuracy*. The positioning technology comes along with a error in measurement like any other physical measurement. This error results in an inaccuracy depending on the current environment. Locationing at different times on the same location can obtain in different measurements. User and application have to be aware of this inaccuracy.

2.2.3 WLAN Positioning Features

The previous section described many different technologies to realize an indoor wireless positioning system. As a result WLAN seems to be the most promising solution. In special use cases alternative methods favorable, but in most assignments the following arguments are accomplished. First of all, to apply an indoor positioning system acquisition and maintenance costs for the needed infrastructure have to be low. Secondly, the accuracy has to fulfill the requirements for all intended scenarios. The positioning technique providing location-based services is usually in addition to the regular wireless communication. For complex location-based services an adequate data network is needed furthermore. Altogether WLAN is the first choice to assure most LBSs scenarios.

In [129] we examined the suitability of WLAN as positioning system for LBSs. Unfortunately, several features of the wireless network influence the attributes used for positioning, some advantageously some not. Radio frequency signal propagation in the spectrum used by WLAN underlies attenuation, shadowing, reflection, refraction, and scattering effects [139]. The *attenuation* at increasing distance is very useful, but is accompanied by attenuations caused by walls, windows and floors as well as atmospheric influences like changing humidity levels. An extreme form of attenuation or absorption is *shadowing*. Obstacles like mountains, buildings or vehicles disturb be propagation of the RF signal. Indoors e.g. solid steel in thick walls has the same effect. Moreover, for high frequencies (WLAN uses 2.4/5.0 GHz, mobile phones e.g. 9.9 GHz) even obstacles like the water in human bodies cause shadowing effects.



(a) Attenuation (b) Shadowing (c) Reflection (d) Refraction (e) Scattering

Figure 2.4: *Radio-frequency signal influences*

Signal *reflection* depends on the objects and its material due to the physical laws. As an advantage, transmitter and receiver do not need a free line-of-sight. On the other hand multi-path signal propagation is the consequence of reflection. Shadowing and reflections are caused by objects much larger than the wavelength of radio frequency signals. Refraction and scattering are wave-effects when dealing with objects in magnitude of the emitted wavelength (GSM for instance has a wavelength about 10cm). After a refraction usually at an edge of an obstacle the direction of the originally signal is slightly changed. The last effect to mention is scattering and produces multiple weaker signals in different directions - like white light is split into rainbow colors. Because all effects can occur simultaneously their signal strength, direction and further propagation is hardly to predict. Figure 2.4 illustrates all signal influences.

2.3 Computer-supported Cooperative Work

Previous sections argued about context-based applications and as a more restricted approach location-based applications or services. Technical aspects like positioning techniques and positioning technologies are important to understand the technological possibilities in this area, but in the same way *application scenarios for context-based systems* contribute important aspects. They give an impression on how the technology is used, which convenience they provide, and, most important, motivate to further steps in application and technology development.

In the case of LBS the classification in section 2.2.1 gives an first impression on possible application areas. This work started with an collaboration scenario building virtual communities which illustrate vividly the convenience and power of complex context-based or location-based services. A more general term for this kind of scenario is *computer-supported cooperative work* (CSCW) [60]. It combines the understanding of the way people work in groups with the enabling technologies of computer networking, and associated hardware, software, services and techniques [161]. Therefore, CSCW-systems consider collaboration and coordination issues, as well as coordination and communication in computer systems. They collect and aggregate information about users, which cannot be derived directly from user communication or behavior separately.

Applications in the CSCW-area are typical examples for context-aware systems. A group of users can cooperate and share their knowledge in many different ways. In a cooperative relationship context information is an important source of information. It can be used to automate processes and provide convenience for communication and collaboration. For example, a group of students meet in a room to work on the same task, or a teacher is lecturing his/her class. Nowadays, massive usage of modern hardware and software supports cooperative collaboration in these examples. Mobile devices like PDA's and notebooks as well as high-end equipment, such as electronic whiteboards or video conferencing systems, and complex software systems like Lotus Notes are used. With this infrastructure electronic documents and other materials are shared among the participants on a media-level. Contact information can be shared on a personal-level to keep contact after meetings. All these materials and information build different contexts as described in section 2.1.

An example, especially in a mobile environment, is the notice and identification of potential collaboration partners. It is essential for a successful cooperative work. A collaboration can be established by filtering user contexts for location information. The relative proximity to other users can for instance lead to the foundation of a collaborative group (e.g. all users sitting around the same table). Even without the knowledge of the absolute position information, this can accomplish a useful support for cooperation processes.

With location-awareness social relationships among participants of the collaborative environment can be created. Different roles and privileges can be defined depending on each user's context. The manual assignment of user roles and privileges in a collaboration scenario is a complex task, where the administrative overhead can be reduced by using context information. For example, is a user located at the speakers desk and other users are located as a group around a table, the roles of moderator and audience as well as their resource access authorization can be automatically assigned easily. In [128] we introduced for example a location-aware file-sharing service which can be accessed only in a certain proximity. Therefore, resources can be easily shared in an ad-hoc collaboration group.

CSCW-systems are a good choice to illustrate the benefits and capabilities of context- and location-aware applications. They work in a mobile environment and have highly interactive and cooperative processes demanded by groupware systems [45]. Other reasons are the possibility to implement complex and adaptive services, which introduce new challenges.

In [90] we introduced a proximity-based technique for virtual community collaboration. It takes advantage of recent advances in wireless technology and the increasing demand to extensive user mobility. It senses automatically the proximity of mobile users and offers a flexible peer-to-peer platform for collaboration services. The system works with the infrastructure mode of IEEE 802.11 wireless access points to create different zones according to the ranges of each access point. A user is registered automatically when in range of one access point and is mapped to its zone. In this virtual community interaction with other members can take place without any prior exchange of contact information or agreeing on communication to a dedicated server. The collaboration platform supports the user by providing a background communication over an universal service protocol. It can easily integrate several tools and custom services to support cooperation and collaboration. No central component, which is usually needed in CSCW-systems, is required. The collaboration is based on peer-to-peer communication which supports the typical no resident behavior in a mobile collaboration scenario. Figure 2.5 illustrates the location-based approach for a collaboration platform.

In [89] we presented the implemented protocol with more detail and also introduced first services for the platform. A messaging service, which is an important part of any CSCW-system, fulfils the basic communication requirements. The status, role and identification of each user can be comprehended with the displayed user status and group structure of the current session within the messenger's window. Messaging and file transfer are designed in the usual look-and-feel of well established messengers to provide an intuitive user interface. A second service implements the document sharing idea mentioned above.

A CSCW-systems acceptance is based on its productivity and usability. In [88] we introduced therefore more complex services such as audio/video module and a shared-whiteboard extension to increase the systems capabilities. Subsequently, now each member can share the new services to collaborate on difficult tasks. A group can now also be characterized by its

functionality (e.g. ongoing video conference) or particular security settings (e.g. private whiteboard sessions), where the user has to ask for invitation, before he/she can enter the forum.

With ongoing work we discovered major problems sharing documents, distributing audio and video stream, and reliable mobile communication on the peer-to-peer based platform. Due to hardly predictable transfer rates and node mobility in wireless networks, especially in ad hoc and peer-to-peer structures, we developed specific routing algorithms [82, 83]. DHT[44] are commonly agreed to be a good approach when storing information on a peer-to-peer based network [134, 147]. However, existing algorithms are often limited to $1:1$ communication relationships and are thus not suitable for typically resource sharing, which is essential in CSCW-systems. Our reactive routing algorithm for sharing resources in wireless ad hoc networks is based on swarm-intelligence. In opposite to existing approaches, it gives estimate initial pheromone values and introduces $n:m$, $n,m \geq 1$ communication. The system builds lowest stretch routes and considers fault tolerance and load balancing aspects. For more details, see [82, 83].

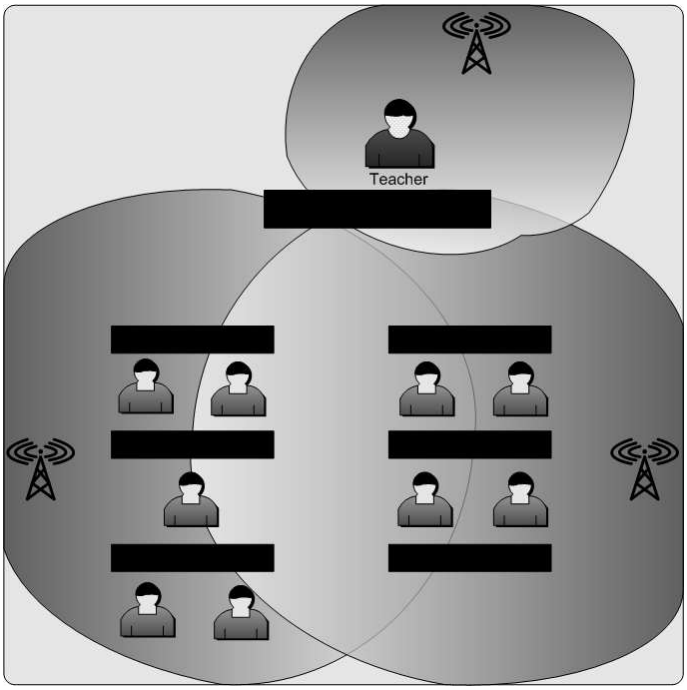


Figure 2.5: *Classroom example where location information defines collaboration roles and privileges*

Further attempts to provide more capabilities, usability, and overall convenience in the usage of a CSCW-system, lead to ongoing thoughts in location estimation accuracy, automated service discovery and better service matching for user requirements. The next chapters consider these aspects discussing semantic issues.

[44] Distributed Hash Tables

2.4 Summary

In this chapter we introduced context-awareness. Context related terms were defined and modeling approaches for context information were discussed. Filling context descriptions with content turned out to be eventually incomplete and ambiguous. Sensing the context automatically can overcome these drawbacks. We focused on the location information as major component in a context resulting in a detailed review of location-based services. After classification and characterization of this type of services an overview of current positioning techniques and technologies was given.

To develop highly interactive and complex context-aware services useful to the remaining service matching aspects in this thesis, the indoor positioning via WLAN signal strengths was identified as an interesting approach. Here, devices like PDAs and notebooks can be localized without additional hardware components. A short review of the application area computer-supported cooperative work was given at the end of the chapter. CSCW is one application scenario where indoor location-based services are very useful.

3 Semantic Web Services

> We've all heard that a million monkeys banging on a million typewriters will eventually reproduce the entire works of Shakespeare. Now, thanks to the Internet, we know this is not true.
>
> Robert Wilensky

The WWW[1] has changed the way people communicate with each other and the way business is conducted. It lies at the heart of a revolution that is currently transforming the developed world towards a knowledge economy and, more broadly speaking, to a knowledge society. Originally computers were used for computing numerical calculations. Currently their predominant use is for information processing.

Most of today's Web content is suitable for human consumption. Even Web content that is generated automatically from databases is usually presented without the additional structuring information found in databases. Keyword-based search engines, such as Google[2] and Yahoo[3], are the main tools for using today's Web. It is clear that the Web would not have been the huge success it was, were it not for search engines. However, despite improvements in search engine technology, the difficulties remain. The amount of Web content outspaces technological progress [97].

An alternative approach is to represent Web content in a form that is more easily machine-processable and to use intelligent techniques to take advantage of these representations. The Semantic Web, propagated by the W3C[4], is an initiative that will accept this challenge. It is important to understand that the Semantic Web will not be a new global information highway parallel to the existing WWW; instead it will gradually evolve out of the existing Web.

[1] World Wide Web
[2] `www.google.com`
[3] `www.yahoo.com`
[4] World Wide Wed Consortium

3.1 Understanding Web Services

Today's Web content is very often presented in form of Web services. They do not merely provide static information, but involve interaction with users and often allow users to effect some action. For example, user interaction with an online music store involves searching for CDs and titles by various criteria, reading reviews and listening to samples, adding CDs to a shopping cart and buying it by providing credit card and shipping details. Web services accomplish these tasks in a brilliant way. The task of creating and deploying Web services is not all that different than what developers currently do for traditional Web applications. The tendency on all platforms is to automate as much as possible in creating Web services under interoperable standards. Web services are software applications that can be *discovered*, *described*, and *accessed* based on XML[5] and standard Web protocols over networks.

Several definitions range from very generic to very specific. Basically: *a Web service is a collection of protocols and standards used for exchanging data between applications* [1]. More detailed: *Web services are self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces* [154]. Very specific: *a Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols* [22].

The concept of Web services is not new. It has been around since the early days of the Internet. Distributed objects in order to created interactive distributed systems were developed by several companies under different names. Microsoft called them DCOM[6], IBM called them SOM[7], and Java RMI[8]. The company consortium Object Management Group converged upon the standard called CORBA[9] [112]. Their result is a standards-based open interface on which distributed client/server applications can be developed. Web services offer the same features, but achieved a break-through by using platform-neutral, Internet-scalable, and language-independent standards [1].

Besides the use of open and free standards [68] other advantages for Web services exist. The communication over TCP/IP is usually done via the mature HTTP[10] protocol that avoids problems with firewalls occurring e.g. in CORBA, DCOM, or RMI systems. Web services can encapsulate other services to reuse older or proprietary services. Platform-neutrality and language-independency hide system heterogeneity. Altogether a flexible and established architecture evolved [1]. On the other side Web services have a problem with performance. The heavy use of XML requires parsing partially big data files and the considerable overhead for administration have a negative influence on the performance. In the field of security aspects a lot of catch-up work has been done recently [13, 67, 79, 106], but making integrity, confidentiality, authentication, and authorization broadly available needs still some efforts [74].

Web services already brought new ideas and influences to EAI[11] [65, 76, 96] and Grid-

[5] eXtensible Markup Language
[6] Distributed Component Object Model
[7] System Object Model
[8] Remote Method Invocation
[9] Common Object Request Broker Architecture
[10] Hypertext Transfer Protocol
[11] Enterprise Application Integration

Computing [31, 53]. The main application area is doubtless the business-to-business sector where business processes can be undertaken beyond company borders [28].

Web services are usually associated with different roles if taken into action [42]. In a SOA[12] architecture [110] a *service provider* implements and provides a specific service, which must be published in a *service registry*. The *service consumer* can invoke a service after looking it up in this registry. The dynamic binding between consumer and provider is done by exchanging further information via open protocol communication. In [128] we examined this aspect for location-based services.

Completing this high-level overview of the basics of Web services questions about service access, description and discovery have to be answered. SOAP [107] is the "envelope" that packages the XML messages that are send over HTTP between clients and Web services. As defined by the W3C, SOAP is *a lightweight protocol for exchange of information in a decentralized, distributed environment*. It provides a standard language for tying applications and services or services and services together. The syntax of SOAP is fairly simple and contains the envelope that wraps the message, a description of how the data is encoded, and the application-specific message itself in a so-called body. Request and response messages are exchanged which can get complicated. Luckily, developers do not necessarily have to understand the details of SOAP. Many tools, often integrated in Web service development tools, create SOAP handlers automatically.

Whereas SOAP is the communication language of Web services, WSDL[13] [23] is the way communication details and application-specifics are described. The W3C defines WSDL as *an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information*. To know how to send messages to a particular Web service, an application can look at the WSDL and dynamically construct SOAP messages. WSDL describes the operational information: where a service is located, what the service does, and how to invoke it. The format is not intended to be human-readable, but toolkits like [4] create WSDL to existing Web services automatically.

If the location and function of a Web service is known, applications can easily communicate with them. However, for a dynamic search for Web services based on the features they provide, a registry is needed. UDDI[14] [154] was created to facilitate the discovery of business processes. It is an evolving technology being implemented and embraced by major vendors. Organizations can register public information about their Web services and types of services with UDDI. The company's contact information, business categories, and technical information about the services are exposed. UDDI is limited to "browsing" the information to discover relevant services. Obviously, there are security concerns about placing information in a public registry. Much like public Internet Web pages, a organization has to decide what information should be published. Although the limited flexibility and capabilities of UDDI leaves a lot functionality to be desired, private UDDI registries are very common for internal integration in huge companies [43, 98].

Web service technology has become a major building block or business processes over the Internet. The used free and open standards have matured and are well established in orga-

[12] Service-Oriented Architecture
[13] Web Service Description Language
[14] Universal Description, Discovery, and Integration

nizations environments. Great potential for current research lies in the area of semantics for Web services. The Semantic Web and Web services go hand in hand. XML, a self-describing language, and WSDL, a language that describes the SOAP interfaces of Web services, are not enough. Automated support is needed in dealing with numerous specialized data formats. Ontologies are the key enabling concept for the Semantic Web and Semantic Web services, interweaving human understanding with machine processability.

3.2 The Semantic Web Vision

What is the Semantic Web? The word *semantic* implies meaning. For the Semantic Web, *semantic* indicates that the meaning of data on the Web can be discovered – not just by people, but also computers. The driving force on the Semantic Web initiative is TIM BERNERS-LEE, the very person who invented the WWW in the late 1980s. He expects from this initiative the realization of his original vision of the Web, a vision where the meaning of information played a far more important role than it does in today's Web [3]. In his vision [17] software as well as people can find, read, understand, and use data over the World Wide Web to accomplish useful goals for users. People surf the Web, read the labels on Hyperlinks, and decide which link to follow. A machine cannot do this, yet. In brief, the Semantic Web is supposed to make data located anywhere on the Web accessible and understandable, both to people and to machines.

When in 2001 TIM BERNERS-LEE published his vision of the Semantic Web in the well known Scientific American article [17], a lively discussion about the concept arose and is still ongoing. In this vision, he describes the principles of what he proposes as the successor of the World Wide Web as we know it today. The key aspect is to make the information on the WWW understandable for computers by creating semantic annotations in addition to the human-readable information presented on Web pages. The main components of the Semantic Web, are firstly *structured annotations* on Web pages (expressed in XML), which extend the traditional WWW and enable applications to capture certain facts. These facts can be linked to further information contained in *ontologies*, which make it possible to determine the "meaning" of the information and to deduce further information by using inference rules. Concepts stored in ontologies can be identified by using *URIs*[15] as identifiers. URIs provide a way to create bridges from one ontology to another by linking certain concepts that exist in both ontologies. At last, *agents* are taking actions, which are computer programs or scripts that act on behalf of a human and are usually considered as electronic "personal assistants". Significant properties of agents are *autonomy* of action, *ability to communicate*, *adaptiveness*, *learning aptitude* and *mobility*. Within the Semantic Web scenario, agents are expected to carry out tasks by communicating and negotiating with each other, using ontologies and inference capabilities to establish a common but dynamically vocabulary.

To realize this vision different technologies were developed to fulfill the required aspects step by step. Instead of mixing layout and content together in HTML[16], XML allows to define

[15] Uniform Resource Identifiers
[16] Hypertext Markup Language

own data structures and separate layout from content. RDF[17] encodes assertions about specific things written in XML in form of simple triples. These triples form webs of information about related things. With OWL[18] the common understandings of certain concepts and ideas is possible. More complex relationships than in RDF can be expressed. For a specific domain the common terminology and the concepts are shared and can be easily referenced.

3.3 Syntactic Standards

In order to bring the vision of the Semantic Web into being, the research community created standards, recommendations, development frameworks, APIs[19], and databases. These languages and frameworks evolved and matured over several years and form the major building blocks for the Semantic Web and the foundation for further steps in this thesis.

3.3.1 Visualizing Information with HTML

Creating a Web page on the Internet is currently the most frequently and extensively used technique for sharing information. These pages contain information in form of free and structured text, images, and possibly audio or video sequences. The HTML language is used to create these pages. The language provides primitive tags that can be used to annotate text or embedded files in order to determine the order and style in which they should be visualized. It is important to note that the markup provided by HTML does only by accident refer to the content of the information provided. Usually it covers only the way it should be structured and presented on a page. It was created to make information processable by machines, but not understandable. The freedom of saying anything about any subject led to a wide acceptance of this technology. However, the inherent heterogeneity is a most challenging problem giving systems access to this information. XML is one way to handle this heterogeneity without reducing the “freedom” too much.

3.3.2 Exchanging Information with XML

In order to overcome the fixed annotation scheme provided by HTML, XML was proposed as extensible language allowing users to define his/her own tags to indicate the type of content annotated by the tag. It turned out that the main benefit of XML actually lies in the opportunity to exchange data in a structured way.

The XML specification [24] provides a formal grammar used in well-formed documents. An XML document is valid if it has an associated type definition in a DTD[20] file and complies with the grammatical constraints of that definition [157]. In a document each element is delimited by a start and end *tag*, a *type* and a set of *attributes* consisting of a name and a value. The additional constraints in a DTD refer to the logical structure of the document including tag nesting and tag allowance/requirement. Better than DTD’s, XML Schema [46] provide a

[17] Resource Description Framework
[18] Web Ontology Language
[19] Application Programming Interface
[20] Document Type Definition

much more comprehensible way to define the structure of an XML document. Elements used in a XML Schema definition are of the type “element” and have attributes that define restrictions and default values to be used when no attribute value is defined. The information within such element is simply a list of further element definitions. XML Schema support constraints on attributes, provide a sophisticated structure, and use the namespace mechanism allowing combinations of different schema [46]. It is possible to encode rather complex data structures and exchange information across different formats. However, a user must commit to a data model (XML schema) in order to make use of the information. Therefore, it lacks an important advantage of meta-information, which is provided in RDF. In addition, many tools such as parsers have been developed that enable the information exchange between applications. However, there is no inherent meaning associated with the nesting of the XML elements. It is up to the application to interpret the nesting. Figure 3.1 shows two different XML documents expressing the same fact.

```
<device>
  <id>4223</id>
  <owner>Ulf</owner>
  <room>F2.104</room>
</device>
...
<device id="4223" owner="Ulf">
  <room>F2.104</room>
</device>
```

Figure 3.1: *Different XML documents expressing the same fact*

3.3.3 Service Interface Description with WSDL

To advertise and discover Web services a structured and non proprietary description is necessary. The Web Service Description Language (WSDL) [23] uses XML and XML Schema to provide a machine-readable description delivered together with the Web service itself. A Software program can consult this description to learn exactly how to invoke this Service, which inputs are required, and which outputs are generated.

WSDL has five first level elements: `type`, `message`, `portType`, `binding`, and `service`. The `types` provide data type definitions used in messages for data exchange. XSD[21] types are the default type system preferred by WSDL. Simple types like `xsd:string`, `xsd:integer`, or `xsd:boolean` are simply linked. Complex types consist of numerous simple or complex type recursively together with e.g. restrictions or control structures to accomplish a higher complexity. The WSDL `message` element represents an abstract definition of data to be transmitted as request or response. It consists of a number of parts referring to parameters and their types. The concrete operations of a Web service are defined with the `portType` element. Each `wsdl:operation` refers to input and output messages any their type. Types specify the mes-

[21] XML Schema

sage intentions for e.g. one-way, notification, or request-response usage. The `service` tag is used to aggregate a set of related `portTypes`. The wsdl:binding specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType. It specifies a style of interaction (for instance RPC[22]), the transport protocol (for instance HTTP), and the encoding style used along the operation definitions. Usually a SOAP binding is used provided by the `soap:binding` and `soap:operation` elements. The concrete inputs and outputs to invoke a specific Web service method are thus specified.

3.4 Resource Description with RDF

XML and WSDL are purely syntactical and structural in nature. It often encodes an application-specific data model. Consequently, further approaches have to look for a meta-model to describe information and define its meaning. The RDF specification [99] has been proposed as a data model for representing meta-data about objects using XML syntax.

The basic underlying model is very simple. Every type of information about a resource, which may be a Web page or an XML element, is expressed in terms of a triple (subject, predicate, object) so-called RDF *statements*. Thereby, the *predicate* is a two-placed relation that connects the *subject* (or resource) to a certain *object*. The object can be a data type, another resource or an untyped value (literal). Additionally, the value can be replaced by a variable representing a resource that is further described by linking triples making assertions about the properties of the represented resource. Using the *reification mechanism* we can make statements about facts (other statements) by nesting triples [99]. RDF also allows multiple values for single properties in collections (bag, sequence, and alternative). A problem arising from the nature of the Web is name collisions when something is referring to a different Web site, that might use different RDF models to annotate data. RDF uses *name spaces* to connect a name to a source that is then used to annotate each name in an RDF document defining its origin.

RDF is domain independent in that no assumptions about a particular domain of discourse are made. It is up to the users to define their own ontologies (see section 3.5) for the user's domain in an ontology definition language such as RDF Schema (RDFS) [25]. Unfortunately, the name RDF Schema is not a good choice, since it suggests that RDF Schema has a similar relation to RDF as XML Schema to XML. This, however, is not the case. XML Schema constraints the structure of the XML document, whereas RDF Schema defines the vocabulary used in the RDF data model.

For example, the ontology to express the fact above 3.2 has to define the concept of a "device" and the relationship "has name" in its vocabulary. Expressing the fact above as RDF statement needs a subject (a resource) to make a statement about. To be able to make a statement about a device the URI `http://lobass.com/device_id4223` is used. The predicate defines the kind of information to be expressed about the subject. The statement "hasDeviceName" is defined in an ontology and used be the URI `http://lobass.com/ontology#hasDeviceName`. The object defines the value of the predicate. Here, it's the name of the device "`nb-ulf2`". Figure 3.2 shows the triple put together in a complete document.

[22] Remote Procedure Call

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:lobass="http://lobass.com/ontology#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  <rdf:Description rdf:about="http://lobass.com/device_id4223">
    <mymusic:hasDeviceName>nb-ulf2</lobass:hasDeviceName>
  </rdf:Description>
</rdf:RDF>
```

Figure 3.2: *RDF syntax of an example statement*

3.5 Ontology Languages

RDF provides a way to express simple statements about resources, using subject-predicate-object triples. However, to use RDF we also need the possibility to define the vocabulary that is used in the RDF statements. This controlled vocabulary is also called *ontology*. For a given domain the ontology defines the concepts found in the domain, the relationships between these concepts, and the properties used to describe the concepts. Ontologies are defined in an ontology definition language. Good descriptions including introductions can be found in [47, 56]. In the following paragraphs the two ontology languages the W3C defined for the Semantic Web: RDF Schema and OWL are introduced.

Different approaches for capturing semantics have been developed in different scientific communities [61]. To argue for ontologies as a technology for approaching the problem of explicating semantic knowledge about information has several aspects. The sharing can be done at different *levels of formality* and different *extents of explication* [84]. Also the *translation process* between two vocabularies is an important aspect [150].

In general, each person has his/her individual view on the World and the things he/she has to deal with every day. However, there is a common basis of understanding in terms of the language we use to communicate with each other. The common understanding using this shared vocabulary lies on the idea of how the World is organized. This is often called "conceptualization". Using a shared terminology according to a specific conceptualization of the World remains unfortunately mostly implicit. Ontologies have set out to overcome the problem of implicit and hidden knowledge by making conceptualizations of a domain explicit. This corresponds to one of the most popular definitions in computer science for the term ontology, where *an ontology is an explicit specification of a conceptualization* [59]. An ontology is used to make assumptions about the meaning of a term available.

Typically, an ontology has two levels, the instance level and the terminological level, also called the *assertional box* (A-Box) and the *taxonomical box* (T-Box) in description logics. The instances in the A-Box model the concrete individuals in the specific domain, while the terminological level defines the abstract concepts and relationships between these concepts. A standard example is an ontology about families. In the T-Box, concepts like "person", "male", "female", "father", "mother", "grandfather", and so on are defined. Additionally, relationships between these concepts can be defined like "isChildOf", "isParentOf", etc. In the A-Box, the individuals and their concrete relationships are defined, e.g. "Alice isChildOf Bob" and "Alice

is female" etc. An important part of ontologies is the possibility to encode generic facts about the domain which is modeled. For the family ontology, these can be facts like "if A isChildOf B, then B isParentOf A", or "a mother is always female", or "if A is a brother of B's mother or father, then A is an uncle of B".

Reasoners and inference engines are tools that evaluate generic facts and draw conclusions from these facts. This makes information explicit which is only contained implicitly in the ontology. A reasoner may be queried via some query language to deliver instances and their values, as well as concept and attribute names on the ontologies known to the reasoner. There are a lot of types of formalisms to describe an ontology and therefore there are also numerous reasoners such as DL reasoners, rule engines, or first order logic provers [130].

3.5.1 RDF Schema

RDF Schema (RDFS) [25] is a simple ontology definition language that allows users to define the vocabulary needed to describe the resources in the domain with meta-data. To define the ontology RDFS uses the RDF triples format. In RDFS users can define classes, properties, and relationships to model the concepts in the domain. Terms that are defined in the RDFS language specification have the XML namespace prefix `rdfs:`; terms defined in the RDF specification have the prefix `rdf:`. A *class* is defined as a group of things with common characteristics like in object-oriented programming. Inheritance is realized when the subclass inherits the characteristics of a superclass. The classes used as resources in the ontology definition are `rdfs:Class`, `rdfs:Literal`, and `rdfs:Properties`. The concept of containers and reification can be used for a more clearly and strict modeling. A RDF Schema model also includes several property primitives. These are instances of the `rdf:Property` class and provide a mechanism for expressing relationships between classes and their instances or superclasses. The most important ones are `rdf:type`, `rdfs:subClassof`, `rdfs:domain` and `rdfs:range`. A full list of all classes and properties can be found at [25].

To get an impression of an ontology, a small example can help. Taking the domain of location-based services, some basic concepts (classes) such as "notebook", "PDA", "mobile phone", "mobile device", "owner" and "room" can be identified. As relationship between these concepts one can think of "has owner" and "is in room" for example. Subclass relationships between "mobile device" and the classes "notebook", "PDA", and "mobile phone" are obvious. As properties to describe the classes "has owner name", "is in room number", or "has device name" can be used.

Figure 3.3 shows the RDF visualization as graph of a simple ontology for the described location-based services example. In the ontology the resources `MobileDevice`, `Owner`, `Room` and `Notebook` are classes, because they are related to the `rdfs:Class` resource with the `rdf:type` property. E.g. the class `Notebook` is also declared to be a `rdfs:subClassOf` the class `MobileDevice`. The ontology further defines five properties. The `ownerName` property can only be used to describe an instance of the `Owner` class (and all its subclasses), since `rdfs:domain` is specified (room respectively). The `hasOwner` property can further only take objects as value, that are an instance of the `Owner` class, since `rdfs:range` is specified. The properties `ownerName` and `roomNumber` take a literal as object, since the range is defined to be a `rdfs:Literal`.

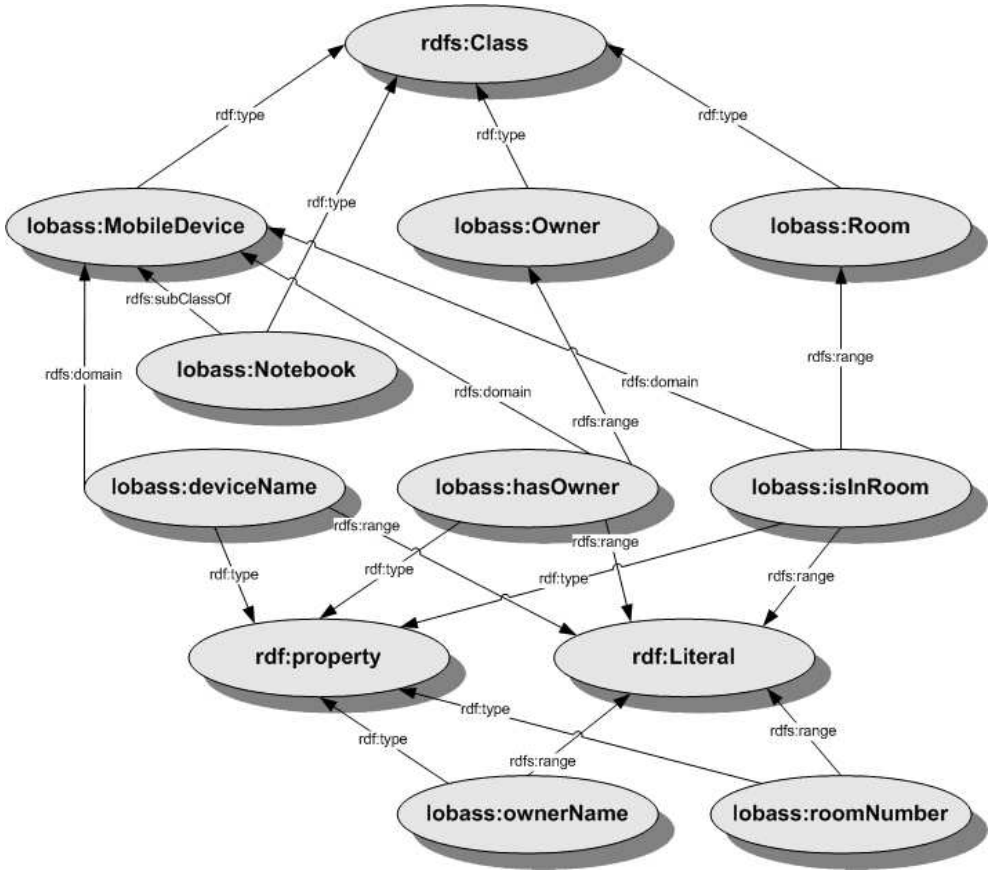


Figure 3.3: *RDF graph of an example ontology*

An RDF graph that uses this ontology is described in figure 3.4. The `hasOwner` property points to an *anonymous resource*, which is instance of the class `Owner`. Anonymous resources are used for resources that never be referred to directly from outside the RDF description. However, it is needed to represent an instance of the `Owner` class, that is described by the `ownerName` property.

The expressiveness of RDF Schema introduced in the previous section is very limited. RDF Schema can be used to define subclass hierarchies, properties, and domain and range restrictions of those properties [3]. However, a number of characteristic use cases that cannot be covered by RDF Schema are mentioned in the literature [70]. Researchers identified the need for a more powerful ontology modeling language. This lead to a joint initiative to define a more expressive language, called DAML+OIL[23] [34]. DAML+OIL was in turn the starting point for the W3C to define the Web Ontology Language (OWL) [15], now the recommendation for ontology definitions.

[23] Join of DARPA Markup Language (DAML) [146] and Ontology Inference Layer (OIL) [75]

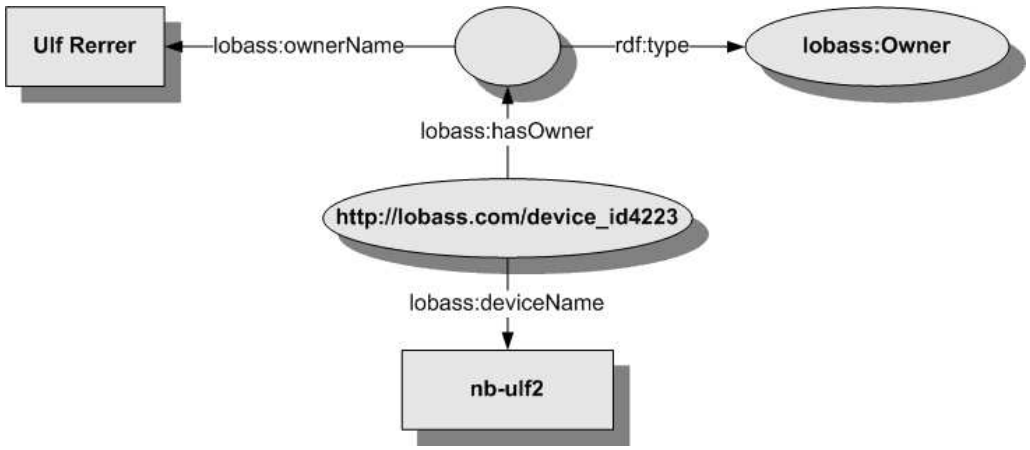


Figure 3.4: *RDF graph of an ontology that uses the above example ontology*

3.5.2 OWL

OWL provides another possibility to specify ontologies besides RDF Schema. It provides a number of additional modeling primitives that increase the expressiveness compared to RDF Schema [3]. For example the *local scope of properties* is one shortcoming of RDF Schema which can be expressed in OWL. If the property `eats` should be defined in the domain of `Sheep` and `Tiger` classes, the `rdfs:range` primitive cannot be used defining that a `Sheep` only eats plants, while a `Tiger` only eats meat. *Disjoint classes*, such as the classes `Male` and `Female` cannot be declared in RDF Schema. Also, *boolean combination* of classes are not possible. E.g. the class `Person` is the disjoint union of the classes `Male` and `Female`. RDF Schema does not provide *cardinality restrictions* such as a person has exactly two parents or other special *characteristics of properties* (transient (e.g. "greater than"), unique (e.g. "is mother of"), or inverse (e.g. "eats and is eaten by")).

The W3C followed two major goals while specifying the OWL language. First, to define an ontology language with the maximum expressiveness. Secondly, providing efficient reasoning support. Since these goals hardly go together, three different sub-languages were defined, each geared to fulfill different aspects of the full requirements [15].

OWL Lite supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 (not existing) or 1 (existing). Quantifier like "all" and "some" exist also. It should be simpler to provide tool support for OWL Lite than its more expressiveness, but also has the lowest formal complexity of all OWL sub-languages.

OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL got its name due to the correspondence with description logics. It includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class).

OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with all possibilities, but no computational guarantees. For example, in OWL Full a cardinality constraint on the class of all classes can be imposed, limiting the number of classes that can be described in an ontology. It is unlikely that any reasoning software will be able to support complete reasoning for every feature in OWL Full.

Regarding the above example of an ontology for a location-based service system (see figure 3.3), this ontology can be defined in OWL. OWL documents usually become large very fast. The complete OWL ontology can be found in the appendix. Here, only a few impressions on the possibilities of OWL are shown. Figure 3.5 shows at first the definition of the class `Notebook`

```
<owl:Class rdf:ID="Notebook">
  <rdfs:subClassOf rdf:resource="#MobileDevice"/>
</owl:Class>
...
<owl:Class rdf:about="#Notebook">
  <owl:disjointWith rdf:resource="#PDA"/>
  <owl:disjointWith rdf:resource="#MobilePhone"/>
</owl:Class>
...
<owl:Class rdf:about="#MobileDevice">
  <rdfs:subClassOf>
    <owl:Restriction>
       <owl:onProperty rdf:resource"#deviceName"/>
       <owl:minCardinality rdf:datatype="&xsd;string">
         1
       </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class
...
<rdf:Description rdf:ID="4223">
  <rdf:type rdf:resource="#Notebook"/>
</rdf:Description>
```

Figure 3.5: *Ontology example in OWL*

as a `rdfs:subClassOf` of the class `MobileDevice`. It is defined as disjoint from the classes `PDA` and `MobilePhone` using the `owl:disjointWith` element. After that an example for a property restriction with `owl:Restriction` follows. The property `deviceName` is only allowed once for the class `MobileDevice`. OWL does not have any predefined data types; instead it allows one to use XML Schema data types such as in `rdf:datatype="&xsd;string"`. At last, instances of classes are declared as in RDF, as you can see in the closing example.

3.6 Semantic Web Services

At present, the use of Web services requires human involvement, as discussed previously in section 3.1. The Semantic Web vision, as applied to Web services, aims at automating the

discovery, invocation, composition, and monitoring of Web services by providing machine-interpretable descriptions of services. Besides the pure syntactic handling of Web services the combination of Semantic Web technologies and Web service technologies lead to a new and very powerful instrument: Semantic Web (enabled) Web services (SWWS). Figure 3.6 was introduced by FENSEL in [3] and illustrates the combination of technologies.

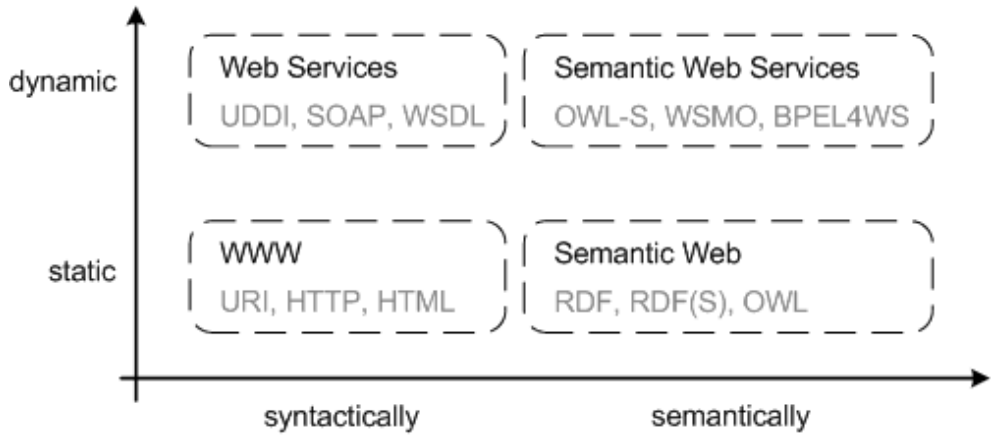


Figure 3.6: *Technology classification for semantic Web services*

3.6.1 OWL-S

Web sites and consequently Web services should be able to employ a set of basic classes and properties by declaring and describing services. Therefore, an ontology of services is needed. OWL-S [100] is an initiative that has developed an ontology language for Web services. It migrated from DAML-S [102] and makes use of OWL, so it can be viewed as a layer on top of OWL. Other languages such as WSML[24] [27], BPEL4WS[25] [2] or WSE[26] [104] exist to describe SWWS. They have strict intentions like BPEL4WS to model business workflows, or require complex reasoners like WSML. This makes OWL-S the most flexible and most accepted language to model SWWS. The ontology-based approach encompasses efforts to populate the Web with content and services having formal semantics. The ultimate goal of OWL-S is to provide an ontology that allows software to discover, execute and compose Web services automatically [100]. Currently the structure of OWL-S is threefold and consists of a service profile for advertising and discovering services, a process model which gives a detailed description of a service's operations and a service grounding which provides details on how to interoperate with a service via message exchange. These constructs describe a Web service in adequate details to automatically discover, invoke, compose and execute the service [100]. The following paragraphs briefly describe the particular elements. For more detailed descriptions see [100].

[24] Web Service Modeling Language
[25] Business Process Execution Language
[26] Web Service Enhancements

Profiles

In service discovery profiles are applied in two ways. On one hand, they are used by service providers to publish Web services. These profiles are called *advertisements*. On the other hand, profiles are used by a service requestor to describe the service to be searched for. During a regular discovery this request is compared with published advertisements to find matching services.

The `profile` class describes the properties of a service. This contains the description of the service and its *functional* and *non-functional properties*. The functional description is based on the transformation of data and states during the execution of a Web service. The profile specifies the inputs that a Web service requires, the outputs that it generates, the preconditions that must hold in order to execute the service, and effect that the execution causes. Therefore, it is known *what* a service does. The IOPEs (inputs, outputs, preconditions, and effects) are specified by referring to the classes `process:Input`, `process:ConditionalOutput`, `process:Precondition`, and `process:ConditionalEffect` of the process ontology.

Non-functional parameters are divided in two sections. First as semi-structured information intended for human readers that has no relevance for the semantic service discovery, e.g. `profile:serviceName`, `profile:textDescription`, etc. Second non-functional parameters can be specified as sub-classes of profile:ServiceParameter to incorporate additional requirements regarding service capabilities into the discovery process, e.g. security or quality-of-service requirements.

Process Models

The process ontology is used to define process models that describe the execution of a Web service in detail by specifying the data flow and control flow between particular methods of the Web service. Hereby, the service's application logic is presented to the outside. In order to achieve the results defined in a profile a user or a software has to execute the corresponding process model step by step considering all defined dependencies between inputs, outputs and preconditions, effects.

The execution graph of a process model can be composed of different types of processes and control constructs. OWL-S defines three classes of processes. From the view of a caller *atomic processes* are executed in a single step which corresponds to the invocation of a Web service method. *Simple processes* are used to specify abstract views of concrete processes and are thus not executable. *Composite processes* are specified through composition of atomic, simple, and composite processes recursively. Control constructs define the specific execution orderings on the contained processes. `Unordered`, `Sequence`, and `If-Then-Else` are the most used constructs here.

Groundings

The previously described profiles and process models serve as abstract specifications of Web service characteristics. The grounding enables concrete communication with a Web service by binding abstract inputs and outputs of atomic processes to message formats. It also pro-

vides XSLT[27]-Stylesheets for transformations between XML and RDF. The service grounding defines *how* atomic processes are invoked over WSDL operations and hence executed.

OWL-S defines exemplarily an ontology or grounding process descriptions to WSDL. Three corresponding elements of WSDL and OWL-S are described. Firstly, an atomic process corresponds to a `wsdl:operation`. Secondly, inputs and outputs of an atomic process are referred to `wsdl:parts` in the input and output message definitions of WSDL. Thirdly, the types used in the inputs and outputs in OWL-S correspond to the concept of abstract types in WSDL.

A `WsdlGrounding` contains one `WsdlAtomicProcessGrounding` for each atomic process of the corresponding process model. `WsdlOperationRef` defines the access to the Web service methods using properties `operation` and `portType`. Additionally, the WSDL messages are referenced using `WsdlInputMessage` and `WsdlOutputMessage`. The mapping of parameter types is defined using `rdf:List` of `WsdlInputMessageMaps` or `WsdlOutputMessageMaps`. OWL-S considers two ways of referring the appropriate OWL representations. Firstly, the Web service can refer to the OWL class representing the parameter type directly by `owlsParameter`. Secondly, for contemporary Web services OWL-S uses XSLT to convert parameter descriptions from OWL-S to XML Schema and vice versa.

Figure 3.7 shows an overview of all major classes in the OWL-S ontology. More detailed descriptions can be found in [100].

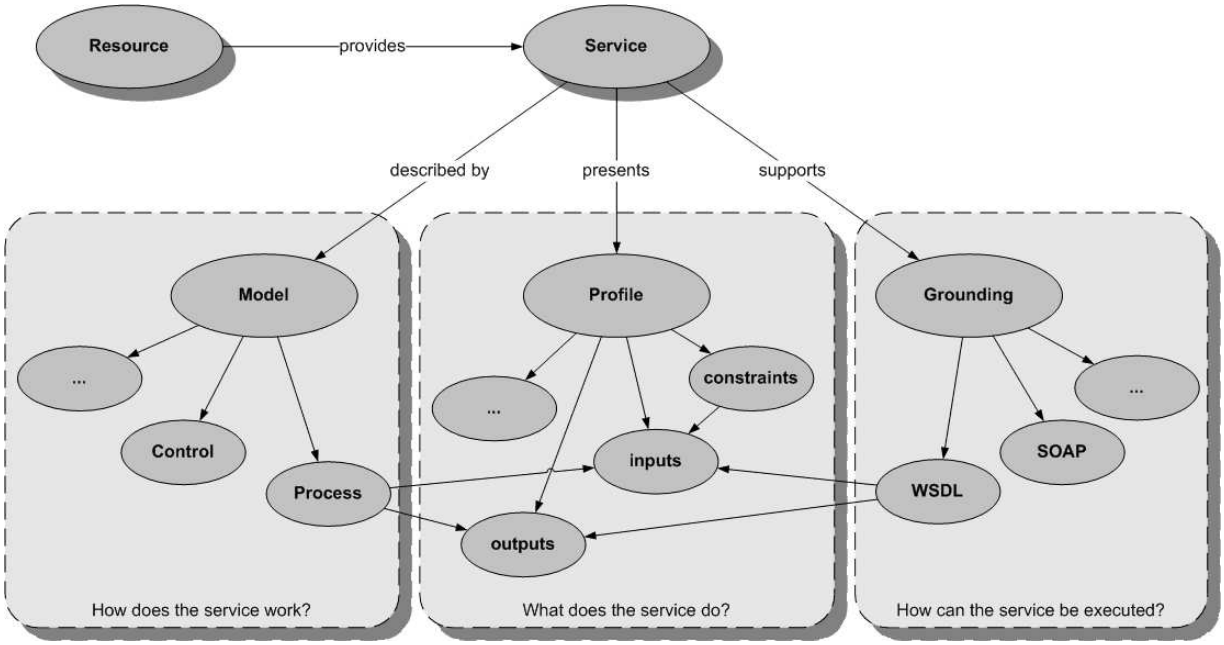


Figure 3.7: *OWL-S ontology overview*

[27] eXtensible Stylesheet Language Transformation

3.6.2 Service Discovery

The OWL-S profile is the main data structure used for service discovery. Its role is twofold. Both service requests and service advertisements are usually described as profiles. In order to search for suitable Web services the requestor creates a request containing functional and non-functional properties of the Web service looked for and sends it to a service registry. The service registry compares the request to the registered advertisements and returns the matching results. This process is called *matchmaking of services*. In a realistic application scenario it must be assumed that an advertisement does not match a request exactly in most cases. This thesis offers some solutions to this problem.

3.6.3 Service Execution

Very likely, different Web services which generate the same results may differ in their execution. E.g. a location-aware printing service require to register for a customer account before printing to set to default context environments, whereas others could require full customer information and settings each time. The information how a Web service must be executed in detail is provided by the process model.

To be able to execute an atomic process of the process model by invoking a Web service method a user must first allocate values for the required parameters and provide appropriate containers for the result to be returned. All necessary information for this task is either derived from the user's knowledge-base or enquired by interaction with the user. Hereafter, the user uses appropriate messages for communication with the Web service using the grounding definition. As part of this, the parameter values must be transformed into semantically lower representations, i.e. XML schema values in the case of SOAP. After successfully invoking the particular Web service method the user can process the results.

In the Semantic Web vision the execution of semantic Web service is usually done by agent representing users. They have a local knowledge-base specifying the users task and intentions described in shared and understandable terms due to the usage of ontologies. The agents are able to understand the requirements and results of a Web service. Together with the semantically service description the just mentioned tasks like invoking a Web service method, transforming the result, creating proper communication messages, and forming appropriate result containers are manageable tasks for this kind of agents. [36] gives an closer look on the capabilities and behavior of Semantic Web agents.

3.7 Summary

In this chapter we started with an introduction to Web services. With the presentation of the Semantic Web vision semantic Web services as combination of Web services and semantics became a very powerful instrument for machine-to-machine interaction. The following part reviewed the building blocks of this technology, discussing standards like XML, WSDL, and RDF capable of modeling and structuring knowledge. We introduced OWL as expressive tool to describe ontologies and identified OWL-S as perfect language to describe semantic Web services and populate Web services with content and formal semantics simultaneously. The

goal of OWL-S is to provide an ontology that allows machines to discover, execute, and compose Web services automatically.

4 Semantic Service Annotation

If there is no struggle, there is no progress.

Frederick Douglass

The aim of Web services is to make services available, reuseable and easily accessible to a broad community addressing humans and machines. As shown in section 3.1 the old-fashioned way to publish, discover, invoke and execute Web services is using a UDDI registry. The intentions of the Semantic Web are to make the services and information on the World Wide Web more accessible through machine-readable meta-data. The Semantic Web procedures help to understand services more detailed resulting in a better service discovery. Due to the connection via ontologies to commonly accepted knowledge the meaning of a service's function as well as the intention of a service requestor can be comprehended. This shared understanding allows service requestor and providers to talk in the same terms about their intentions. This enables a high quality service matching. Although major improvements in this field have been made in recent research activities, a fully functional semantic discovery is far away. This work concentrates on the different aspects *annotation*, *composition* and *contextualization* towards a semantic service matching, which is the important part of the service discovery.

The usual way for a service requestor to discover an appropriate Web service is to browse through registries which list all available services for a certain domain. Here, service providers publish their services and give manually created descriptions of the service's functions. The level of detail as well as the used vocabulary in the description is usually not prescribed or restricted. In UDDI for example, as the most famous service registry, this description can be written by an external consulter, an internal economist or the actual service developer himself. All of them can describe the same service properly, but use different styles, words and background knowledge. Therefore, it is hard to match the right service only with this description. The conventional way in service discovery is to classify the available services roughly into unchangeable and non-expandable service categories. After that only the service interfaces are described on a syntactic level specifying the numbers and types of service inputs and outputs. Often this interface description is done in WSDL and can be generated automatically having the source code of the service available.

To overcome this drawback of unusable service descriptions and use the additional information Semantic Web tools can be integrated in service registries mechanisms. This allows comprehending semantic service descriptions provided by the service publisher which can be matched against semantic queries from service requestors. Due to a common understanding of shared terms by using ontologies it is possible to match services and queries exactly assuming both sides deliver a complete, correct, and meaningful semantic description. On the publisher side the semantic service description needs to be created manually. No tools are available to automatically generate semantic service descriptions such as the WSDL tools to generate syntactic descriptions because the meaning of a service cannot be extracted pure syntactic interface descriptions or the source code itself. Editors like Protégé [95, 108] support developers in editing annotations or service descriptions, but only can check for inconsistencies or errors. Still, the description has to be made by hand.

The aim in this part of this work is to propose a framework for the Web service developer. This framework guides the developer through the whole development process starting at the service design stage up to actual programming and annotation process closing with the service deployment. This makes it possible to collect and deduce enough semantic information about the created service to automatically generate a semantic annotation.

4.1 Problem Definition

A semantic annotation to a service specifies inputs, outputs, preconditions, and effects precisely. It uses terms and concepts described in ontologies to describe the services. The relations within the ontology entities enables machines to process the interface descriptions easily. The meaning of the service is processable and false interpretations due to synonyms (e.g. “buy” and “purchase”) and homonyms (e.g. “order” in the sense of proper arrangement and “order” in the sense of a commercial document to a request).

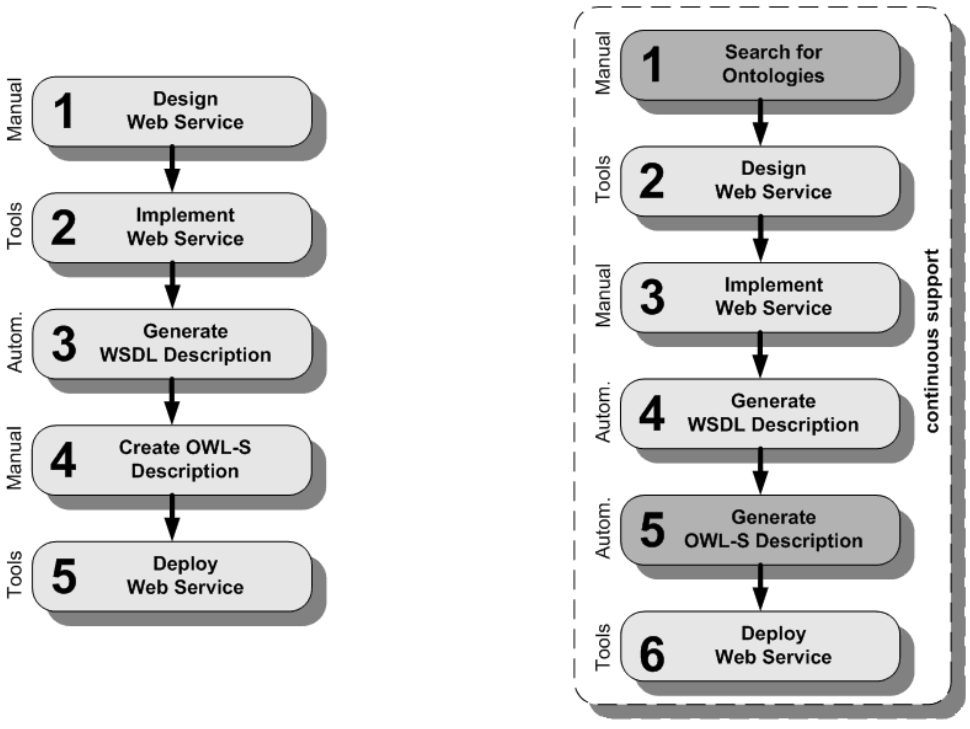
To a given Web service implementation it is possible to deduce a generic interface description. For example the JAVA2WSDL tool [4] generates automatically a WSDL document to a given java program. An automatized semantic service annotation, e.g. in form of an OWL-S document, is impossible only regarding the Web service implementation. The semantic information is not available or deducible directly from the source code. It is implicitly integrated and only humans may be able to figure out their meaning. But at the stage of the service development, the programmer can easily associate an appropriate ontology for the service’s domain. Now it is possible in most cases to assign the correct concepts in the ontology to the data types used in the Web service implementation. These data types are used to describe the service interfaces and therefore it is possible to deduce the semantic meaning of the service resulting in a new WSDL-to-OWL-S algorithm generating automatically a semantic annotation.

In the example of a *book selling* service, the developer specifies an ontology about book and an ontology describing business processes to the service. Then, terms like “book”, “credit card”, or “customer delivery address” and the relations between them are defined. An algorithm is now able to compare the simple and complex data types used in the WSDL interface description to the concepts described in the ontology. An automatically association is possi-

ble in most cases. If any ambiguities occur, a small selection of alternatives can be presented to the developer in order to continue the process. At the end, a fully semantically describe Web service is created without investing significantly more work. The following framework fulfills this purpose.

4.2 Framework for Semantic Web Service Development

A thorough annotated semantic Web service is the foundation of precise service matching. Useful annotations are usually made manually constructing WSDL and OWL-S descriptions after implementing the Web services. Several tools exist to support the design and implementation process. For some programming languages like Java tools automatically generate WSDL descriptions, but high level descriptions as OWL-S need to be created manually. This section presents a framework we introduced in [18] which supports developers in the annotation process during the service implementation and automatically generate OWL-S descriptions with the help of a new WSDL-to-OWL-S algorithm.



(a) Usual annotation procedure

(b) New annotation framework

Figure 4.1: *Comparison of the usual and the new semantic Web service annotation procedure*

The usual annotation procedure is done mostly by hand and therefore an error-prone and time-consuming work. The stepwise procedure is shown in figure 4.1 a) starting with the Web

service design, followed by actual implementation. After that tools can be used to generate WSDL descriptions directly from the source code. Subsequently, the developer can create a semantic annotation producing an OWL-S description using his/her background knowledge. This step can be assisted by several tools and editors, but the semantic information has to be entered manually in all cases. Finally, the annotated Web service can be deployed.

The framework described in detail in the following paragraphs consists of six steps guiding the developer from the conception stage up to a running and well annotated Web service. In comparison to the usual annotation procedure our framework can automatically generate the OWL-S description in step five according to a WSDL-to-OWL-S algorithm. Figure 4.1 b) shows that to collect the needed semantic information a continuous support during the whole development process is necessary. Furthermore, before beginning to design the Web service the developer has to pick appropriate ontologies in the specific domain of the service. This allows referring to known semantic concepts and terms during the whole procedure.

4.2.1 Obtaining Ontologies

Step one of the new framework covers the acquisition of relevant ontologies. It is likely that the service provider or maintainer has already existing semantic descriptions or ontologies for other purposes, but in the same domain of the service to be implemented. Thus, the shared knowledge within the provider's environment can be used and integrated in the service description. This method is also advantageous considering the fact, that many developers agree to the same facts and interpretations to reduce mistakes and inconsistency errors in the ontologies. Even if no such previous work is available to the developer, other open source meta-ontologies such as OPENCYC [113], WORDNET [33], and SUMO [111] exist or search engines like SWOOGLE [156] designed for semantic information can be used. With a set of ontologies the developer to revert to known concepts in the company's domain acting as knowledge-base.

4.2.2 Web Service Design and Implementation

The next two steps according to the framework are the design and the implementation of the Web service. Design refers to as the common way of programmers to model their approach with standard tools like UML before implementation. The new part realizing the continuously support is the use of the classes of the chosen domain ontology and thus its vocabulary to describe the service's program and communication. Still, in a semi-automatic way a programmer needs suggestions which ontology classes are more likely than others during the implementation. The suggestions depend on the selected names and structure of the used data types. With this help the implementation can be completed easily.

In [18] we used the programming language Java resulting in some adjustments during this step. The framework performs an OWL to Java transformation where for all relevant ontology classes and relationships corresponding Java classes are created like in [87, 108]. As OWL supports multiple inheritances, Java interfaces are used to maintain class relationships. Properties without a specific domain have their assessor function declared in the abstract interface "thing". Multi ranged properties are a problem as Java only supports one type variables. Lists of objects are used in this case. Other adaptations due to their complexity can be found in [87].

4.2.3 WSDL Description Generation

The next step cares about the creation of the WSDL document belonging to the just implemented Web service. The JAVA2WSDL tool [4] from the APACHE AXIS project can create a WSDL document, based on a Java file and is therefore used in this framework. The WSDL interface description enables SOAP communication like in all standard Web services and also allows the binding of the service to the corresponding OWL-S description. For every public Java method, a WSDL operation is created. WSDL only supports a syntactical description of the input and output messages. The values of the description consist only of data types and the message and operation names. What the service really does is not described in this way. But if the operation and complex type names in the WSDL description are unique during the framework process the automatic annotation algorithm (see section 4.3) can use these names to verify the correctness of the matching between the WSDL operations and the OWL-S simple processes. Then, atomic processes based on the matched simple process can be created building the connection to the ontologies and thus the meaning of the service. Further details can be found in the algorithm, see section 4.3.

4.2.4 OWL-S Description Generation

Based on the WSDL description from the previous step, step five of the framework generates the new OWL-S instance layered ontology document using a new WSDL-to-OWL-S algorithm described in section 4.3. As WSDL is simply an interface description only syntactical issues about the service are mentioned in its description. Semantic descriptions are involved at this stage. We assume, that the number of different meanings of a service is related to the associated category ontology. Each meaning is represented by exactly one simple process in the category ontology. Otherwise, the developer would have created a different category ontology in step one of this framework. To match WSDL elements to OWL-S components several steps are necessary.

First, WSDL complex types of the WSDL messages are matched with their corresponding OWL classes. If the lexical and syntactical structures of the created Java classes from step three of the framework are unaltered, they match exactly due to the generation procedure. If the developer has changed Java classes a check of the correct cardinalities of the OWL class properties is performed. If the matching fails, a new OWL class is created. WSDL messages with simple types can used untouched, because they are using the same XSD types from the XML-schema.

Secondly, WSDL operations are tried to match to OWL-S simple processes. For an exact mapping an atomic process is needed as mentioned before, but the available atomic processes origin from a category ontology the developer found in an archive or somewhere else. Thus, it is usually not possible to alter this atomic process to suit to the developer's special proposes. Therefore, the framework takes the best matching simple process from the category ontology and creates a more suitable atomic process in the OWL-S instance. This creation is only invoked if the simple process matches to the WSDL operation. To achieve this, the algorithm checks if every message parts of the input and output messages of the operation can be matched with exact one input and one output of the simple process. In other terms, ev-

ery input and output of the process has to fit to its corresponding counterparts in the WSDL operation. If successfully, we found a syntactical match for the WSDL description.

The remaining semantic parts are the preconditions and effects of the simple process. There is no way to be certain if the meaning of the simple process really matches the developer's intention. However, the developer chose carefully a suitable category ontology in step one of the framework. This reduces the possibility of misapprehension and wrong interpretations of the service's meaning.

After a successful matching a new OWL-S atomic process is created, based on the selected simple process. It heirs inputs and outputs of the simple process as well as the preconditions and effects. Consequently, the WSDL operation is annotated with a complete semantic description. If the matching fails, a new atomic process is created anyway. It satisfies the WSDL operation structure without taking care of the category ontology. Preconditions and effects have to be added manually.

Concluding activities to complete the OWL-S instance creation consider the service grounding and service profile. Based on the new atomic process an OWL-S grounding is created to bind to the matching operations. If the corresponding simple process refers to a service profile in the category ontology it is copied to the new OWL-S instance, but can of course be altered to be more suitable to the developer.

4.2.5 Accommodate Results, Testing and Deployment

In the final step of the framework the developer is involved to be sure the so far automatic procedures were correct and fit the human intentions and interpretations. We use the Protégé editor to check and if necessary alter the created OWL-S instance. In case of a failure in the previous step of the framework the atomic process description only contains the syntactical frame reflecting the WSDL contents. Unfortunately, the service developer has to add the missing semantic annotations manually here. Minor changes like the correct contact information or further service maintenance facts can be made at this stage easily.

The semantic Web service is now ready for testing and deployment. We use, due to the programming language Java, the APACHE AXIS project [4] in combination with the Web service container TOMCAT [5]. Thus, the Web service and its OWL-S description is published and invocation test such as an invocation by an OWL-S execution engine with simulated preconditions and effects and other trails are applied.

4.3 WSDL-to-OWL-S Algorithm

This section provides a deeper insight into the main step of the semi-automatic annotation framework, the WSDL-to-OWL-S algorithm. Other existing approaches like the WSDL2OWL tool [4] create only empty atomic process descriptions reflecting the in- and outputs of the WSDL document to the OWL-S layer. [85] and [117] only support users during the search for appropriate ontologies, and frameworks like [81, 92, 120] only guide through the manual annotation process and provide no automatisms at all. The intention of this new WSDL-to-OWL-S algorithm is to make as much annotations automatically as possible. This can only be

achieved by gathering additional information from the programmer during the development process. Therefore, the framework guides the service developer during all steps and provides selections of supposable choices in all major aspects.

The structure of our algorithm is illustrated in figure 4.2 and consists of four steps, *document parsing*, *type matching*, *process matching*, and *instance creation*. First the algorithm parses the available documents and extracts the necessary information. Then, it matches OWL classes and WSDL complex types. After that it matches WSDL operations with OWL processes. Finally, it produces a new OWL-S instance containing the Web service annotation.

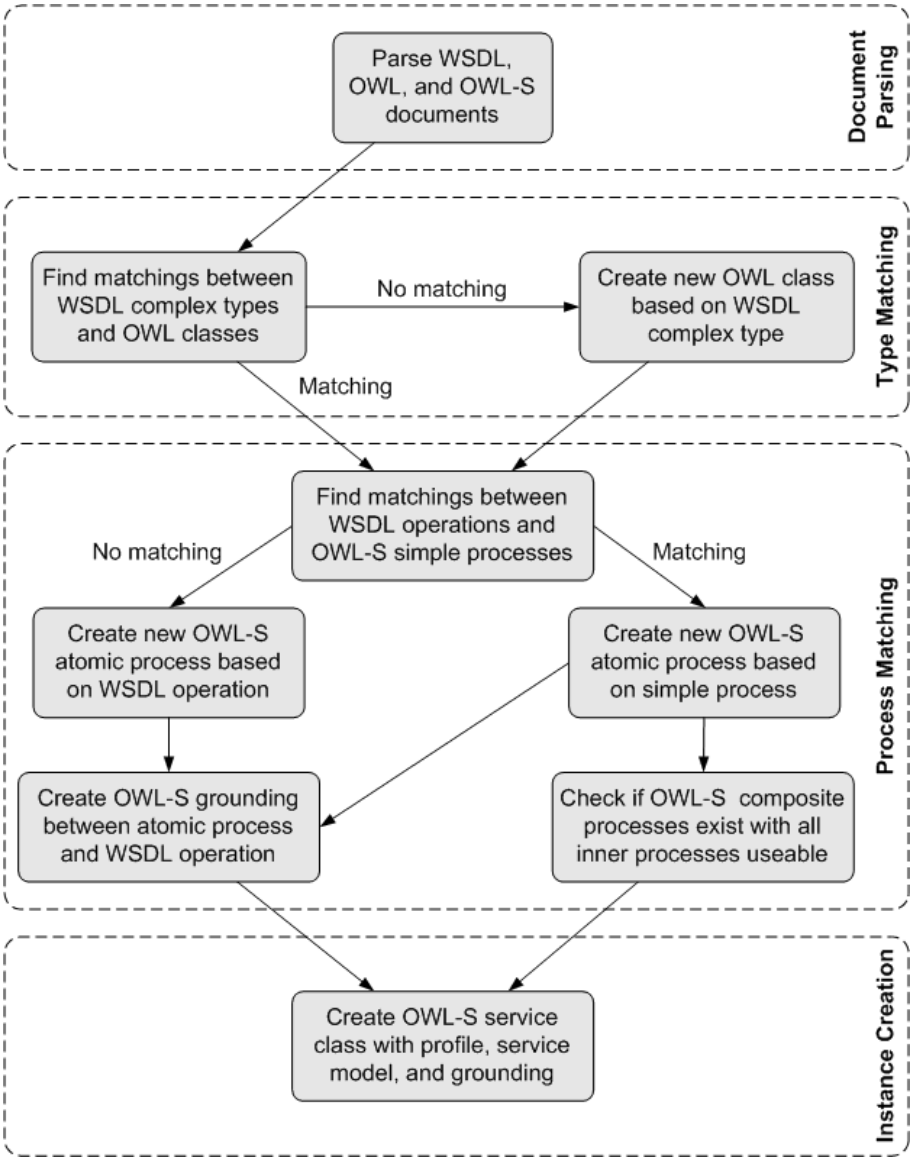


Figure 4.2: *Structure of the WSDL-to-OWL-S algorithm*

4.3.1 Document Parsing

The algorithm needs several documents to extract necessary information to work properly. In most parts the OWL-S API [143] for Java was used to read and store the parsed information into program variables. One essential document is the WSDL description of the Web service to be annotated. This description contains among other things the complex types and operations defining the input and output messages. A `wsdl:type` is used to describe complex types, which have OWL classes as counterparts. A `wsdl:message` contains `wsdl:part` tags with attributes `name` and `type`. The type attribute defines if its `simple` or `complex`. Simple message parts refer to a XSD schema type, where complex types refer to an ordered (e.g. sequence) number of elements such as again simple or complex types. The `wsdl:portType` tags contains `wsdl:operation` linking to the corresponding processes. `wsdl:input` and `wsdl:output` connect `wsdl:messages` to these operations.

The second input document is the OWL file. It describes the domain ontology anchoring the Web service to the shared understanding in the specific domain. Classes, properties, and restrictions are part of this file. Several classes in this ontology have the same syntax, but different meanings. The ontology is restricted to those classes, which are actually used to avoid problems during the matching phase of the algorithm. To connect an OWL class with a WSDL complex type every `owl:Class` construct is linked to an `wsdl:complexType`. All elements of the complex type are then linked to an `owl:DatatypeProperty` or `owl:ObjectProperty`.

The OWL-S category description is another input to the algorithm. There are simple processes, composite processes, and a profile description. If several simple processes with the same syntax but different meaning exist here, a restriction to the actually used processes simplifies the algorithm also. The algorithm only needs the input and output properties of atomic or simple processes `owls:hasInput` and `owls:hasOutput` from the very large OWL-S process class set.

Finally, the algorithm is able to use a default profile as input. It provides specific information for the service maintainer, which are usually the same for all Web services of the same organization. All these information from the WSDL, OWL, OWL-S, and profile files are hold in a data structure providing this data to the algorithm for further process.

4.3.2 Type Matching

With all necessary information at hand, the algorithm starts with the type matching procedure finding or creating OWL classes, based on WSDL types. Every message in a WSDL document has a type. For each type a corresponding class or data type has to be chosen in OWL-S. Primitive XSD types like `xsd:string` or `xsd:integer` are not converted as the same types are used in OWL-S. The structure and the syntactical naming of complex XSD types is compared with the structure and naming of the OWL classes. An exact matching is obtained if the structure and the naming is exactly the same of the `wsdl:complexType` and the `owl:Class` and both are linked together. No matching is reached if no conformity can be found at all. In the case of multiple matchings, more than one matching alternatives are possible. Here, the possible solutions are presented to the developer with the request to select the appropriate one. The selected OWL class is then linked to the WSDL complex type.

In the case of no matchings the `wsdl:complexType` has no relation to the given domain ontology. Thus, the Web service developer seems to have created a new Java class and therefore the algorithm creates a new `owl:Class`. Only a frame of the `owl:Class` can be provided automatically here, copying the name of the complex type from the Java class and arranging all simple XSD types and already matched complex types which are components of the complex type definition.

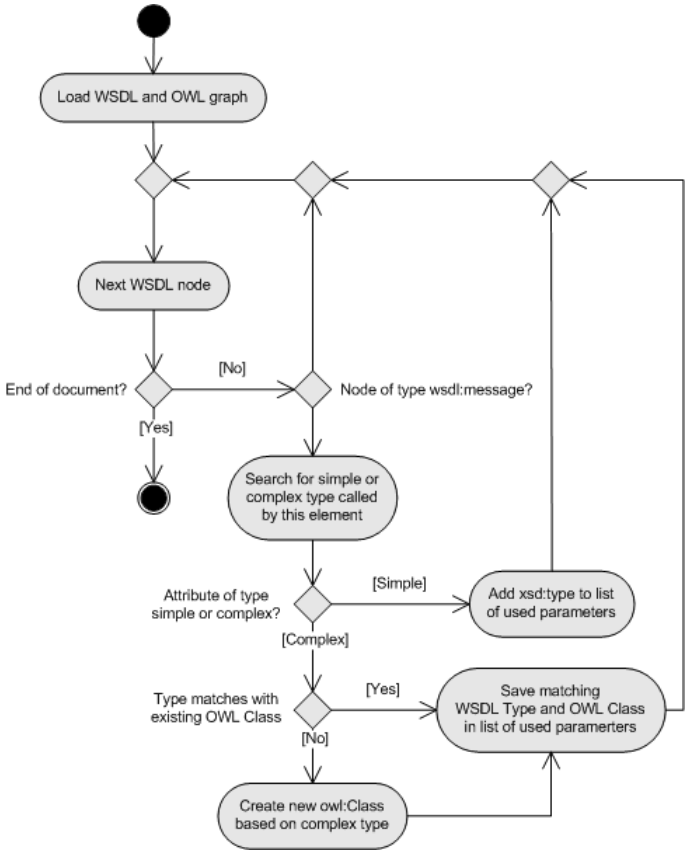


Figure 4.3: *The process flow of the OWL:Class and WSDL:Type matcher*

4.3.3 Process Matching

The third step of the algorithm matches an OWL-S simple process with a WSDL operation and creates an *atomic process* with a proper grounding as result. A WSDL operation is originally equivalent to an OWL-S atomic process. Because the available atomic processes origin from a category ontology the developer found in an archive or somewhere else, it is usually not possible to alter these atomic processes. Therefore, the algorithm matches the WSDL operation to the simple processes and creates a new atomic process. The matching is similar to the procedure in the previous type matching process. The process matcher compares the structure and syntactical naming of the WSDL operations with the OWL-S simple processes.

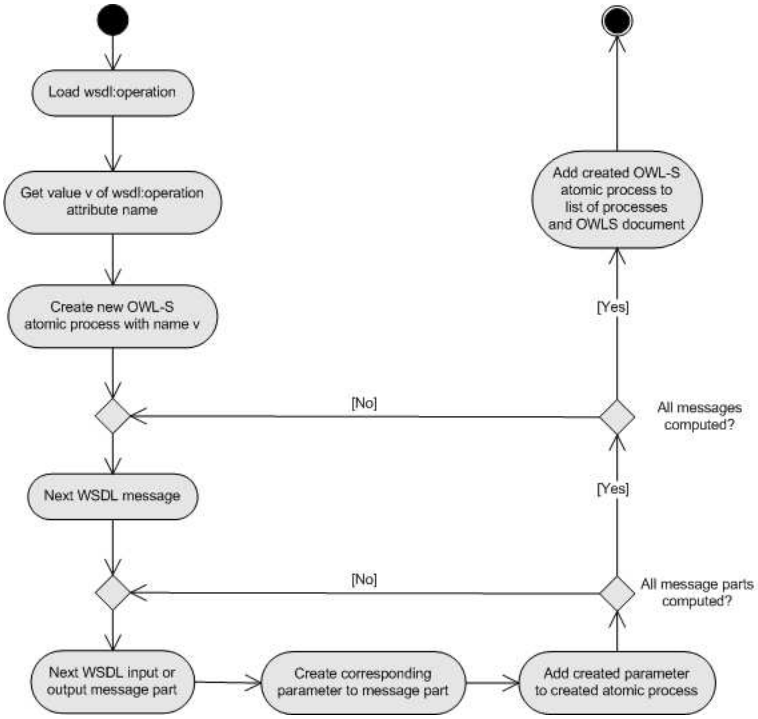


Figure 4.4: *The process flow of the atomic process creator*

The in- and output message parts are compared with the in- and outputs of potential simple processes. If an exact matching is found a new atomic process is created based on the matched simple process. For instance, the operation name will be set as identifier of the matched process. In the case of ambiguous matching results the help of the developer is required one more time. If no matching could be found the atomic process class with its parameters is created, based on the WSDL operation, but further contents have to be added manually. To complete the new category instance the algorithm takes the preconditions and effects given by the selected simple process and links them to the new atomic process.

In addition to the created OWL-S atomic process, the OWL-S *process grounding* is instantiated. Its function is to bind the process to the WSDL document. This procedure works automatically due to the reuse of the matched WSDL and OWL-S tags from the previous activities. In detail, the procedure goes on as follows. First, as basic set up for the grounding description the tag `grounding:Process` is linked to the relevant `process:AtomicProcess`. Subsequently, the URI of the WSDL document is linked to the instance of `grounding:wsdlDocument`. Furthermore, the WSDL operation is bound to the grounding. The `grounding:wsdlOperationRef` contains URIs to the WSDL operations and WSDL port types in `grounding:operation` and `grounding:portType`. The instance of `grounding:wsdlOperationRef` is then linked to the `grounding:wsdlOperation`. As fourth step the URI of the WSDL port is linked to the instance of `grounding:wsdlPort`. The same is done with the WSDL reference, WSDL service, and WSDL version linking the instances of `grounding:wsdlReference`, `grounding:wsdlService`, and `grounding:wsdlVersion`. Finally, the input grounding is

created due to the information available from the previous matching process. For each input the `grounding:wsdlInputMessageMap` is instantiated with the input parameter `grounding:owlsParameter` and the corresponding message `grounding:wsdlMessagePart`. The output parameters are bound equally. The `grounding:wsdlInputMessageMapList` enumerates all `grounding:wsdlInputMessageMaps`. Output message maps are treated respectively. This completes the automatic generated OWL-S process grounding for the atomic process.

A final remark in this process matching paragraph involves *composite processes*. So far all simple processes are considered if they are instantiated by newly created atomic processes. Composite processes consist of several simple and atomic processes and are put together by control structures. This allows constructing complex correlations of processes resulting in powerful and complex Web services. Instead of matching WSDL operations to simple processes also composite processes can be used as matching candidates for complex WSDL operations. The algorithm can pass recursively through the construction of the composite process. Existing control structures are split, split-join, sequence, iterate, repeat-until, repeat-while, any-order, choice, and if-then-else.

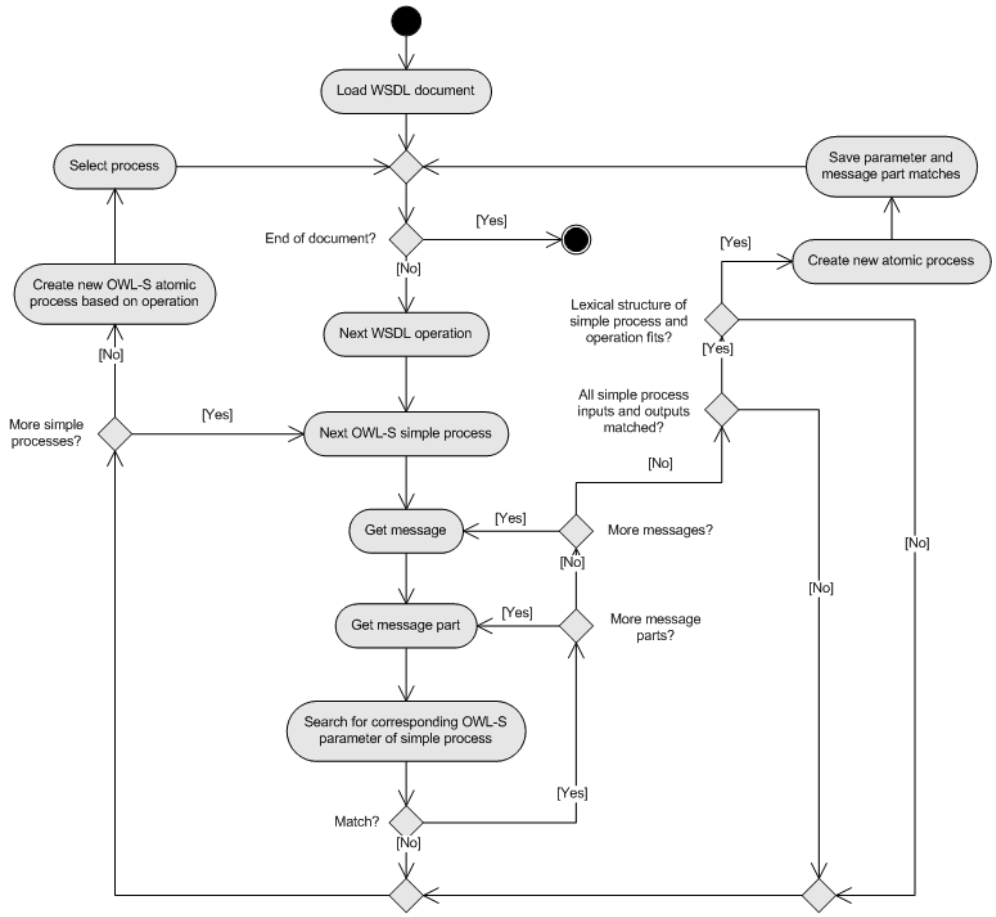


Figure 4.5: *The process flow of the simple process and WSDL operation matcher*

4.3.4 OWL-S Instance Creation

The final step in the WSDL-to-OWL-S algorithm creates the service and profile class of the OWL-S instance completing the OWL-S instance consequently. OWL-S acts as upper ontology in our approach and thus the OWL-S document needs some basic data besides the OWL classes, OWL-S processes, profile, and service descriptions. This data is independent to the performance and outputs of the algorithm. We provide a template describing this data such as defining the produced document as OWL-S document and standard imports and namespace definitions [18].

The *profile* class of the OWL-S instance can be easily created using information from the algorithm so far. The `profile:hasProcess` should be linked to the corresponding atomic process for instance. The same can be done with profile properties `profile:hasPrecondition` and `profile:hasResult` if existing, as well as the `profile:hasOutput` and `profile:hasInput` tags. As mentioned in section 4.2 we use a "default profile" description to fill out contact information etc. in the `profile:contactInformation`, `profile:serviceCategory`, and `profile:serviceClassification` tags. This data can be altered if necessary by the service developer afterwards. The tags `profile:textDescription`, `profile:serviceParameter`, and `profile:serviceProduct` should be instantiated finally. These tags provide data about the atomic process and can be therefore copied from the chosen simple process corresponding to the atomic process.

The *service* class can also be created easily. The service has to connect the process model, service grounding, and service profile. Therefore, a new service class needs to be instantiated and linked to the simple and atomic processes. The `service:supports` property links to the service grounding and the `service:presents` property links to the profile. In an additional step the atomic process, the grounding, and the profile have to be linked to the service in the opposite direction. The value `ServiceName` finally becomes the name or ID of the service.

Now, the OWL-S instance is *completed*. Most parts are created and filled out automatically. However, in most cases the Web service developer has to correct the output of this WSDL-to-OWL-S algorithm afterwards. Nevertheless, this algorithm is a great help in the annotation process of a semantic Web service.

4.4 Concept Appliance: Printing Service Annotation

This section describes a short walk-through of the suggested framework with an illustrative example. By doing this, the abstract framework becomes more intuitive and our conclusions for the remaining parts in this thesis more understandable. The example will be a *printing* service implemented at the University of Paderborn. The service is able to receive any postscript document and print it on any network printer within the university. It is realized as location-based Web service, which enables the service to integrate location information and location constraints from a service request into the service behavior and outgoing results. For instance, a requestor can demand in a request to *print the document to the nearest printer available*. In the query the requestors location will be specified and the nearest available printer can be determined.

4.4.1 Ontology Selection

Before the programming begins, we follow step one of the framework and select suitable ontologies in the domain of the new Web service. We can provide the framework with several ontologies, one is a *location ontology*, which describes geometric characteristics like “Coordinate”, “InBuilding” or “InRoom” and concepts like “Room”, “Floor”, “Corridor” etc. A *printer ontology* covers basic concepts and relations concerning printers at our university. Here, terms such as “PrinterName”, “InputFileType”, “MaxResolution”, “MemorySize”, “DuplexCapability”, etc. are described.

We select both domain ontologies, *location* and *printer* together with the upper ontology OWL-S. The OWL-S ontology specifies that the resulting Web service will have inputs, outputs, preconditions, and effects. It can capture the atomic, simple, and complex processes of the service as well as the service and process model.

4.4.2 Web Service Creation

The second step in the framework states, that now the Web service can be modeled. The developer is free to choose any preferred tools supporting this process. Due to the very small and simple service we want to illustrate here, we do not present any UML state charts etc. Instead, we give a short overview of the service’s interface and its basic process modeling.

The inputs and outputs of the service are easy to determine. The above service description requires that the service has a file and the requestor’s location as input. Only both information enable the service to realize a location-based printing service. The output will be the ID respectively the name of the printer selected by the service to print the document. Preconditions and effects are usually harder to determine. In this case, the preconditions are related to the inputs of the service. One precondition is, that the specified location has to be within a university building. Otherwise, the service will not be able to determine the nearest printer available to the requestor’s position. The other precondition is the type of the input file. All network printers are only capable of printing raw postscript documents directly on the printers. Any other file types have to be converted by software, which is not part of this service. The effect of the service has to describe, that the printer prints the received document. Figure 4.6 shows the Web service and its interface.

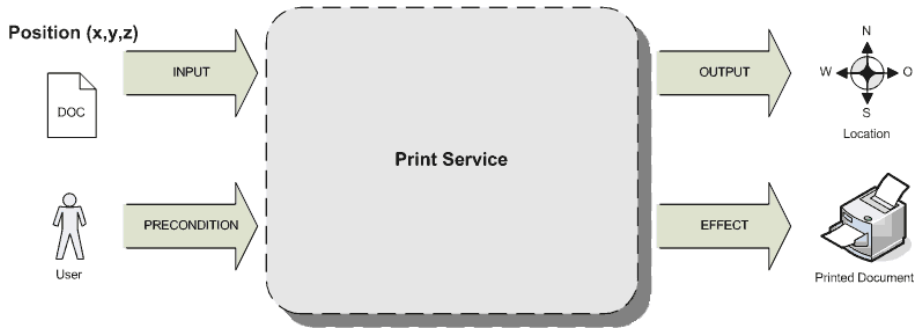


Figure 4.6: *In- and outputs of the location-based printing Web service*

Based on this description the Web service can be implemented. We decided to implement the functionality in Java in order to use the Java2WSDL tool [4] in the next step. Another reason is the usage of the OWL2Java tool [87] which is able to generate corresponding interfaces and classes for the Web service from the domain ontologies directly. The classes are conform to the Java Bean standard and are a good starting point for the remaining application logic to be implemented.

4.4.3 WSDL Creation

The Java2WSDL tool from the APACHE AXIS project creates a WSDL interface description file based on the Java classes implementing the location-based printing service. For every public Java method a WSDL operation is created. If the Java methods use simple Java variables as parameters, the communication messages in WSDL are realized by simple data types and the corresponding XSD types. Figure 4.7 shows as example a Java code snippet realizing a method returning the printer name to a corresponding printer ID `getPrinterName(String PrinterID)`. It has the printer ID of type string as parameter and the printer name as return value. The corresponding part in the WSDL document is shown in Figure 4.8.

A complex type does not occur in this snippet of the example, but is conducted easily with the `<complexType>` element. For instance, the Web service input requires the location of the requestor. This is specified with a relative coordinate within the university. The complex type `coordinate` consists of a `x`, `y`, and `z` component, each of simple type `xsd:float`. They are arranged as sequence with the `<sequence>` element in WSDL.

```
1  String PrinterID = "f2-425-1w";
2  String PrinterName = "Worf";
3  ...
4  /**
5   * gets the printer name by ID
6   * @param   PrinterID    ID of the printer
7   * @return  PrinterName  name of the printer
8  */
9  public String getPrinterName (String PrinterID)
10 {
11   if (PrinterID == this.PrinterID)
12     return this.PrinterName;
13 }
```

Figure 4.7: *`getPrinterName` method in the location-based printing Web service*

4.4.4 OWL-S Creation

At this point in the framework, the WSDL-to-OWL-S algorithm is activated. It creates a new OWL-S layered ontology document based on the WSDL description, the OWL-S category, and OWL domain ontologies. After parsing all relevant documents, the algorithm performs the type matching. If primitive types like `xsd:integer` or like in the `getPrinterName` method

```
<?xml version="1.0"?>
  <definitions
    <!-- Namespace definitions -->
    targetNamespace="http://upb.de/MyNamespace"
    xmlns:tns="http://upb.de/MyNamespace"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    ...
    <!-- Messages definitions -->
    <wsdl:message name="getPrinterName_Input">
      <wsdl:part name="PrinterID" type="&xsd;string"/>
    </wsdl:message>
    <wsdl:message name="getPrinterName_Output">
      <wsdl:part name="PrinterName" type="&xsd;string"/>
    </wsdl:message>
    ...
    <!-- PortType definitions -->
    <wsdl:portType name="getPrinterName_PortType">
      <wsdl:operation name="getPrinterName_Operation">
        <wsdl:input message="tns:getPrinterName_Input"/>
        <wsdl:output message="tns:getPrinterName_Output"/>
      </wsdl:operation>
    </wsdl:portType>
    ...
    <!-- Binding definitions -->
    <wsdl:binding name="getPrinterName_SoapBinding" type="tns:getPrinterName_
        PortType">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
          http"/>
      <wsdl:operation name="getPrinterName_Operation">
        <soap:operation soapAction="tns:setPrinterName"/>
        <wsdl:input>
          <soap:body parts="PrinterID"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body parts="PrinterName"/>
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    ...
  </definitions>
```

Figure 4.8: *Corresponding WSDL descriptions to the `getPrinterName` Java methods*

`xsd:string` types are used, they are not converted and used directly in the OWL-S document. Sequentially, WSDL complex types are compared with OWL classes. If a `wsdl:complexType` has the same structure and naming as an `owl:Class` both are linked together. In the location-based printing example, the complex type `Coordinate` has the same structure and naming as the `owl:Class rdf:ID="Coordinate"` specified in the location ontology. Both contain three

sequenced `xsd:float` elements named `x`, `y`, and `z`.

The second part of the algorithm executes the process matching, matching WSDL operations with OWL-S simple processes. Here, the developer can reuse OWL-S descriptions from other implementations if matching correctly, or new OWL-S template descriptions are generated. In this case, the templates have to be completed manually by the developer. We

```
<process:SimpleProcess rdf:ID="getPrinterName">
  <process:hasInput>
    <process:Input rdf:ID="PrinterID">
      <process:parameterType rdf:resource="&xsd;string"/>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="PrinterName">
      <process:parameterType rdf:resource="&xsd;string"/>
    </process:Output>
  </process:hasOutput>
</process:SimpleProcess>
```

Figure 4.9: *Simple process `getPrinterName` in the OWL-S description*

need an atomic process mapping each WSDL operation from available OWL-S descriptions, but often the atomic processes in a category ontology cannot be altered. Simple processes describe an abstract view of an atomic process hiding certain details that are irrelevant in this view or confidential. The structure and naming of each WSDL operation is compared to each available simple process in the selected OWL-S descriptions. Here, the WSDL operation `getPrinterName_Operation` matches the OWL-S simple process `getPrinterName` from an already existing OWL-S service description, see figure 4.9. The in- and output message parts from the WSDL document (`PrinterID` as `xsd:string` and `PrinterName` as `xsd:string`) are compared with the in- and outputs of the simple Process. Here, they match exactly. If this is not the case, a template will be generated containing the correct numbers of inputs and outputs, the process names and parameter data types. The developer can then easily fill out the remaining information.

After successfully matching a simple process to the WSDL operation, a new corresponding atomic process can be generated. Figure 4.10 shows the atomic process `getPrinterName`.

The WSDL to OWL algorithm continues with the instantiation of the OWL-S grounding binding the new process to the WSDL document. The `grounding:process` references to the `process:AtomicProcess` and `grounding:wsdlDocument` is referencing to the WSDL document. Further, `grounding:wsdlOperationRef` point to the WSDL operations and `grounding:portType` to the WSDL port type definitions. Message maps are generated for each WSDL message part and thus each OWL-S parameter. The grounding is usually a simple but very long document mainly with references connecting the WSDL elements with the OWL-S elements and therefore not shown here.

The remaining steps in the OWL-S instance creation are the generation of the service and profile class. Here, the default preamble template (partly shown in figure 4.11) is used contain-

```
<process:AtomicProcess rdf:ID="getPrinterName">
  <process:hasInput>
    <process:Input rdf:ID="PrinterID">
      <process:parameterType rdf:resource="&xsd;string"/>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="PrinterName">
      <process:parameterType rdf:resource="&xsd;string"/>
    </process:Output>
  </process:hasOutput>
</process:AtomicProcess>
```

Figure 4.10: *Generated atomic process* `getPrinterName`

ing standard imports, namespace definitions, etc. including references to the used ontologies. The profile and service information are added using standard information easily available to the service developer.

```
<?xml version="1.0"?>
  <rdf:RDF
    <!-- Namespace definitions -->
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:grd="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    ...
    <!-- Service definitions -->
    <service:Service rdf:ID="PrintingService">
      <service:presents rdf:resource="#PrintingService_Profile"/>
      <service:describedBy rdf:resource="#PrintingService_ProcessModel"/>
      <service:supports rdf:resource="#PrintingService_Grounding"/>
      ...
    </service:Service>
    <!-- Profile definitions -->
    <profileHierarchy:PrintingService rdf:ID="PrintingService_Profile">
      <profile:has_process rdf:resource="PrintingService_ProcessModel"/>
      <profile:serviceName>Printing Service</profile:serviceName>
      <profile:contactInformation rdf:resource="#PrintingService_Contacts"/>
      <profile:hasInput rdf:resource="#PrinterID"/>
      <profile:hasOutput rdf:resource="#PrinterName"/>
      ...
    </profileHierarchy>
    ...
```

Figure 4.11: *Preamble template, service and profile class for the OWL-S instance*

Now, the OWL-S document for the example printing service is complete. Framework steps

six and seven are the accommodation of the results and the Web service testing and deployment.

4.5 Summary and Results

The goal of this chapter is to get a thoroughly annotated semantic Web service. We assumed to support the developer at the service development directly is the best way. The proposed framework helps the developer to create mostly automatically the needed semantic descriptions and documents for Web service in development. Thus, the manual creation of complex and error-prone descriptions is not necessary any more.

This framework allows gathering semantic meta-data directly from the Web service developer and is no longer encoded implicitly in the syntactical descriptions. A lot of semantic information can be derived from the WSDL descriptions together with the assigned domain and category ontologies. The new WSDL-to-OWL-S algorithm is able to find a corresponding simple process in the category ontology matching the WSDL operations. Atomic processed based on the chosen simple process are created and bound to the operations. A well described OWL-S instance is created afterwards to complete the Web service's semantics. In [18] prototype tests showed a fine working environment if the amount of available simple processes is limited. Too many very similar processes increase false assumptions and a lot of user interaction. Therefore, the benefits of an automatic approach are lapsed. But, in a Web service developer environment, as assumed, the framework gives considerably support.

5 Semantic Service Composition

> Insanity: doing the same thing over and over again and expecting different results.
>
> Albert Einstein

The prior chapter described the automatic annotation process of a Web service. Standard Web services become semantically described Web services by using shared terms out of an ontology and respecting structures and descriptions due to the upper ontology OWL-S. Because the manual creation of a semantic service description is a time-consuming and error-prone work, we presented a WSDL-to-OWL-S algorithm supporting Web service developers. Unfortunately, a semantic service matching is in many cases not enough. In order to come to a fully satisfying service matching the possibility of service composition is discussed in this section. The second step in our approach is therefore to check the composeability of basic semantic Web services to fulfill complex user requests. The semantic annotation allows thereby a much more advanced and precise composition than in traditional composition approaches such as BPEL4WS[1] [2] or WSCI[2] [7]. MCILRAITH ET AL. introduced a semantic-based Web service composition [103], but our approach is not as restricted as their approach depending on explicit business goal definitions [145]. We introduced some basics of our approach, a semantic Web service matchmaker, in [64]. It takes the semantically described inputs, outputs, preconditions, and effects into account and realizes through a clever matching result reduction the composition of Web services. The following sections present the problem definition, requirements, and workflow of the matchmaker.

5.1 Problem Definition

Our approach of an OWL-S matchmaker provides different basic matching functions such as ontology matching, service grounding matching, service profile matching, and service process matching for a composition of semantic Web services using WSDL, OWL-S, and SWRL[3].

[1] Business Process Execution Language for Web Services
[2] Web Service Choreography Interface
[3] Semantic Web Rule Language

This concept is embedded in a framework considering several steps resulting in a service composition. In a common scenario a user requests a service accomplishing a specific task. With the help of semantic annotated services the meaning of request and available services can be understand to match each other. Unfortunately, one service does usually not satisfy a request directly. Therefore, a composition of multiple basic services is often required to meet the request. The standard example is such a case is the request for a *travel booking* service combining *hotel*, *flight*, and *car booking* services. Many problems arise in this scenario ranging from transaction safety to service security aspects which are not goal of this thesis. We concentrate on matching the outputs, preconditions, effects of one service to the inputs and preconditions of a second service. This can be repeated several times and the resulting combination will then hopefully match the user's request. The challenge lies in the fact, that services from different domains with different ontologies are combined here. Thus, the matchmaker has to meet some fundamental requirements described in the next paragraph.

In the domain of *printing services*, which was multiple mentioned so far, the following example is possible. A user is looking for a service printing ".doc" files to the nearest printer available. The semantic description of a location-based printing service helps to identify the exact outputs and effects matching the request. Unfortunately, the service is only capable to handle postscript files as input to the service. Only in combination with another service transferring the ".doc" file into a postscript file satisfyingly matches the user's request. The OWL-S matchmaker is capable of finding these suitable combinations.

5.2 Matchmaker Requirements

To conduct a service matching towards a service composition the matchmaker has to perform several steps. Each step fulfils a certain task in the corresponding framework and has specific requirements.

First, all elements affecting the matching which originate from ontologies, syntax, and semantic descriptions need to be identified. Thus, all information from the user request, participating ontologies, and service information need to be available. In a *matching-cut definition* (see section 5.3) all these elements are collected and a reduction to the relevant elements (for the matching) is performed. This reduces the matching complexity and makes a semantically service matching possible.

After gathering all information, the each two different domain ontologies need to be compared with an *ontology matching*. The meaning of connecting elements between two services needs to be the same. Thus, parts of both ontologies assigned to the services have to be aligned. In a *semantic matching* component, the framework needs to compare the service profiles, service models, and service groundings. Relevant information describes essential parts of each service turning the balance of a successful of unsuccessful service matching. Besides all semantic information, a *syntactical matching* considering the in- and outputs have of course to be checked. Like before, only the relevant parts in the WSDL syntactical description need to be considered in the service matching process.

Having worked with ontologies, syntax, and service descriptions separately, the overall correspondence between services has to be determined. A weighting of each aspect has to be

performed carefully to calculate the *service matching degree*. If the service matching degree is sufficient a *service composition* can be prepared. The service matching results in precise connections of inputs and outputs of all participating services.

5.3 Matching Workflow

The matchmaking process realizes different matching parts considering ontologies, syntax, and semantic service descriptions. Languages providing this information are WSDL, OWL, and SWRL. Preconditions and effects of services for example, can be modeled precisely with SWRL in OWL statements [144]. Starting with ontology comparisons using the FOAM framework from EHRING and SURE [44], the OWL-S API [143] for Java is used again like in the annotation process in chapter 4 before. Thus, all relevant information is available in efficient data structures on the implementation level. Further semantic information like the service profile, service process descriptions, and service grounding details are used to realize a substantiated service matching. The service composition after a successful matching is realized with the "automatic composition and invocation" approach of SHESHAGIRI ET AL. [142].

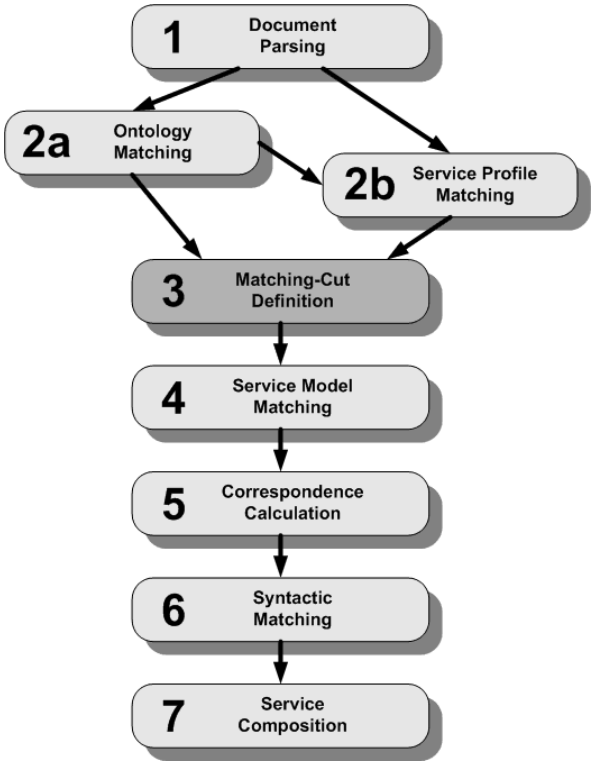


Figure 5.1: *Overview of the OWL-S matchmaker composition process*

The matching procedure consists of eight steps illustrated in figure 5.1. The previously mentioned requirements note several comparisons of different service description parts in no specific order. As we look closer to the matching procedure a naturally order appears. First, of

course the necessary information has to be loaded and parsed. After that the service profile matching and the ontology matching can be done parallel, due to their independency. They result in the matching-cut which has to be evaluated afterwards. Surprisingly, the next step considers further semantic activity. In step four and five of the matching procedure the service model matching and the overall semantic correspondence is determined. Only now the syntactic information originating from the WSDL description is considered before in the last step the service composition is done. This goes along with the following idea. The meaning of for example a Web service input parameter can be described in many different ways and of course in many different syntax fashions. The syntactical descriptions may differ significantly causing a clear failure of the matching, but the semantic descriptions are much closer related due to the ontological connections. Therefore, the semantically comparison is executed before and syntactical comparison. The following paragraphs discuss each step more precisely.

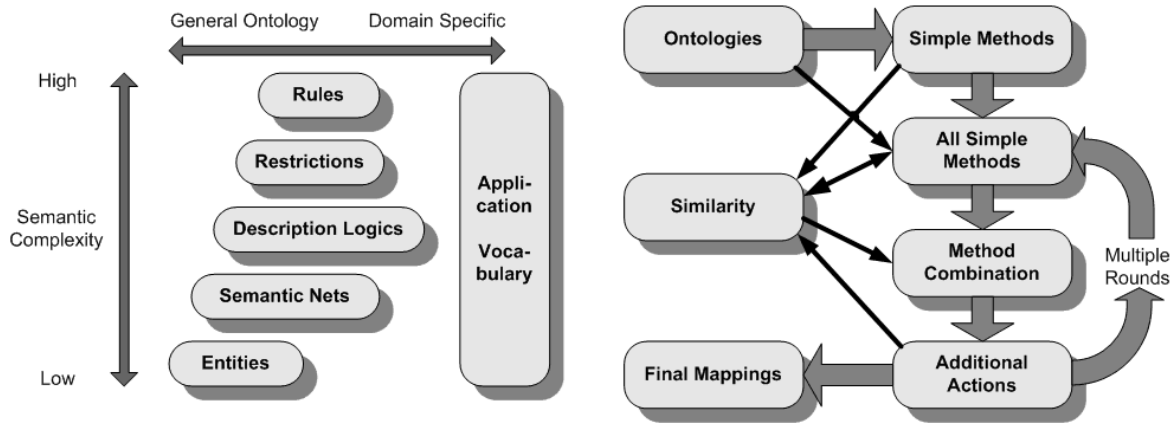
5.3.1 Document Parsing

The matching procedure requires some necessary information from several documents usually provided with an adequate annotated semantic Web service. The relevant ontologies are parsed to have the shared understanding of each service available which is defined in the ontology's terms, concepts, and interrelationships. The OWL-S API [143] ensures a storage in efficient data structures to save the document's structure and naming issues. Also the service descriptions via OWL-S files are parsed to get the service's profile, classification, maintainer information, in- and output descriptions, and grounding details. The syntax information from the according WSDL description is also parsed and loaded like in the annotation process in section 4.3.1.

5.3.2 Semantic Matching: Ontology Matching

Steps two through four involve the semantic matching parts of the matching procedure. Firstly in 2a) the participating ontologies are matched. The ontology comparison uses the FOAM procedure from EHRIG and SURE [44]. Within this procedure corresponding parts out of different ontologies are searched using different methods to evaluate the similarity. The goal is to find for each concept in an ontology A a corresponding concept in ontology B with the same or very similar semantic. The methods are building a stack (see figure 5.2a)) starting with two methods working on entity level, two for semantic networks, ten for description logics, one for restrictions and two for the application vocabulary. All seventeen methods are used and contribute their results, which are weighted and aggregated to an overall result. The aggregation also controls the run of *multiple rounds* (see figure 5.2b)) of each method using preliminary results again as input to improve the matching performance. The usage of each single method, its weights and thresholds in the aggregation, and the number of rounds are parameterized and can be adjusted from outside. The configuration of the threshold is of major importance for the quality of the result. Raising the threshold increases the matching precision, but decreases the amount of hits for corresponding ontology concepts. Hence, FOAM is a flexible, adaptive, and automatic way to compare ontologies and determine their common concepts ideal for the needs in this thesis. We evaluated other approaches and the

exact parameterization of the FOAM procedure. We received best results with the strategy "DecisionTree", which defines the weights of the given methods. The number of iterations is set to 19, so the process runs 20 comparisons. A threshold of 90 percent produces comprehensible results in an afterwards manual consultation.



(a) FOAM similarity stack (b) FOAM matching process

Figure 5.2: *Similarity stack and matching process of* FOAM *[44]*

5.3.3 Semantic Matching: Service Profile Matching

In 2b) the service profiles are matched. Figure 5.3 shows the process flow of the service profile matching. Profiles are originally intended to advertise a service and include non-functional parameters like `profile:serviceName`, `profile:serviceParameter`, and `profile:contactInformation`. Functional parameters are included as well in the profile describing the in- and outputs (`profile:hasInput`, `profile:hasOutput`) and the preconditions (`profile:hasPrecondition`). The effects are not considered, because they only describe the changes to the outside world and are included in the outputs implicitly [142]. Their comprehension does neither improve the comparison nor the additional non-functional parameters like security or quality-of-service. They need to be evaluated in the service discovery long before the actual service matching [19]. Requestors usually also express their queries by defining a service profile they expect. Therefore, it is likely all major characteristics of a service are defined in the service profile. The profile matching in this work is integrated in a more complex matching process to achieve a higher matching quality. The profile matching (step 2b)) can be done partly in parallel to the ontology matching (step 2a)). The non-functional parameters concerned in the profile matching are independent to the used ontologies, but the functional parameters are using concepts of the participated ontologies already.

The similarity of each non-functional profile component is determined by their lexicographical property. This approach is like in other keyword-based comparisons, because the non-functional profile description is independent to the assigned ontologies, and thus we do not have a shared understanding or meaning useful for the comparison. We evaluated that if

more than 75 percent of the letters of the corresponding elements are identical, the parameters can be evaluated as match. If the agreement is between 50 and 75 percent, a correspondence is possible and with less than 50 percent a correspondence is highly improbable. Summarizing each parameter similarity can therefore lead to a matching decision of the profile element, although it is not very precise.

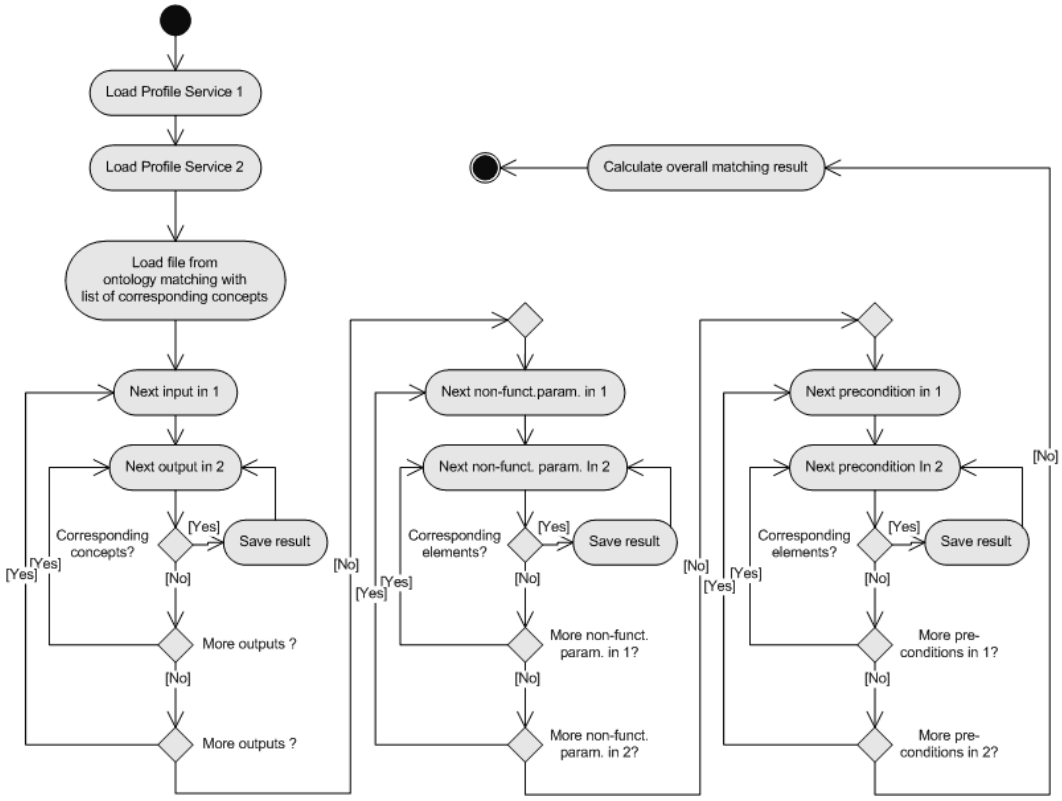


Figure 5.3: *The process flow of the service profile matching*

For the correspondence of the functional components in the service profile of the Web service the results from the ontology matching are used. The FOAM result file contains all corresponding elements of both ontologies. The in- and outputs in the service profile description are constructed with these elements. Aim of this matching is the composition of two Web services. Thus, one or more outputs of one service have to match one or more inputs of another service. Only in this case a successful sequence of both services can be achieved. So, if for instance an output of one service profile uses exactly the same elements as an input in the other service profile – corresponding to the FOAM result file – these functional parameters match perfectly and the services are composeable.

The FOAM result file also delivers a similarity degree of two corresponding concepts. Consequently it is possible that one output matches exactly one input, but the use concept is only to a certain percentage corresponding to another concept. In this case a user-defined threshold can be used to determine flexibility of the matching process in this aspect. We sug-

gest a threshold of 75 percent. Any similarity degree below this value increases the chance of failure in the whole matching process. If more than one functional parameter is available on either Web service several combinations become possible. The service profile matcher simply iterates all inputs and outputs matching every possible constellation. This is a time-consuming task, but necessary to recognize all composition configurations. For each input-output-combination the pair with the highest similarity is chosen. To determine an aggregated similarity degree for the composition of two Web services the lowest similarity of all functional parameter pairs is identified. A calculated average similarity would not reflect the natural impression. For example, if a pair of Web services is composed with two pairs of functional parameters, the weaker connection is dominant for the whole connectivity. Therefore, we forward only the weaker result to the further steps in the matching process.

The existence of one successful combination of a pair of functional parameters is needed to proceed with the matching. Therefore, we define its non-existence or its existence with a similarity degree below the threshold of 75 percent as an *stop criterion*. If more than one successful combination defines the connection between two Web services the weakest similarity degree defines the overall similarity degree.

The similarity of two service profiles is the common basis for many semantic matching approaches. Unfortunately, considering possible Web service compositions increases the amount of semantic information and the amount of matching candidates dramatically. Only a reduction to the relevant core of needed information for a semantic matching makes this approach possible.

5.3.4 Semantic Matching: Matching-Cut Definition

At this step of the matching procedure we combine the results of the previous ontology matching and the non-functional and functional parameter matching of the service profiles to the definition of a so-called *matching-cut*. This procedure is new, because other approaches like [114, 152] only consider the service profile alone to match service advertisements with user requests. Any composition of services if normally far to complex to be solved in sufficient time. By combining the results of the ontology matching and the profile matching it is possible to reduce the available semantic information of two services to the relevant information for a successful composition. The corresponding elements in both participating ontologies, as well as the non-functional and functional parameter matchings from the service profile constitute the basis for the matching-cut. The comparison of two ontologies alone does not state any similarity of two Web services. Likewise, the similarity of non-functional parameters in two service profiles of two Web services does not state any similarity. Only the functional parameters described with highly similar elements from corresponding ontologies assigned to each Web service and a certain similarity in the non-functional service profiles is significant. This significant similarity decides about the Web service's composeability. Furthermore, after the matching-cut aggregation we consider the service model describing the internal Web services structure and the processes used in the services to support this decision extensively.

The matching-cut defines the "working area" for the matching. The goal of this procedure is to reduce the amount of all relevant information to a manageable amount of only important information considering a composition of two Web services. It allows composing Web

services with a small syntactical, but with a high semantically compatibility. This information reduction is already done with the preliminary steps 2a) and 2b). Now we aggregate and weight this information. The following information is available in the matching-cut:

- corresponding ontology elements (intersection of ontologies)
- mapping of corresponding ontology elements and the functional parameters
- similarity degree of the functional parameters
- similarity degree of the non-functional parameters

The first fraction in the matching-cut contains the matching elements of the ontologies which are used by the Web services. Figure 5.4 illustrates this. The corresponding ontology concepts are used to gain a better appreciation and interpretation of the applied terms. The second fraction consists of the matching parts in the non-functional parameters like the contact information, which is not as important as the first fraction. The third part of the matching-cut imports the matching of the functional parameters with its mappings to the shared ontology concepts. This is the most important part of the information aggregation. If has a major influence on the composeability and thus the matching result. For instance, only if a service output matches another service's input regarding its parameters and semantic interpretations a composition is successful. This information lies in the mapping between the functional parameters and its ontological roots.

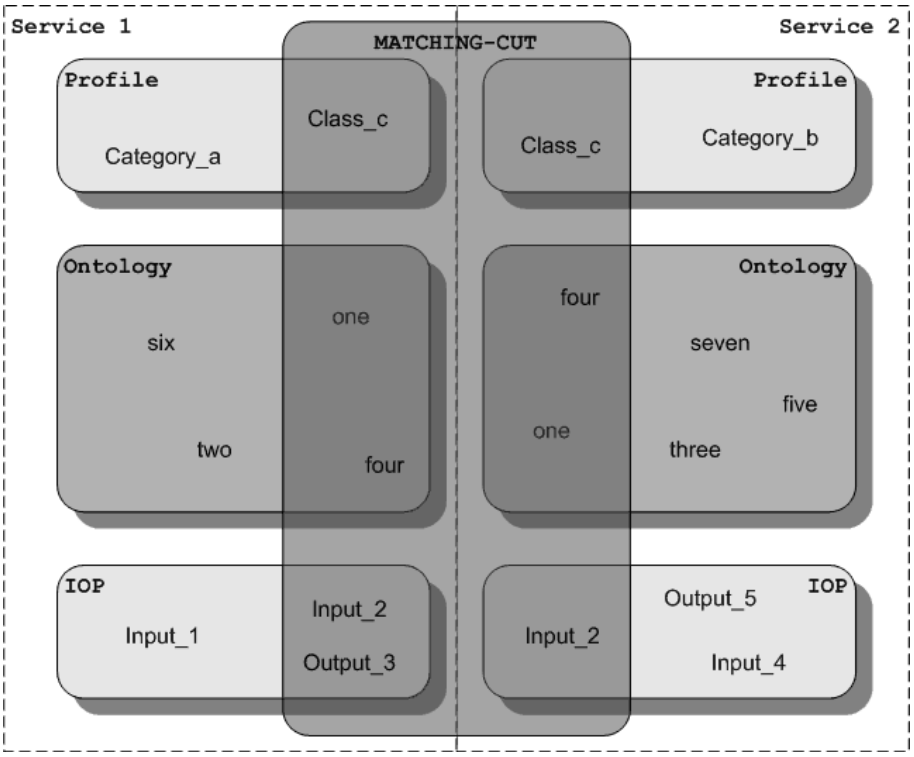


Figure 5.4: *Matching-cut illustration*

The non-functional matching result comes with a matching degree of full, maybe, or none correspondence. This is taken into account with 25 percent total participation. The functional parameter matching contributes 75 percent to the result. We say a *fully correspondence* is achieved with at least one successful match between a pair of in- and outputs across two Web services where the similarity degree of the partitioned concepts is greater than 90%. *Maybe correspondence* is awarded if at least one successful matching pair exists where the similarity degree is greater than 75%. *No correspondence* is reached if no pair of input to output or output to input of two services can be found. This result can be taken to decide for a premature end of the matching procedure. Considering information in the service model description can lead to further improvements in the next step. After that the overall matching degree is determined and a final decision towards composeability is made.

5.3.5 Semantic Matching: Service Model Matching

The service model describes the internal logic of the Web service defining the atomic, simple, and composite processes constructing the service process behavior with control constructs. Each process has at least in- and outputs and applicable also preconditions and effects. The internal process structure is not interesting for a service matching. They only result in a detailed functional and transformation description, which does not improve the quality of a service matching with hardly composeable services. However, the service model can contribute to the matching. The parameter descriptions on model level can hold more information as the parameter descriptions on the profile level such as quality demands on certain parameters. The profile level is more like an interface description of the service where as the model level is a module description of the service.

In this work only the atomic processes are regarded, because they are the basis for the simple and composite processes. The control structures do not influence the matching results at all and thus do not improve the matching quality. Best-practice to describe the parameter conditions arising are SWRL preconditions [144]. The procedure to use SWRL will be important part of the next OWL-S specification version 1.2. The restriction to atomic processes is further limited in combination with the usage of elements only from the matching-cut. This means only atomic processes with more than one parameter containing shared descriptions is used for the matching. Thus, only atomic processes with a significant contribution to the service's interfaces are considered.

The so far gathered information from the functional parameters of the service profile are confirmed and verified by the service model matching. The service model is evaluated for both semantic Web services. The IOPEs (inputs, outputs, preconditions, and effects) of the service profile are compared with the IOPEs of the relevant atomic processes to identify differences. *Fully correspondence* between both Web service's service models is given if no difference can be determined or the model describes more than all functional parameters within the service profile. A *maybe correspondence* emerges if several elements differ between service profile and service model, but still all parameters in the service profile are described in the service model. Are not all functional parameters of the service profile covered in the service model an inconsistence is highly possible. Therefore, *no correspondence* is given and the Web service composition is aborted.

5.3.6 Semantic Matching: Overall Correspondence Calculation

This is the last part in the sequence of semantic matching issues calculating the overall semantic Web service's correspondence. We consider the results of the matching-cut which includes the ontology correspondence and the functional and non-functional service profile correspondence. Furthermore, we take the results of the service model matching into account. All three procedures were described in the prior paragraphs and are now aggregated with certain weights. The matching-cut represents the most important part and is weighted with 60 percent. Service profile matching and service model matching are each weighted with 20 percent and all three correspondence degrees are grouped in a tuple of (full, maybe, none) correspondence. The first component represents the sum of all three full correspondence percentage values, the second the sum of maybe correspondences, and the third the sum of all none correspondence percentage values. This tuple is then interpreted as result after four steps of the matching procedure as follows.

A Web service *matching* between two semantically annotated Web services is *possible* if the first component of the tuple (full correspondence) is greater than 75 percent, or the first component is greater than 50 percent together with at least 25 percent in the second component (maybe correspondence). This constellation stands for a major influence of the matching-cut, due to its 60 percent proportion. The matching-cut evaluation is most important and cannot be compensated by other minor matching results. An automatic service matching without further user interactions can be done on this basis, unless the service model matching results with maybe. In this case the user has to identify implicit functional parameters for the service composition.

The *matching* between two semantically annotated Web services *maybe possible* if the first component of the tuple is below 50 percent, but the sum of the first and the second value is greater than 50 percent. This constellation indicates a possible service correspondence, but not enough information for an automatically composition. The user has to decide about the continuation of the matching process in this case.

At last, the *matching* is *not possible* in all other cases. Here, the matching-cut indicates insufficient corresponding elements and thus no basis for a suitable service composition is given. The matching process is terminated here.

5.3.7 Syntactic Matching

The basic service matching procedure is completed with the overall semantic correspondence calculation in step five. The syntactic information is usually described in the language WSDL which is part of a semantic Web service annotation. We considered this information implicitly due to the service model treatment, where OWL-S atomic processes are examined. They are based on WSDL operations.

In this paragraph we take the WSDL description as preparation of a Web service composition. The results from step four are taken to check the signatures of the functional parameters. Figure 5.5 shows the WSDL matching. In the case of mismatching data types often a simple transformation can deliver a suitable output of a Web service to an input of another Web service.

Additionally, the emerging composed service can derive is own annotation from the internal services. The input parameters of the composed semantic Web service are the sum of all inputs of all internal Web services without the inputs directly served by outputs of other Web services. The output parameters of the composed semantic Web service is analogously the sum of all outputs of all internal Web services without the ones delivering inputs to other services directly. The preconditions of the composed service are the sum of all preconditions.

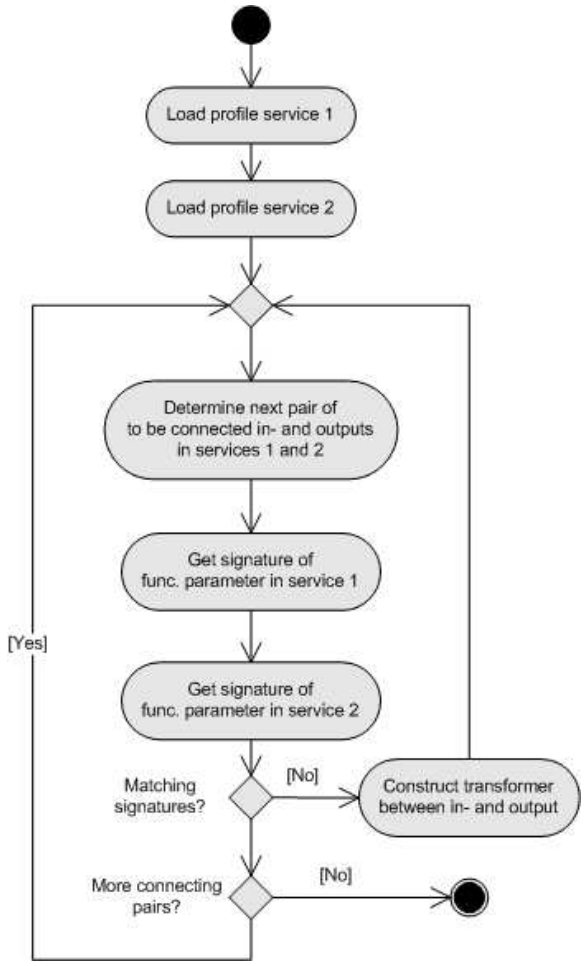


Figure 5.5: *The process flow of the WSDL matching*

5.3.8 Service Composition

The final step of the matching procedure is done with the announcement if two semantic Web services are composeable. In an additional step this composition now has to be realized. Therefore, different workarounds like the one of SHESHAGIRI [142] can be used. The results of the matching procedure are extensive and the composition should be realized without any troubles. Figure 5.6 illustrates an example of a composed Web service containing semantic Web services.

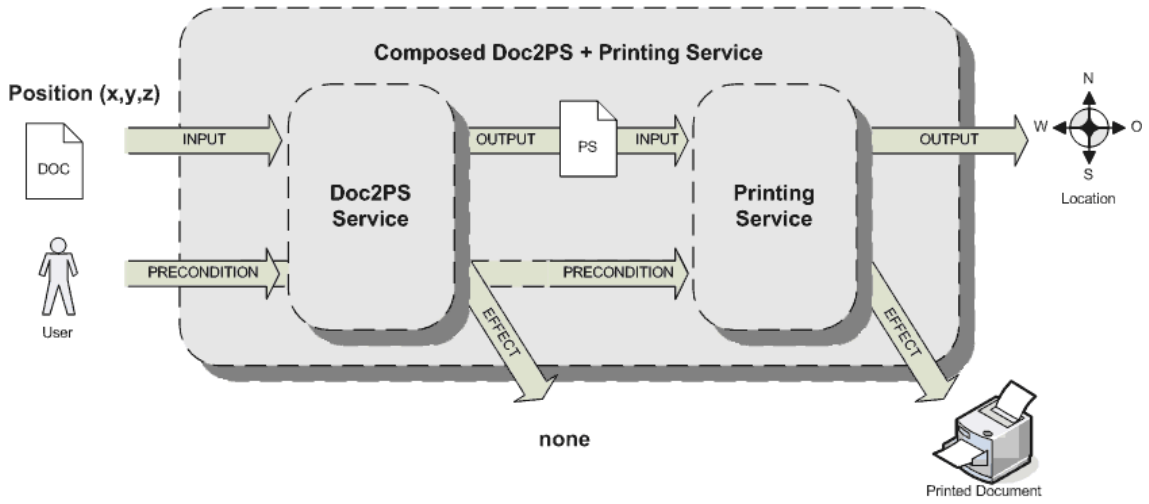


Figure 5.6: *Example of a composed semantic Web service*

5.4 Concept Appliance: Printing Service Composition

An example service composition walk-through is presented in this section to illustrate the OWL-S matchmaking procedure proposed in this chapter. Two services exist in this scenario. The *doc2ps* transforms any ".doc" file into a postscript file and the *location-based printing* service previously described in 4.4 is capable of printing any postscript document on network printers within the university. Only a composition of both services is able to match a request like *print a ".doc" file on the nearest printer available*. Both services are semantically annotated, but originate from different domains. The *doc2ps* service has an OWL-S description and is connected to a *document* ontology describing different types of documents and concepts useful in this domain. The *printing* service has also a OWL-S description and uses concepts from a *printing* and a *location* ontology.

5.4.1 Ontology and Service Profile Matching

In the first step of our proposed OWL-S matchmaking procedure all relevant documents are parsed. Here, the *location*, *printing*, and *document* ontologies are loaded, as well as the OWL-S documents for the *doc2ps* and the *printing* services. The ontologies provide background information about the used terms, concepts, and interrelationships between the concepts and the OWL-S files provide information about the service profile, maintainer, in- and outputs, and grounding details.

By performing step 2a) the participating ontologies are matched with FOAM tool regarding special settings. Selecting the "DecisionTree" strategy, 19 iterations, and a 90 percent threshold delivers the corresponding concepts of the ontologies. Table 5.1 shows an excerpt of the FOAM result file.

Concept Ontology 1	Concept Ontology 2	Matching Degree
http://localhost/services/ document.owl#Person	http://localhost/services/ location.owl#Person	1.0
http://localhost/services/ document.owl#GMT_UTC-1	http://localhost/services/ printing.owl#GMT_UTC-1	1.0
http://localhost/services/ document.owl#PrinterName	http://localhost/services/ location.owl#Printer	0.8
http://localhost/services/ document.owl#Creator	http://localhost/services/ printing.owl#Submitter	0.3
http://localhost/services/ document.owl#PSFile	http://localhost/services/ printing.owl#PostscriptFile	1.0
http://localhost/services/ document.owl#Inputdocument	http://localhost/services/ printing.owl#Inputfile	0.9
http://localhost/services/ document.owl#PrinterName	http://localhost/services/ printing.owl#PrinterName	1.0

Table 5.1: *Ontology matching excerpt of* FOAM *result file*

In step 2b) the service profiles of the *doc2ps* and *printing* services are matched. The functional parameters (inputs, outputs, and preconditions) and the non-functional parameters (service category, service classification, and maintainer information) are compared. Figure 5.7 shows the service profile of the *doc2ps* service. Unfortunately, classification, category, and maintainer information are completely different for the *doc2ps* and the *printing* service. The keyword-based comparisons of e.g. "doc2ps converter" and "location-based printing service" in `profile:serviceName` are zero. This usually occurs when comparing services from different domains.

```
<profile:Profile rdf:ID="Doc2PS_Profile">
    <service:isPresentedBy rdf:resource="#Doc2PS_Service"/>
    <profile:serviceName xml:lang="en">doc2ps converter</profile:serviceName>
    <profile:textDescription xml:lang="en">This service converts a .doc file
        into a .ps file.</profile:textDescription>
    <profile:contactInformation>
      <actor:email>urerrer@upb.de</actor:email>
      ...
    </profile:contactInformation>
    <profile:serviceClassification rdf:resource="#Document"/>
    <profile:serviceCategory rdf:resource="#Documentconverter"/>
    <profile:hasPrecondition rdf:resource="#FileExists"/>
    <profile:hasInput rdf:resource="#DocFile"/>
    <profile:hasOutput rdf:resource="#PSFile"/>
    <profile:hasEffect rdf:resource="#FileConvertedEffect"/>
</profile:Profile>
```

Figure 5.7: *OWL-S profile of* doc2ps *service*

In the case of functional properties there is a match. “PSFile” is an output specified by `profile:hasOutput` in the *doc2ps* service profile and “PostscriptFile” an input specified by `profile:hasInput` in the *printing* service. Both are identical due to the corresponding concepts in the ontology matching (see Figure 5.1, line 5).

5.4.2 Matching-Cut Definition and Semantic Matching

As the sole comparison of used ontologies for Web services does not state any service similarity, the matching cut is defined now for the printing example. To come to a composeability decision too much irrelevant information is available. Unused ontology concepts or useless non-functional parameters do not contribute to the composeability check. The relevant elements are divided in three fractions. The first fraction contains the corresponding concepts used in both service descriptions, e.g. “Person”, “PrinterName”, or “PSFile”. The second fraction contains the non-functional parameter matching parts, which is empty in our example. The third and last fraction contains the functional parameter matching parts. The “PSFile” with “PostscriptFile” matching as described before gets an successful match with a similarity degree of 1.0. According to the weighting with 75% of 1.0 (functional parameters), 25% of zero (non-functional parameters), and 0% of zero points (ontology concepts) the matching-cut has a valuation of 0.75. Due to the once awarded *fully correspondence* the matching process continues.

```
1  <process:AtomicProcess rdf:ID="DocFile2PSFile">
2    <process:hasInput>
3      <process:Input rdf:ID="#DocFile">
4        <process:parameterType rdf:resource="&xsd;base64Binary"/>
5      </process:Input>
6    </process:hasInput>
7    <process:hasPrecondition rdf:resource="#FileExists"/>
8    <process:hasEffect>
9      <process:Effect>
10       <process:Effect rdf:resource="#FileConvertedEffect"/>
11     </process:Effect>
12   </process:hasEffect>
13   <process:hasOutput>
14     <process:Output>
15        <process:parameterType rdf:resource="#PSFile"/>
16     </process:Output>
17   </process:hasOutput>
18 </process:AtomicProcess>
```

Figure 5.8: *Atomic process in the OWL-S service model of the* doc2ps *service*

The next step of the matching process performs the service model matching. Atomic processes are the basis for simple and composite processes and describe the internal processing of a service. The used control structures do not influence the matching result and thus do not improve the matching quality. The IOPEs of the service profiles are compared with the

IOPEs of the atomic processes to identify differences. Only atomic processes using functional parameters appearing in the matching-cut are considered. Figure 5.8 shows the atomic process `DocFile2PSFile` of the *doc2ps* service which uses only functional parameters mentioned in the matching-cut. There is no difference between the IOPEs in the service profile (see figure 5.7) and the IOPEs in the service model. Thus no inconsistencies are detected and *fully correspondence* is given in this example.

Now, the semantic matching parts are completed and the overall correspondence can be calculated. The matching-cut result is weighted with 60% and the service profile and service model matching each with 20%. Each correspondence degrees are grouped in a tuple of (full, maybe, none) correspondence percentage values. In the matching of the *doc2ps* and *printing* service the following result tuple is calculated: The matching-cut concludes a full correspondence (100, 0, 0), the non-functional parameter parts of the service profile matching did not match (0, 0, 100), and the service model matching concludes a full correspondence (100, 0, 0). Weighted and summarized this results in (80, 0, 20) representing two very good matching services. The first tuple component is greater than 75%. This means the service matching is *possible* and therefore the matching process continues.

5.4.3 Syntactic Matching

The atomic processes are mapped to WSDL operations in the service grounding part of the semantic OWL-S service description. The WSDL description is provided with a semantically annotated service, but its information is also provided in the OWL-S service model implicitly and already processed in the service model matching.

```
1    <!-- Messages definitions -->
2    <wsdl:message name="DocFile2PSFile_Input">
3      <wsdl:part name="PSFile" type="&xsd;base64Binary"/>
4    </wsdl:message>
5    ...
6    <!-- PortType definitions -->
7    <wsdl:portType name="DocFile2PSFile_PortType">
8      <wsdl:operation name="DocFile2PSFile_Operation">
9        <wsdl:input message="tns:DocFile2PSFile_Input"/>
10       ...
11     </wsdl:operation>
12   </wsdl:portType>
```

Figure 5.9: *WSDL operation of the* doc2ps *service*

Here, we take the WSDL description as preparation of the Web service composition. The signatures of the functional parameters with its naming and data types of both to be composed Web services are compared. If the data type does not match, a transformation component is needed. The output “PSFile” of the *doc2ps* service will be connected to the “Postscript” input of the *printing* service. The respective WSDL operations and data types in WSDL are identical. Figure 5.9 shows the WSDL operation of the *doc2ps* Web service.

5.4.4 Service Composition

After successfully checking the composeability of the *doc2ps* and the *printing* service the actual composition can be performed. The input parameters of the composed semantic Web service are the sum of all inputs of all internal Web services without the inputs directly served by outputs of other Web services. Due to the used input of the *printing* service in the composition the only input of the composed service is the "DocFile" file. The output parameters of the composed semantic Web service are the sum of all outputs of all internal Web services without the outputs directly served by inputs of other Web services. Here, the only output is "location" of the printed document. Figure 5.10 shows the final composition of the *doc2ps* and *printing* service. The SHESHAGIRI workaround realizes the actual composition and invocation of the composed service.

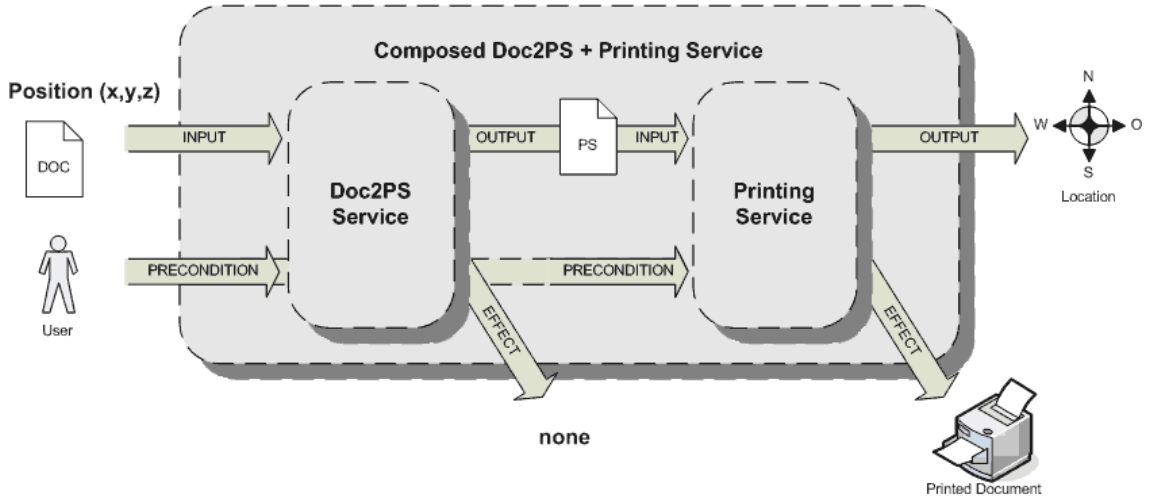


Figure 5.10: *Composed* doc2ps+printing *semantic Web service*

5.5 Summary and Results

The aim of this chapter was to take a thoroughly annotated semantic Web service and understand more of its inputs, outputs, preconditions, and effects. Traditional service discoveries only search for dedicated services matching exactly the user's service requests. With a service composition we want to meet more users and serve better results using the full power of the service's semantic descriptions. The composition can be seen as preliminary step towards a more precise, comprehensive, acceptable, and successful service matching.

With the concept of the OWL-S matchmaker we introduced a complex tool to check the composeability of two semantically annotated Web services. It takes the inputs, outputs, preconditions, and effects into account and reduces the amount of available information to the essential parts with the matching-cut procedure. This allows a good performance and unambiguous results towards a service matching. In [64] we tested several services and partly complex scenarios. We discovered problems if services are imprecisely with their descrip-

tions. For instance, complex data types like an ISBN number are difficult to model, due the lack of modeling constructs in OWL-S. If an ISBN number is modeled with a `xsd:string` data type it is difficult or the matchmaker to match this parameter exactly. Assuming extensive and suitable modeled Web services satisfying matching results are guaranteed.

In the last part of this chapter we presented the concept appliance by describing a detailed example of a *doc2ps* and *printing* service composition.

6 Semantic Service Contextualization

> The important thing in science is not so much to obtain new facts as to discover new ways of thinking about them.
>
> Sir William Bragg

Service contextualization describes the process of integrating context information into service description. This procedure is useful in several ways. As users might seek for any services or service compositions that match their requirements, these generic queries generate rather huge result sets. By using automatically provided context information describing the user's circumstances a lot of mismatches can be eliminated. Also by collecting context information on the user and the service side, new facts and preferences can be concluded. Both aspects have a significant impact on the quality of the matching results.

Chapter 4 describes a way to annotate semantic Web services thoroughly and mostly automatically during the development process of a semantic Web service. This step is necessary for a semantic service matching incorporating the meaning of a service. In chapter 5 we used this annotation to compose services. This allows matching more complex service requests only satisfiable through service combinations. Here, services from different domains with matching semantic descriptions are composed. Therefore, much more service combinations are available delivering a better request matching result. This section considers now the general concept of contexts to improve the matching results further profiting of the additional information contexts can deliver. We propose a contextualized service matching approach to involve the benefits of contexts in semantic service matching. This approach promises a high precision service matching result combining all three mentioned components.

6.1 Upper Context Ontology

In order to share knowledge provided by context an ontological representation is most advantageous. Despite the fact, that an ontology describes the shared understanding of a specific domain, usually different ontologies exist describing the same area. Heterogeneous sources

develop different knowledge representations sometimes due to ignorance of other existing ontologies, sometimes because they do not agree on partial interpretations of the domain knowledge.

Here, we present an extensible context ontology. Due to the evolving nature of context aware computing and the huge variety of systems, sensors, and environmental facts a complete formalization of all context information is infeasible. In other domains huge ontologies are created [155] trying to cover all knowledge in a very small domain suitable for all potential users. In [18] e.g. we defined a location ontology collecting the knowledge what a location is. In the domain of context-aware computing this is impossible. Future developments can lead to completely different structures and facts for contexts, which are today unthinkable. Therefore, we developed a hierarchical structure of context ontologies to overcome this drawback. At a very detailed level, several context ontologies exits to be flexible and extensible for future developments. An upper-level context ontology is now able to connect these domain ontologies with a few very basic concepts all domains can agree upon. This procedure is very common in the ontology modeling area, but has not been applied successfully to context-aware computing so far. Approaches like the CONON ontology [158] or the COBRA ontology [32] suffer from inflexibility and too domain specific context modeling.

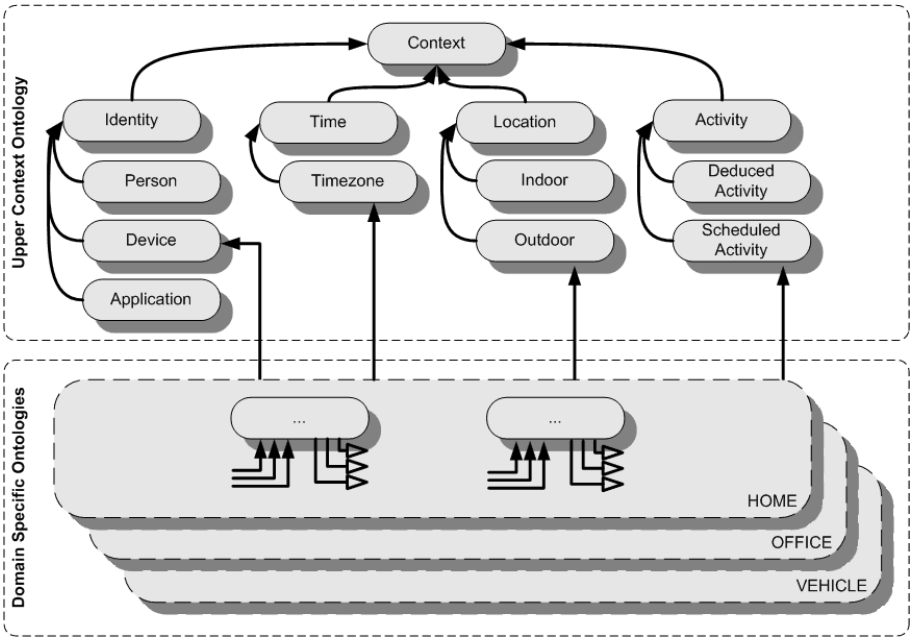


Figure 6.1: *Overview of the context ontology modeling*

We found, that “identity”, “time”, “location”, and “activity” are most fundamental context concepts to capture the information about a situation. These primary contextual entities not only form the skeleton of a context, but also act as indices into associated information. The objectives of our context model include modeling a set of upper-level entities, and providing flexible extensibility to add specific concepts in different application areas.

In realistic context-aware environments, applications, systems, and services are usually grouped as a collection of sub-domains for different environments such as home, office, or vehicle. Context information in each sub-domain shares common concepts such as identity, location, time, or activity that can be modeled using a general context model. While the detailed features in the sub-domains differ significantly, the general context model is highly unspecific and global, but common to all models below, that connections can be made. Therefore, the separation of application domains encourages the reuse of general concepts, and provides a flexible interface for defining specific knowledge.

Summarizing, the upper context ontology is a high-level ontology which captures general features of basic contextual entities. The specific context ontology is a flexible and extensible collection of domain-specific ontologies, which define the details of general concepts and their features in each sub-domain. Figure 6.1 gives an overview of the ontology modeling.

The concrete upper context model is structured around a set of abstract entities representing the primary context types as well as a set of abstract subclasses, see figure 6.1. Each entity is associated with properties (e.g. `Person:name` or `Location:longitude`) and relations with other entities (e.g. `locatedIn`). The hierarchical structuring is done with the `owl:subClassOf` property, thus providing the extensions. Figure 6.2 shows a partial definition of a specific ontology or an office application domain. In the specific context ontology concrete sub-classes are defined to model the given environment closer (e.g. `Building` or `Room` for an indoor location).

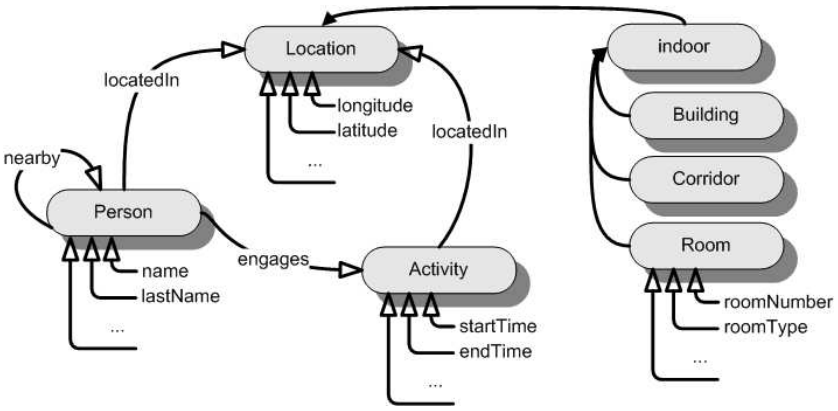


Figure 6.2: *Partial definition of a specific context ontology for an office application*

6.2 Problem Definition

The service contextualization integrates context information of any kind into the service description. Examples can consider *identity* information like the name, surname, or email address of a person. Properties concerning *time* and *location* like a request sending time or the current user's position are often very useful in matching appropriate services. Also *activity* contexts like a user is "reading emails in the office" or "watching a movie at home" help to find more suitable services. The contextualized service matching approach uses this information

to improve the matching quality.

Gathering context information from the current environment of an user or service can be done by sensors. Maintaining this information is essential, because sensor information is only useful if correct and up to date. Often several services or several users share specific context information. For example, a WLAN monitor can provide information about the activity and location of several users in the area. As a context provider this information can be shared and used in several applications and services. With the help of the previously described upper context ontology it is possible to connect different context descriptions on a very high level. Identifying corresponding concepts in heterogeneous context ontologies make it possible to compare context information from different domains. Finally, the contexts help to perform a high quality service matching.

6.3 Context Matching Approach

A clearly defined context ontology is helpful to describe user, system, service, and application environments with a commonly agreed terminology. This thesis makes the approach to use context information for service matching. A semantic Web service delivers annotated information to classify and understand the service. With detailed knowledge of the service interfaces provided with OWL-S individual service composition is possible. A service request depends highly on the current situation of the requestor. Furthermore, the provider offer services depending on the current environment. Both arguments involve context information. For example, the request "print the document on a printer nearby" is difficult to fulfill. Considering semantic Web services together with their contextualized situation makes it possible to accomplish this task. This section describes a service matching approach integrating context information. Therefore, we call it *contextualized* service matching.

6.4 Context Matching Workflow

Figure 6.3 shows the schematic overview of the contextualized service matching process. In a first step context information must be acquired reading for example data from low-level sensors or special middleware facilities. After that this information has to be stored in a proper way (e.g. ontologies) to maintain its information easily. Continually maintenance such as consistency checks of the acquired knowledge is essential. To provide the context information to other applications or services a knowledge sharing step is required afterwards. Here, the context domain ontologies are assigned with a newly developed upper context ontology to share the context information from different domains. In a further step service requestors and service providers have to commit to a specific context domain ontology when describing their requests and advertisements. A context ontology matching is performed afterwards matching the corresponding concepts of the different ontologies. The approach is concluded by the contextualized service matching using the share knowledge of the contexts.

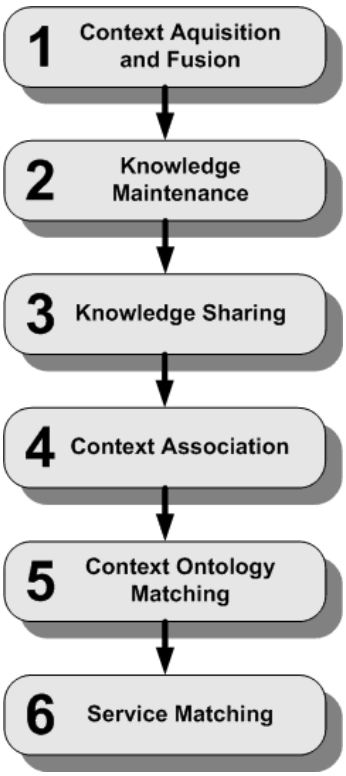


Figure 6.3: *Overview of the contextualized service matching process*

6.4.1 Context Acquisition

Context is any information that can be used to characterize the situation of a person, a device, or a non-computing object. To enable context-awareness, a system must provide users access to context. The process of acquiring context from the physical environment is called *context acquisition* and the first step in this approach.

One approach to enable systems to acquire context is to *access directly low-level sensors*. To access contexts in the real world requires sensors, but thanks to the advance of sensing technology, a diverse type of contexts can be made available to users and other software systems. Many mobile devices have user interfaces to directly access e.g. proximity range sensors or touch sensors. From the proximity range sensors, the device can determine an approximate distance between a physical object and the device itself. From touch sensors a device can determine whether a user is holding it or not and how long. The variety of hardware sensors is immense and adaptable to every situation imaginable. In [89] we introduced a collaboration platform using WLAN access points as proximity sensors. WLAN devices in the range of a specific access point are automatically registered and assigned to local collaboration groups. Each access point has an internal table managing the connected devices holding device specific information and statistics. With SNMP[1] this information is accessible and is retrieved and

[1] Simple Network Management Protocol

used by the collaboration system. However, regardless of which software system acquires the raw data from low-level sensors, individual software applies their own interpretation of contexts. This approach has some potential problems. In order to extend an existing system to consider additional contexts, the system must be modified and the context acquisition procedure rewritten. Unfortunately, in most cases the acquisition procedure is tightly coupled with the application implementation, thus such design discouraged code reuse.

Another approach is to acquire contexts from some kind of *middleware infrastructures* that in turn interact with low-level sensors and users to acquire contexts. To overcome the tight coupling of sensor accessing and application implementation, middleware approaches have been proposed to facilitate context acquisitions [38, 91]. We introduced in [127, 128] the location-based service platform LoBaSS with is a middleware for indoor location-based services sensing WLAN signal strengths and determine a device's position. The system hides the actual location determination. A location component offers the client's position to other services for further processing. This encapsulation makes it possible to develop more general and less specific location-based services and the location sensing technique can be replaced trouble-free with other technologies and techniques. Middleware approaches separate low-level sensing details from high-level implementations. However, in a dynamic environment middleware approaches face knowledge consistency problems when supporting large-scale systems. Furthermore, individual software modules can usually only access context through their own context acquisition components. Because context sensed by low-level sensors can be noisy and possibly inaccurate, inconsistencies and conflicting knowledge are highly possible [41]. Moreover, complex middleware infrastructures cannot be integrated into devices with limited computing resources. Sensing context and interpreting context are often computation intensive operations. Quite this category of small mobile devices is the target application area for context-aware systems, but the resource-poor devices cannot accommodate the demands of middleware infrastructures.

The third and best way for our purposes is to acquire contexts from servers which maintain situational knowledge about the current environment and function as context providers. The idea is to relieve the burden of sensing context and computing context completely from the local clients by shifting these tasks into resource-rich servers. These servers appear as *context providers* and are enable of reasoning contexts as their main advantage. A context server can maintain contextual knowledge on the behalf of a service, software component, or software agent. It aggregates low-level sensor information to deduce additional knowledge. A context server also divides the physical world into specific domains where customized reasoning rules can be applied. A team of context servers can therefore provide contexts from different domains and users or software applications can assign to one context server using the contextual knowledge provided.

6.4.2 Knowledge Maintenance

The knowledge maintenance follows the knowledge acquisition procedure. Knowledge intensive applications with constantly changing knowledge-bases require continually maintenance [63]. This is definitely the case for context ontologies representing facts about highly dynamic environments. For all knowledge acquisition techniques mentioned in the previ-

ously section a continually revision of the contextual knowledge is important. It ensures a consistent and coherent knowledge-base which is fundamental for further processing and upcoming conclusions.

Maintenance activities are triggered when inconsistency in the current ontology is detected in the information adding process or when information is removed by a user, for example for privacy reasons. A typical scenario where inconsistency may occur is a noisy physical environment. The noise results in difficulties while sensing context. The acquired information here my be incorrect and can cause an inconsistent knowledge-base. Another aspect how inconsistencies may occur is that the stored knowledge does not accurately reflect the dynamic changes in the environment.

The first step in this maintenance part of our approach is to detect inconsistencies. Once detected, further actions can take place. Due to the context modeling via ontologies inconsistencies can be detected by regular reasoners or inference engines. We use the RACER reasoner [62] which works excellent with OWL ontologies and its performance is adequate for these small domain context ontologies. For example, we model the rule that "a person cannot be in two different rooms at the same time". The fact "Alice is attending a meeting in room F2.425" does not cause any problems. If another fact like "Alice is using the elevator in building Fuerstenallee 11" is added the knowledge-base becomes inconsistent. Here, the reasoner detects the incorrectness of the ontology.

If inconsistencies are detected several actions can be applied ranging from avoiding inconsistencies, to diagnosing and repairing inconsistencies, to trying to reason in the presence of the inconsistent facts, and to tracking the inconsistencies over the development history of an ontology. [63] gives an nice overview of these techniques. We try to avoid inconsistencies in the first place. A radical approach is to forbid all actions which lead to inconsistent ontologies. The drawback here is to loose the dynamic properties which are typically for context-aware systems and additionally the loss of expressive power of the ontology is substantial. The strategy we consider here is to continually monitor the changes to an ontology and check the impact of the change operation. Note, that an ontology can only become inconsistent by adding facts due to its monotonicity [63].

If anyway an inconsistency is detected, it is localized by determining all unsatisfiable concepts. This subset is the source of the problem. The conflict between the identified entities has to be resolved. This task is difficult, because in most cases there is no unique way of resolving a conflict but a set of alternatives. Often, there is no logical criterion selecting the best solution. Therefore, a common approach is to let the user resolve the conflict.

6.4.3 Knowledge Sharing

To execute the following two steps knowledge sharing and context association we use the concept of *context providers* introduced in the context acquisition step. The context providers act as abstraction of the different software components acquiring and providing context information to specific applications. Here, a context provider acquires information from several sensors, applies it to the internal ontological context model, and continually refreshes the knowledge-base.

This opens the possibility for several applications and services to use the same context

model and due to the semantic contents share the same context interpretation. The context provider holds information about a specific domain stored in a local ontology. This ontology takes care of the semantic interpretations shared in this domain. If two applications use the same context provider application to application data exchange is very easy. The information is interpreted in the same way and therefore no semantic mismatching arises. We call this procedure *knowledge sharing*.

6.4.4 Context Association

A close related step and a consequently continuation of this idea is the process of freely *context association*. A typically publisher and subscriber scenario is used here. Several context providers offer their context information and interpretation for specific domains. Of course other context providers can specify the same information in the same domain differently. In this case two context providers offer the same information, but serve different interpretations and modeling approaches. Applications or services can then subscribe to these providers using the offered data and semantic interpretations which are most suitable to their purpose. By this context association the knowledge is shared among several entities, but the domain specific properties and variety in interpretations are kept alive.

6.4.5 Context Ontology Matching

A side-effect resulting from the previously context association step is the generalization of the context providing. Instead of two services or applications share the same context provider, we can use this technique for general service matching. A service requestor can, in the same way as a service provider, subscribe to a context provider and therefore associate the request like a provider with a domain specific context description. Again, the context providers can be identical, or different to suit the intended interpretations and models.

In the first case, the context providers are identical. Here, no further context ontology matching is needed. The underlying ontologies for the service provider and the service requestor are the same. They share already the same terms, concepts, and interpretations of their contextual information.

In the second case, the context providers and therefore the ontologies to describe the contextual information for service provider and service requestor are different. Similar to the semantic service composition procedure during the composition technique mentioned in the previous chapter the approach from EHRIG and SURE [44] is used to match both ontologies. As result, a new ontology is created containing the shared concepts of both ontologies. Due to the configuration of the procedure of the ontology comparison, the result is more or less successful.

We evaluated again a parameterization with the “DecisionTree” strategy and an iteration count of greater than 15 iterations as most promising. Unfortunately, the context ontologies can be relatively incompatible. To manage this situation we introduced the upper context ontology in section 6.1. The general concepts in this ontology act as connectors between two domain ontologies. A domain context ontology A is compared with the upper ontology with the ontology matcher. The same procedure is done with the second ontology B and the upper

ontology. At last, both results are compared with the ontology matcher resulting in a set of shared concepts of both ontologies A and B.

With this procedure it is possible to find the shared concepts of a service requestor and a service provider even if they use different context providers with different modeling approached to describe their contextual information.

6.4.6 Contextualized Service Matching

The final step in this procedure is the actual usage of the contextual information for the service matching. The preliminary steps aim at constructing the same ontological structure of the context information. They managed to build a connection through the upper context ontology between both service descriptions to be compared. Furthermore, they now share the same terms and concepts still with different properties.

The matching works in a sequential order to compare the available property information. The knowledge sharing and context association steps delivers an ontology for the service request and an ontology for the service advertisement, both connected to the upper context ontology working as synchronization points. The service matching starts on the highest level in the upper ontology. Here, the four primary context types act as nodes separating the below concepts in four branches. The matching process picks one of it, e.g. “location” and tries to match the information contained in the properties belonging to this concept in both ontologies. This comparison works like in the semantic composeability test in the previous chapter. Due to the context association process corresponding service request and service advertisement descriptions are lexicographically identical.

If this high-level concept has a relationship to other concepts more information in this branch is available. The matching proceeds with the next concept by following the relationship descriptions to the next concept and compares its properties with the corresponding concept in the other ontology. This method is repeated until all concepts in this branch are processed. To avoid circles and repetitions a list of already processed concepts is memorized.

If the algorithm picks for example the “location” concept of the upper context ontology, the succeeding concepts are described in domain ontologies (e.g. an office domain ontology) and connected via relationships (e.g. “Building” with “inBuilding”). A further concept in the process is for example “Room” with the “hasRoom” relationship or “Floor” with “hasFloor” which are equally leveled in the ontology modeling, but processed sequential here. At each matching step the concept’s properties are compared.

Which concepts are considered for the service matching depends on the available information and the ontology modeling. Regardless how the information is encoded and how much information is available, the shared ontological representation ensures only available and corresponding concepts described in both participating ontologies are used and compared with each other. In both cases, either successful or unsuccessful matching one concept in a single branch of the ontology, the comparison continues until all corresponding concepts are processed.

After comparing every available context information from both ontologies, the overall matching degree is calculated. This is done by summarizing the number of successful matched properties. Each match is multiplied with its level. The level is defined as shortest

path to one of the four high level concepts in the upper context ontology. E.g if two properties match at the highest level at the "location" concept, it is multiplied times one. If properties on a lower level, e.g. "Building" the multiplier is two, matching "Room" properties are on level three. The same procedure is done with the remaining branches identity, time, and activity.

The matching of one service request with one service advertisement results due to this procedure in a single number. Comparing several advertisements with the same request can therefore be ranked easily. The highest ranked service is the best matching service or the request.

6.5 Concept Appliance: Printing Service Contextualization

In the domain of printing services the following scenario of location-based printing service can be applied again to illustrate the contextualizing service matching approach. A user operates with a notebook using a WLAN network device. When looking for a service to print a postscript document on the "nearest network printer available" different context information is used. The service request uses the context information from a *WLAN* context provider in the WLAN network determining the current location of the notebook within the building. The printing service also uses the locations of all printers in the building provided with current status information from the *printer* context provider. When looking for best matching services to the printing request the context ontologies of both providers are linked together and common concepts are identified. Thus, the location information in the request and the service advertisement can be used to specify the nearest printer available.

6.5.1 Context Acquisition

The first step of the contextualized service matching concerns the process of context acquisition. From the three possible approaches (directly access low-level sensors, use middleware infrastructure for sensors, use abstract context providers) to acquire contexts the printing example chooses the context providers. Here, a *WLAN* context provider holds information about the connected devices in the wireless network. Technical information like the IP and MAC addresses are provided, as well as activity issues like current routing information. A connected positioning system like the component in the LoBaSS system (see chapter 7) can determine the current position of an active device due to signal strength measurements.

On the services side the printing service uses a different context provider. The *printer* context provider holds static information about each network printer in the building. The printer names, types, and capabilities e.g. duplex or color are available in the provided context. To each printer its location is specified via concrete coordinates of the building (e.g. `(23.3, 56.7, 7.3)`) and an informal text description (e.g. `in room F2.510`).

6.5.2 Knowledge Maintenance

The highly dynamic environment of mobile WLAN devices forces the next step of knowledge maintenance. The location information is updated recently to receive accurate service matchings. The knowledge maintenance step ensures here the consistency and coherence

of the knowledge-base. Maintenance activities are activated when inconsistencies are detected. So the first step is to detect inconsistencies. For example the location information of a specific WLAN device is stored in the knowledge-base: "`WLANDevice=nb-kao5` isLocatedAt `PosXZY=(5.3, 20.7, 11.5)` atTime `Time=Tue Sep 7 16:58:05 CEST 2006`". Due to signal noise the positioning can be incorrect. E.g if the same device is located several ten meters away only two seconds later. The device cannot move so fast and therefore the second location must be wrong. Another example is that one device cannot be in two different places at the same time. These rules have to be specified and can be controlled by a regular reasoner. Figure 6.4 shows an example of such a rule in OWL restricting the cardinality of location to one. Inconsistencies can be avoided, ignored, diagnosed and repaired, or even tracked over

```
<owl:Class rdf:ID="WLANDevice">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isLocatedAt"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
        1
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Figure 6.4: *Consistency rule in OWL*

the deployment history to be resolved later. In this example we try to avoid inconsistencies by defining restrictive constraints such as the cardinality of the `WLANDevice` class.

6.5.3 Knowledge Sharing and Context Association

The *WLAN* and *printer* context providers acquire information from their associated sensors, apply them to their internal ontological context model, and continually refreshing their knowledge-bases. Any service or application can freely associate with these context providers to use the provided information such as device locations or printer status. Due to the fact that multiple applications or services can apply to the same context provider, its knowledge is shared among each other.

For example the location concept with relations to coordinates, rooms, floors etc. provided by the ontology of the WLAN context provider can be used by the printer service and the location-based printing service together.

6.5.4 Ontology Matching

The step of context ontology matching depends on how many context providers with different context ontologies are involved in the service matching process. Here, the two WLAN and printer context provider specify their own ontologies to model the context information locally. Whatever context provider the user associates, the second way of ontology matching with

separate ontologies has to be chosen. Here, similar to the ontology matching in chapter 5 the ontologies are compared and the corresponding concepts in both ontologies are identified. Now, shared context concepts between service requestor and all context providers are found.

6.5.5 Contextualized Service Matching

The last step in this procedure of matching contextualized services is the actual service matching. The process starts to compare the available property information of each corresponding ontology concept. For example the concept "location" is one synchronization point in the upper context ontology of both context ontologies from the WLAN and the printer context provider. If the service requestor specifies the location of the user "in room F2.425" due to the WLAN context provider, the same location can be identified in the printer context using the very same location. Here, the printer "lw-f2-kao" is in the same room "F2.425" and therefore the nearest available printer.

6.6 Summary and Results

In this chapter we have described semantic service contextualization. This process of integrating context information into service descriptions and therefore into the service matchmaking process is useful and successful in several ways. Firstly, generic queries generate huge result set which can be reduced easily filtering irrelevant matchings. Secondly, mismatches can be eliminated, and thirdly new facts and preferences about the user and the service can be concluded.

We introduced an upper context ontology in order to combine different context ontologies from different domains in the service matching process. The upper ontology specifies the major concepts identity, time, location, and activity where all other context concepts can be subsumed. This allows comparing different context ontologies from different sources. The context information becomes available through the process of context acquisition where low-level sensors can be accesses directly. We suggest the concept of context providers acting as an abstraction providing context information to users and services. By associating the provided context users and services can share their contextual knowledge. This is used to increase the quality of the service matching.

In the last part of the chapter we presented the concept appliance by describing a detailed example of a printing service contextualization.

7 Prototype and Evaluation

> A complex system that works is invariably found to have evolved from a simple system that worked.
>
> John Gall

In the previous chapters we introduced three major approaches to improve the quality of matching semantically annotated Web services. This is achieved by thoroughly annotating services, checking for composeability possibilities and integrating context information into the matching process. In this chapter we discuss the experiences gained and the lessons learned while developing and implementing the prototype CoBaSS which realizes a context-based service system. An evaluation of the system measuring the matching quality improvement we achieved is difficult. We decided to show the operability of our approach by implementing a prototype and apply it to a general test set. No standard test collection for semantic Web service matching exists yet, but the used OWLS-TC2 test set [93] provides over 500 services from many domains and is commonly agreed to be a good basis. It gives a good impression for a real-world service matching and the possibility to compare it to other approaches.

7.1 Prototype

To implement a prototype, we started developing location-based services as a specialized, but good example for context-based services. We realized the multiple mentioned location-based printing service and many other location-related services. To advertise and manage these services the *Location-based Service System* (LoBaSS) [127–129] was developed. It provides a platform for universal location-based Web services and uses WLAN technology to locate mobile devices with different capabilities, especially indoors. The system was used to make own experiences in WLAN positioning algorithms and served as framework for complex location-based Web services. More details of the system together with further extensions can be found in [127–129]. To gain detailed results in the field of semantic service matching for this work, LoBaSS was extended and modified to handle context-based and context-aware Web services. Furthermore, the OWL-S composition checker and interfaces to context providers were integrated. The resulting CoBaSS system is presented in the following section.

7.1.1 CoBaSS - A Context-based Service System

CoBaSS consists of several components with different functionality; a service-manager, service-matcher, special services like a location- and a map-server, context providers, Web services and a client application. The main component is the *service-manager*. It includes a Web-engine for service presentation and user interaction via standard Browsers. A small *client* application forms another component necessary to make service requests including context information like the current location. We used a signal strength positioning technique to determine the location. The transformation from signal emission measurements to locations in a coordinate system is done in the *location-server* component. The *map-server* is another useful part of the architecture generating maps showing the location of the device and surrounding objects in the close-by vicinity. The *service-matcher* finds suitable services to client requests from the pool of known services. Interfaces to general context providers realize connections to clients, services, and the main manager component integrating context information in the whole process. Several encapsulated Web *services* are administrated by the service-manager and use its contextualization and organization features. These services use the provided context and location information. Figure 7.1 gives an overview of the components which are described in more detail in the following paragraphs.

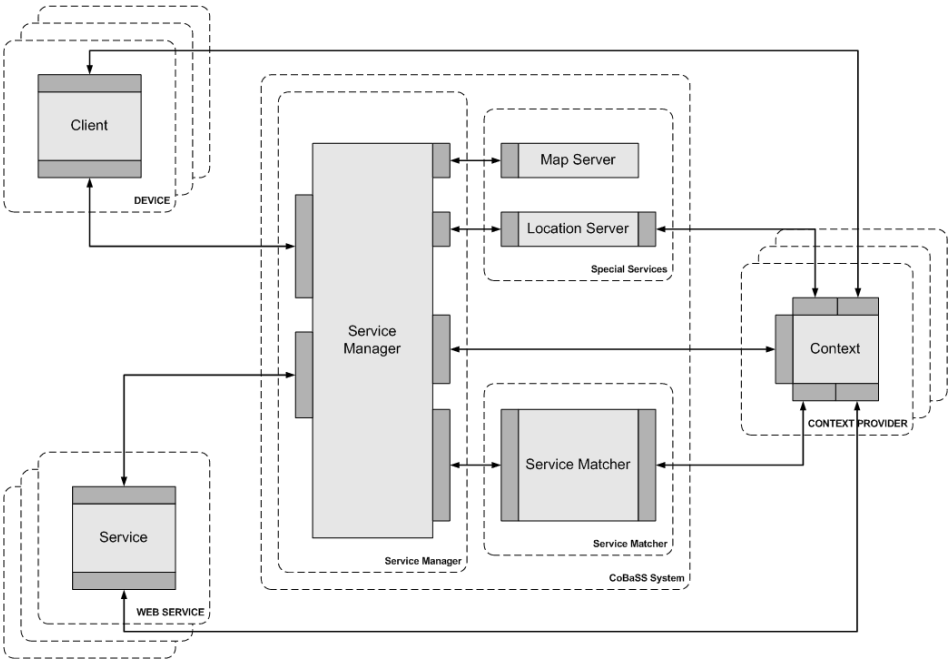


Figure 7.1: *CoBaSS system overview*

The *service-manager* implements the central interface to the system and dispatches all main events from user registration, over service management, to communication bridging. This component includes a Web-engine for different purposes. First, it contains a Web server to advertise the entire system and provides a central portal via standard Browsers for user

interaction. It also acts as registry and gateway to the system. It forwards communication requests to the relevant services and organizes the further client service communication in sessions. A device-dependent presentation for the different device capabilities (e.g. notebook, PDA, or mobile phone) is created with the help of different XSL[1]-stylesheet for different device classes.

The *location-server* implements the core for the position estimation and holds a database with reference values from offline received signal strength measurements. The realized fingerprinting technique combines attenuation and scene analysis techniques to determine the devices location as main context information. Therefore, the location-server receives a vector from the client application with current values of the received signal strength to all available WLAN access points in range. This is compared with the offline values by an enhanced algorithm to estimate the client's positions. As result, the relative location in form of coordinates is returned to the service-manager for further processing. More detailed descriptions and the enhanced algorithm can be found in [129].

The *map-server* component is on the one hand an example for the integration of a basic service into the flexible system architecture. On the other hand it is a frequently used service useful for nearly every other service in the system due to the nature of LBSs, namely illustrating positions. The map-server provides the system with the possibility to generate dynamically multi-layered vector-maps with miscellaneous objects and their positions included. Rooms, floors, devices with dynamic properties and users can be displayed.

The *context providers* gather the location information from the location server and additional context information from all major components. Each context provider holds the information regarding local ontologies and provides upon request detailed context information of services and clients.

The *service matcher* integrates the new approaches of this work by realizing the OWL-S service matcher to check for possible service compositions. Furthermore, it contains the contextualized service matcher, which regards the requested and current context information of client and service using the information from the context providers. The service matcher delivers a result of the best matching services to the service-manager, which presents the result to the requestor.

The *client* itself is usually a mobile device equipped with a wireless LAN communication unit. Due to the broad variety of devices with very differential display capabilities only weak requirements to hard- and software are intended by CoBaSS. The system only needs a Browser which is almost always present in this class of devices. Therefore, any service can display and interact with the user via standard visual capabilities of a Browser. So far client compilations for Windows, Linux, Mac, and WindowsMobile operating systems exist.

At last the system provides an open interface for location-based services using the systems' features. The basic communication between user and service is encapsulated by the service-manager. The location-server delivers the user's position upon service request and the maps created by the map-server can easily be integrated into a new service. So far a location-based print service [127], file-sharing service [128], and public transportation information service [128] have been implemented.

[1] Extensible Stylesheet Language

7.1.2 Environment

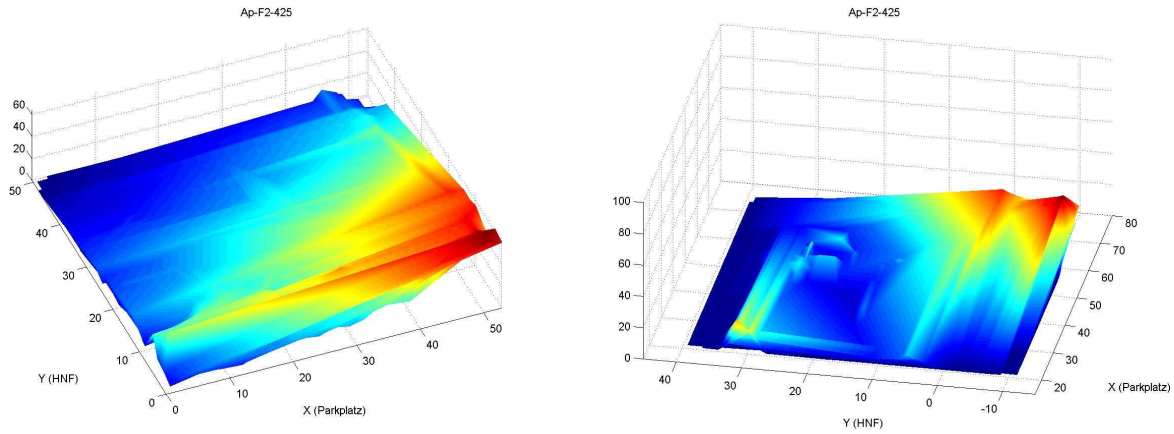
The intentions for the CoBaSS system were at first to make own experiences with WLAN positioning algorithms realizing indoor location-based services. At the beginning the RADAR system and its fingerprinting algorithm described in [10, 11] was taken and implemented, because it seemed most suitable and promised good positioning results. Unfortunately, the original RADAR itself was tested and evaluated in a very small and homogenous environment. This artificial environment is only sufficient for first experiences, but not for true applications. To gain experiences towards a real-world realization, we used the Fürstenallee building of the University of Paderborn for the implemented prototype representing a typical application scenario in an office or industry building. Only by doing this, effects such as the antenna type dependency and floor to floor interferences lead to algorithm improvements compared to the original RADAR algorithm.

The Fürstenallee 11 building accommodates most parts of the computer science department of the university distributed over all three floors. The buildings base is about 53m x 97m. The first and second floor are with 53m x 56m smaller, resulting in an overall expanse of approx. 11000 m^2. The building is equipped with thirteen WLAN access points covering the entire building (four on each floor, one on the roof). Peculiar is the center of the building. An atrium is drawing through the entire building from the base floor to the top which results in difficulties for the positioning approach. Here, access points from many floors and from far away can be received with strong and fluctuating signals. Figure 7.2(a) shows the building and 7.2(b) the atrium.

(a) Fürstenallee building (b) Fürstenallee atrium (c) Floorplan with grid

Figure 7.2: *Testenvironment: Fürstenallee building*

For a fingerprinting positioning approach a precise acquisition of the signal space is indispensable. A homogenous 2.4m x 2.4m grid of measurement points was chosen resulting in approx. 1800 reference points for the entire building. Figure 7.2(c) illustrates the measurement grid. By visualizing the signal space for one access point in figure 7.3 the decision for a fingerprinting approach is confirmed without any doubt. The chaotic signal propagation is obvious. It is not possible to calculate the exact signal propagation considering all obstacles and effects in the building.



(a) AP-F2425 signal on floor F2

(b) AP-F2425 signal on floor F1

Figure 7.3: *Signal strength for the same access point on different floors*

To compare the used environment to others, relevant and objective characteristics can be determined. More meaningful than coverage area, grid size, or quantity of access points are properties such as the access point density, mean access point distance, access point range or access point coverage area. The *access point density* measures the ration between quantity of access points and overall coverage area. The *mean access point distance* is defined by the average distance between two neighbored access points. Finally, the *access point range* and *coverage area* can easily be determined. All these values are strongly related to the algorithm's performance and are calculated in table 7.1 below for the Fürstenallee environment and the original RADAR environment. These values clarify the much more difficult Fürstenallee environment which lead to several smaller, but in the sum significant changes in the fingerprinting algorithm used for the positioning. The RADAR algorithm achieved an accuracy of 3.0m to 4.3m, our changes achieved also a good average accuracy of 3.6m.

Characteristic	CoBaSS	RADAR
coverage area	ca. 11000 m^2	ca. 980 m^2
number of access points	13	3
number of floors	3	1
number of reference points	1800	70
access point density	0.0012 ap/m^2	0.0030 ap/m^2
mean access point distance	28 m	24 m
access point range	40 m	30 m
access point coverage area	220 m^2	320 m^2

Table 7.1: *Environment characteristics of the Fürstenallee building and the* RADAR *site*

7.1.3 Fingerprinting Algorithm

CoBaSS implemented at first the RADAR algorithm described in [10, 11]. Poor positioning results of more than 12m inaccuracy lead to adaptations, extensions and changes in the algorithm. Reasons for the poor performance are mainly the special circumstances in the applied real-world environment mentioned in the next paragraph. Implementation details are presented here. The originally RADAR algorithm can be divided into an off-line and an on-line phase.

During the *off-line phase* the entire working area of the positioning system probed with reference measurements. With regular intervals between two reference points a grid of measurements is created representing the signal space of the radio signal propagation. This so-called *radio map* contains at each point the relative coordinates of the physical space and the signal strengths to each access point and therefore the signal space. These information are stored in a database.

In a second phase, the *on-line* phase, current signal strength measurements on a mobile device at a specific location are determined and compared to all reference points. A metric is used to compare the multi-dimensional vectors in signal space. The best matching reference point according to the metric is computed and its assigned physical location is returned as result of the location estimation for the mobile device. Figure 7.4 illustrates the general procedure of the algorithm with both phases and steps towards a location estimation.

BAHL ET AL. themselves introduced an enhancement in [11] considering more than only one best matching reference point. By triangulating the k best matches the performance increases significantly. Our experiments verify this result. With $k = 5$ the best results were achieved. Taking this setting as basis CoBaSS make additional enhancements and changed in several ways in both phases.

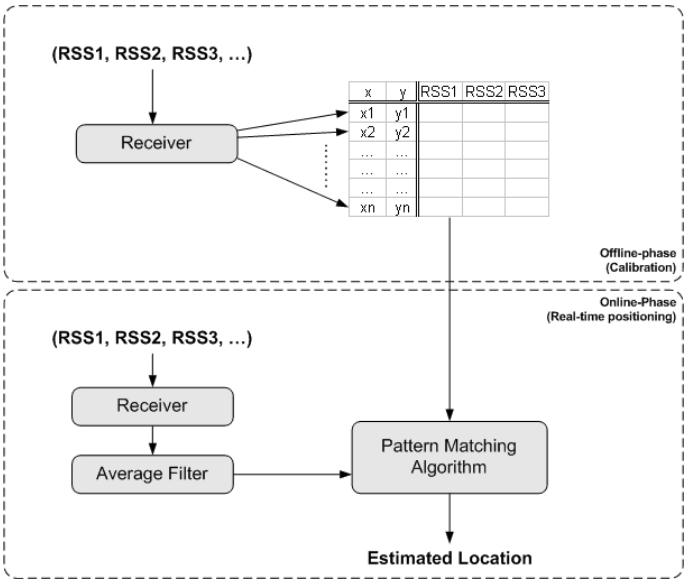


Figure 7.4: RADAR *algorithm and positioning procedure*

First, as mentioned in section 7.1.4 the device orientation has an influence on the signal strength measurement, but it is relatively low. To reduce the amount of measurements this action was abandoned. Instead the grid interval between two reference points is chosen in a way that the resulting positioning accuracy is still accurate enough for the system compared to the expenses of conducting more measurements. In [129] we showed, that the positioning accuracy is in the order of the grid spacing size. The grid layout is shown in figure 7.2 c). The second enhancement is, that at each point the measurement is repeated several times to eliminate dynamic signal irritations. Unrealistic values are erased and the mean value of the remaining measurements is determined as the signal strength for the relevant reference point.

Major changes were also applied in the on-line phase. The device positioning happens by assigning the position of the best matching reference point to the current device position. To determine the best matching point a metric compares the multi-dimensional signal space tuples of the current device SS_c with all reference point tuples $SS_x, x \in R$ in the signal space R. RADAR uses a simple Euclidian distance metric for this purpose.

$$d_{Euklid} = \min_{x \in R} \sqrt{(SS_{x,1} - SS_{c,1})^2 + (SS_{x,2} - SS_{c,2})^2 + \cdots + (SS_{x,n} - SS_{c,n})^2}$$

where $(SS_{c,1}, SS_{c,2}, \ldots, SS_{c,n})$ is the tuple of the current device signal strength measurements to each access point $i = 1 \ldots n$ and $(SS_{x,1}, SS_{x,2}, \ldots, SS_{x,n}), x \in R$ are all tuples in the database (reference points x in signal space R. Our experiences showed that this metric is inappropriate in most situations. It often occurs that the measurements on the mobile device are less precise. Weak access points are often not received and therefore they are $= 0$ in SS_c, but $\neq 0$ in SS_x because the reference measurements were performed with higher quality equipment. Considering this, CoBaSS uses a weighted Euclidian distance with scalars α and β weakening the importance of a specific tuple component if one of the comparing components is equals zero.

$$d_{weightedEuklid} = \left(\sum_{i=1}^{n} \begin{cases} (SS_{x,i} - SS_{c,i}) & \text{if } SS_{x,i}, SS_{c,i} > 0 \text{ or } SS_{x,i}, SS_{c,i} = 0 \\ \alpha(SS_{x,i} - SS_{c,i}) & \text{if } SS_{x,i} = 0 \text{ and } SS_{c,i} > 0 \\ \beta(SS_{x,i} - SS_{c,i}) & \text{if } SS_{x,i} > 0 \text{ and } SS_{c,i} = 0 \end{cases} \right)^{\frac{1}{2}}$$

The last changes were made due to the fact that in the Fürstenallee building as test environment has more than one floor and an atrium. Determining a mobile device's position very close to the atrium causes often high inaccuracy. The estimated position switches between all floors according to the similar signal strength in the open space near the edge of the atrium. This problem is solved by using information from the k best matches result. It the majority of all k matches are on one floor, the estimated position is mapped to this floor. With $k = 5$ this works quiet good. This concerns only the z-coordinate, x- and y-coordinates are calculated via triangulation of all k best matches as before.

The next section presents the results of the modified location estimation approach in CoBaSS. Since the original RADAR algorithm performed poorly with an inaccuracy of 12 meters, an average accuracy of 3 meters in CoBaSS is better than expected handling a difficult application area.

7.1.4 Positioning Results

Besides the resulting accuracy of the applied positioning technique presented in this paragraph, other results were evaluated during the experiments. For instance, observed characteristic WLAN effects have significant influence on the positioning technique. These influencing effects are presented in 7.1.4.

Positioning Accuracy

The primary result for a positioning system like CoBaSS is the accuracy of the positioning. In the environment presented above, several validation points were selected. At least 5% of the amount of reference points on each floor was taken as validation points to determine the accuracy. They are spread randomly over each floor, but it was assured, that "easy" positions to locate (e.g. in the middle of a room or corridor) were considered, as well as "hard" positions (e.g. in niches or corners). For each validation point several location estimation attempts were performed. As result the average accuracy is about 3.12 meters for the whole building. The middle floor F1 performed best, due to the central location with access points on the next floor above and below. Floor F0 performed worst, because the area is twice as big as the other floors holding only the same number of access points. Table 7.2 shows the overview of the results.

Floor	Validation Points	Deviation
average deviation on floor F0	51	3.58 m
average deviation on floor F1	26	2.60 m
average deviation on floor F2	28	3.41 m
overall accuracy	104	3.12 m

Table 7.2: *Average CoBaSS positoining accuracy*

Influencing Factors

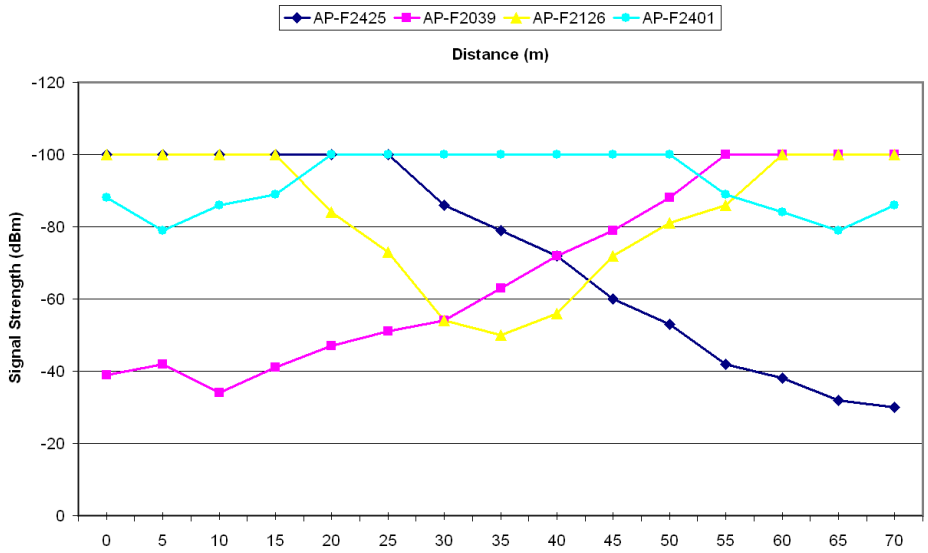
The WLAN signal attenuation is suitable as positioning technique. In [129] we analyzed this suitability closer and detected the usual radio signal effects such as attenuation, shadowing, reflection, refraction, and scattering. With the implementation of the CoBaSS system other, unforeseen influencing factors had to be examined. In [129] we observed that the signal strength at the same place in the building varies over time. The variation lies below the tolerance of the signal strength measurement (1-3 dBm[2]).

Floors, ceilings, walls or their explicit absence in the case of doors, windows, or atriums are *static obstacles* in signal propagation, but produce stable and reproducible signal strength at the same place. The signal strength definitely is a function of the distance to access points (see figure 7.5 (a)) as introduced in [10]. Likewise changes in *atmospheric conditions* like humidity, temperature, or cellular respiration (range of an access point changes with the number of

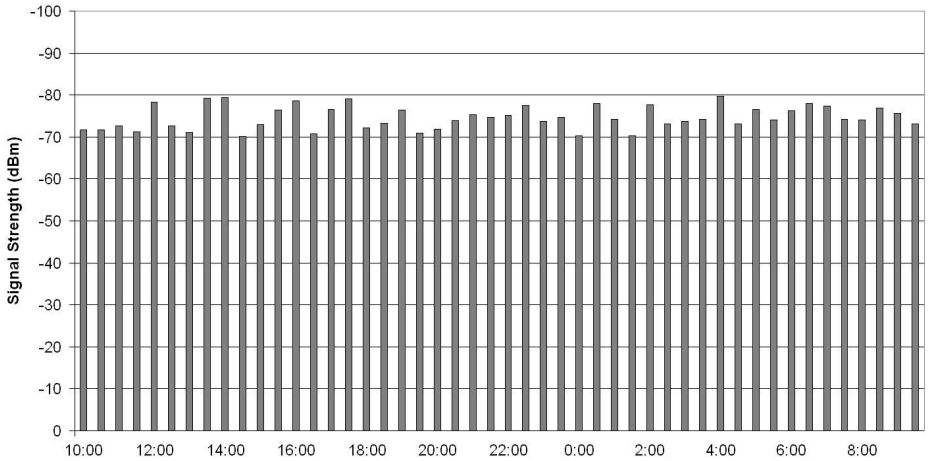
[2]dBm is the ratio between two measurements of electrical power (decibel) referenced to 1 mW

connections) occur in WLAN networks [124], but could not be verified to have a significant effect on the signal strength. The effects were also below signal tolerance, see figure 7.5 (b).

Dynamic obstacles are for instance humans passing by who have a significant signal influence. This can also be recognized by the orientation of the device's antenna. In one direction the user is in the line-of-sight between antenna and the access point, in another direction the line-of-sight is not disturbed. The CoBaSS system manages this influence by collecting several data sets of signal strength measurements in a short period of time. Deleting runaway measurements and taking the average signal strength for the short interval eliminates most dynamic signal displacements.



(a) Signal strength to distance ratio



(b) Long-time WLAN measurement

Figure 7.5: *Signal strength measurements*

The *product diversity* and manifoldness of WLAN device manufacturers is enormous. The different chipsets on the market integrated by the manufacturers and even devices identical in construction receive the WLAN signal at the same place in different signal strengths. The *construction type of the WLAN antenna* has also a significant influence on the received signal strength. Three major types can be distinguished. First, mobile devices usually have network device connected to a USB[3] or PCMCIA[4] slot. These kinds of devices have an integrated small antenna. For the second type, notebooks and similar devices usually have integrated antennas along the device's display. These antennas are larger compared to the expansion cards and have better quality properties. The best results can be achieved with the third type, stand-alone antennas connected to the device. They are usually omnidirectional and have low damping characteristics [124]. Although different devices result in different received signal strengths the measurements with the same device at the same place are reproducible [129]. The CoBaSS system takes this into account by implementing the fingerprinting algorithm only with relative signal strength values. To determine the difference between reference values and current WLAN device measurements the algorithm compares only the relative distance in signal space and not absolute values. Table 7.3 summarizes all influence factors, opposes the significance of the effect, and sketches the CoBaSS adaptation to compensate the effect.

Influence Factor	Significance	CoBaSS Adaptation
static obstacles (walls etc.)	high	fingerprinting algorithm
dynamic obstacles (e.g. humans)	low	average signal computation
atmospheric conditions	none	no adaptation
cellular respiration	none	no adaptation
antenna orientation	medium	no adaptation
antenna construction type	medium	relative comparisons
device manufacturer	medium	relative comparisons

Table 7.3: *Influence Factors and adaptations of CoBaSS*

7.2 Evaluation

The three major approaches proposed in this work improving the matchmaking of semantically annotated Web services need to be evaluated to be compareable to other works. The OWL-S matchmaker for the composeability checks as well as the contextualized matchmaking procedures were integrated in the matchmaking component of CoBaSS. As test set we use the OWLS-TC2 [93] developed by the DFKI[5] which is available as open source[6]. So far, no standard test collection for OWL-S service matchmaking does exist. Other matchmaking approaches usually use small self-created test scenarios which make them difficult to compare objective. The OWLS-TC2 is the best alternative towards the goal of independent evaluation

[3] Universal Serial Bus
[4] Personal Computer Memory Card International Association
[5] German Research Center for Artificial Intelligence
[6] `http://projects.semwebcentral.org/projects/owls-tc`

and compatibility containing over 500 real-world semantic Web services from different domains.

7.2.1 OWL-S Test Collection

The OWLS-TC2 test collection is available in the second version from December 2005. It provides 576 semantic Web services written in OWL-S 1.1. An OWL-S 1.0 description is also available for backward compatibility. The services origin from seven different domains, namely education, medical care, food, travel, communication, economy, and weapon. The majority of these services were retrieved from public IBM UDDI registries, semi-automatically transformed from WSDL to OWL-S, extended with appropriate input and output concepts, and then stored locally for the test collection. The collection also provides a set of 28 test queries each of which is associated with a relevance set of 10 to 20 services. OWLS-TC2 works locally installing a simple local Web server with all services, descriptions, ontologies, and queries in subdirectories. This is necessary, since all ontologies expected to be available locally by several semantic matchmakers, and problems with incorrect paths to access ontologies can thus be avoided.

The services in the collection underlie some restrictions. The process model contains atomic processes only. Groundings in WSDL are not included and the service profiles are kept minimal restricted only to ServiceName, textDescription, hasInput, and hasOutput. Effects and preconditions are not described. Unfortunately, some of this restrictions effect the matching approaches in this work, but their influence is appreciable. The number of services and queries are related to each of these domains are as follows:

Domain	#Services	#Queries
education	135	6
medical care	52	1
food	25	1
travel	106	6
communication	29	2
economy	206	11
weapon	25	1

Table 7.4: *Number of domains, services, and queries in the OWLS-TC2 collection*

The collection contains 28 queries representing different users requesting services. The queries are formulated via OWL-S service profiles and specify certain inputs and outputs relevant to the user. For example query number four represents a user who is interested to know the price of a car and a bicycle and is specified in the file `1personbicyclecar_price_service.owls`. The query describes two inputs (`CAR` and `1PERSONBICYCLE`) and one output `PRICE` including all necessary ontology links and an atomic process description. Further details to all queries and services can be found in [93]. Table 7.5 states all specified queries and their corresponding domains.

No.	Domain	Query
1	economy	car_price_service.owls
2	economy	book_price_service.owls
3	economy	dvdplayermp3player_price_service.owls
4	economy	1personbicyclecar_price_service.owls
5	economy	bookpersoncreditcardaccount_price_service.owls
6	economy	maxprice_cola_service.owls
7	economy	bookpersoncreditcardaccount__service.owls
8	economy	shoppingmall_cameraprice_service.owls
9	economy	recommendedprice_coffeewhiskey_service.owls
10	economy	userscience-fiction-novel_price_service.owls
11	economy	preparedfood_price_service.owls
12	food	grocerystore_food_service.owls
13	communication	title_comedyfilm_service.owls
14	communication	title_videomedia_service.owls
15	education	researcher-in-academia_address_service.owls
16	education	university_lecturer-in-academia_service.owls
17	education	governmentdegree_scholarship_service.owls
18	education	publication-number_publication_service.owls
19	education	novel_author_service.owls
20	education	country_skilledoccupation_service.owls
21	medical care	hospital_investigation_service.owls
22	travel	surfing_destination_service.owls
23	travel	surfinghiking_destination_service.owls
24	travel	geographical-regiongeographical-region_map_service.owls
25	travel	surfingorganization_destination_service.owls
26	travel	citycountry_hotel_service.owls
27	travel	geopolitical-entity_weatherprocess_service.owls
28	weapon	governmentmissle_funding_service.owls

Table 7.5: *List of queries in the OWLS-TC2 collection*

The queries in the collection are each associated with a set of 12 to 21 services that are subjectively defined as “relevant” or “not relevant”. A service is judged “relevant” if any piece of it is relevant, regardless of how small the piece is in relation to the rest of the service description. Table 7.6 shows the set of relevant services of the test collection for query number four. The test set also provides string similarity values calculated for each service in the set to each query. These values give a clue on how a non-semantic matchmaker will perform. The semantic descriptions can be used and compared lexicographically without any semantic meaning. Three different similarity metrics used in the test set are *Cosine (Cos)*, *Extended Jacquard (EJ)*, and the *Jenson-Shannon (JS)* similarity metric [94]. These are the most prominent string similarity metrics for pure text-based information retrieval.

Service	Cos	EJ	JS	OWLS-MX4	OWLS+ Context
1personbicyclecar_price_service.owls	0.98	1.00	1.00	exact	match
1personbicyclecar_price_Kohlservice.owls	0.97	0.96	0.98	exact	match
1personbicyclecar_price_TheBestservice.owls	0.97	1.00	1.00	exact	match
auto1personbicycle_price_service.owls	0.97	0.96	0.97	plugin	match
bicyclecar_priceyear_service.owls	0.81	0.72	0.81	plugin	match
carcycle_price_service.owls	0.95	0.95	0.97	plugin	match
carbicycle_recommendedprice_service.owls	0.93	0.89	0.94	plugin	match
cyclecar_pricetaxedprice_service.owls	0.93	0.84	0.95	plugin	match
cycle1personbicycle_price_service.owls	0.71	0.63	0.75	plugin	match
cyclecar_recommendedpriceineuro_service.owls	0.90	0.87	0.92	subsumes	match
1personbicycle4wheeledcar_price_service.owls	0.97	0.96	0.97	failed	match
4wheeledcarbicycle_price_service.owls	0.96	0.99	0.99	failed	match

Table 7.6: *Relevant set of services to query 4 in the OWLS-TC2 collection*

The last both columns in table 7.6 represent the successful or unsuccessful matching result of two matchmakers. One is the matchmaker integrated in CoBaSS realizing the OWL-S Matchmaker including the contextualized matching concepts named *OWLS+Context*. The OWLS-MX [94] matchmaker is the only other matchmaker so far evaluated its experimental results with the OWLS-TC2 test set. It is a hybrid approach combining logic-based reasoning and content-based information retrieval techniques. The results to query four of the OWLS-MX variant number four are also listed in table 7.6. The result of their total comparison is described in the next section.

7.2.2 Matchmaker Performance

The contextualized OWL-S matchmaker from the CoBaSS system was tested with the services, queries, and service descriptions from the OWLS-TC2 test set. Figure 7.6 shows the summarized matching result for each of the 28 queries matching all 576 services in the test set. In the figure the percentage of successful service matching of relevant services is illustrated. Additionally to the OWLS+Context and the OWLS-MX4 matchmaker results the purely syntactically working Jenson-Shannon (JS) similarity is shown. If the JS value is greater than 0.7 the service is rated as a match, because it can be seen as sufficiently similar. Sometimes this text-based retrieval technique outperforms both semantic working matchmakers due to the simple queries. But in the majority of all queries, the benefit of the semantic descriptions exceeds the results of the JS.

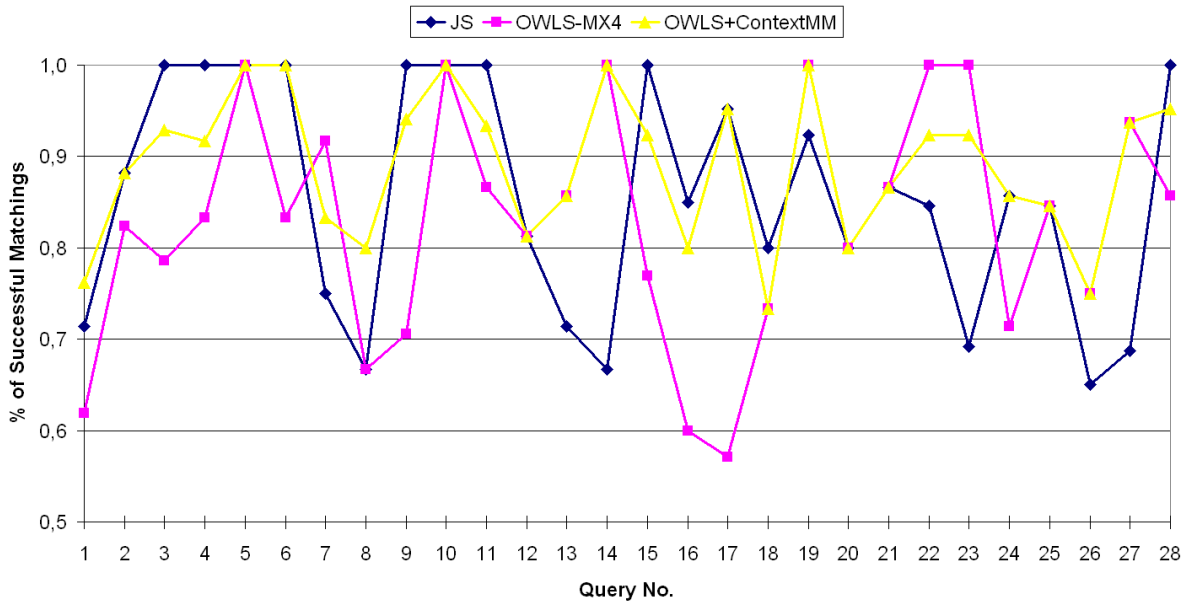


Figure 7.6: *Total matching result of contextualized OWL-S matchmaker for all OWLS-TC2 queries*

The general better performance of the OWLS+Context matchmaker in comparison to the OWLS-MX4 matchmaker can be explained with the integrated matching-cut definition and the used context information. Although the environment in the test set barely includes elements which emphasis the benefits of integrated context information or complete OWL-S descriptions, the matchmaker takes advantage of the semantic reduction in the matching-cut process.

To get an impression on the quality of the matchmaker's performance different measures have to be considered. There are various ways to measure how well the retrieved information matches the intended information. To evaluate this quality of a retrieval result the most frequent used measures are recall and precision. *Recall* is the ratio of the number of relevant entities retrieved to the total number of relevant entities. *Precision* is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. Therefore, with recall we answer the question of "have I found the most relevant material or am I missing important items" (i.e. avoiding false negatives) and with precision we consider "how much junk have I retrieved" (i.e. avoiding false positives).

Figure 7.7 shows the recall-precision-graph. The micro-averaged recall-precision curve shows again the results of both semantic matchmaker OWLS+Context and OWLS-MX4 as well as the JS performance. Both are superior to the text-based retrieval. The OWLS+Context matchmaker performs close to the OWLS-MX4 matchmaker in high precision as well as high recall areas. However, for discovery of semantic Web services, precision may be more important to users than recall, since the set of relevant services is subject to continuous change in practice.

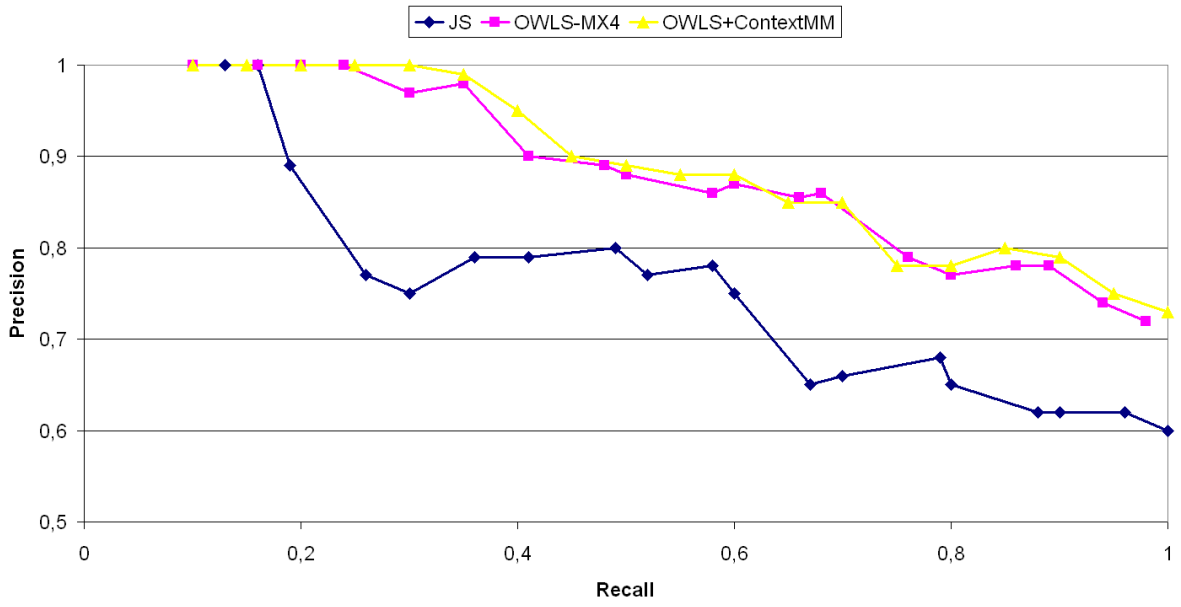


Figure 7.7: *Recall-precision performance of OWL-S matchmaker with contextualization*

7.3 Summary

In this section, we have presented details of the prototype implementation and a thoroughly evaluation of the matchmaker component which shows a good performance of the methods presented in this thesis.

We have shows that it is possible to develop a prototype for complex location-based and context-based services. The position in indoor environments can be used easily without expensive hardware sensors. The accuracy of approximately three meters is sufficient enough to use WLAN signal strength measurements for an indoor positioning system.

We have further evaluated the matchmaker component integrated in the CoBaSS system. It implements the OWL-S Matchmaker approach including the matching-cut definition and integrates the contextualized service matching aspects. The evaluation with the OWLS-TC2 test set containing over 500 semantically annotated Web services was successful. A Recall-Precision analysis showed significant improvement of our approaches compared with other semantic matchmaking approaches. Therefore, the goal of this thesis to increase the quality of semantic service matching is achieved.

8 Conclusions and Outlook

> Luck is being in the right place at the right time, but location and timing are to some extent under our control.
>
> Natasha Josefowitz

The open world of wireless, mobile, and pervasive computing brings information to the finger tips of human beings *anytime* and *anywhere*. More and more devices we carry around are networked and provide us with a never ending stream of information. They range from mobile phones and PDAs with wireless network access to information and entertainment systems in modern cars. The network capability of these devices helps to publish their local knowledge to any person or service interested and receive global knowledge or use information from external services. But this information is only valuable if the surrounding *context* is also captured and the meaning of the information can be understood in its context.

The biggest source of information is the Web. Billions of pages provide information about virtually everything you can imagine. However, this information is pure text presented without any background. Web applications provide their Web pages usually in HTML format only. The content of the Web pages is expressed in natural language, weakly structured with HTML tags, and its semantics not accessible to machines. Because information is only meaningful in its context, it is left to humans to extract the content and combine it with the context of the page to understand the meaning of the information.

In machine-to-machine and human-to-machine interaction *Web services* play an important role realizing a dynamic information source. The more information is available, the harder it is to locate the particular piece one is looking for. The major challenge is to discover and retrieve only relevant information and understand its meaning due to the semantic background and the surrounding context.

The goal of the Semantic Web initiative is to provide tools to make the Web's content machine-processable and give the information on the Web a well defined meaning. It is formalized using semantic meta-data that is based on concepts defined in ontologies. Therefore, the existence of semantically annotated Web pages and Web services is crucial to realize a high quality information discovery.

8.1 Summary

This dissertation addresses the issues of automatic service annotation, service composition, and service contextualization in order to improve the quality of semantic service matching. We used the concept of *context-aware services* with an special focus on *location-based services* as an practical application of our research. Context is a key issue in the integration between humans and computers adding meaning to something. Location is one of four primary components identity, location, time and activity which characterize the context of a situation. An indoor positioning via WLAN signal strengths was identified to be an interesting approach to automatically sense the location of an mobile device within a building. We used among other things the sensed location information to automatically fill context descriptions to create location-based services with an expressive service description.

A thoroughly created service description helps to make the meaning of a service processable to machines. RDF is only sufficient to describe resources. Ontologies, for example specified in OWL, have a greater expressiveness and provide a greater pool of modeling primitives. In the respect of describing services, OWL-S turned out to be the best choice providing a good opportunity to combine Semantic Web technologies with Web service technologies. It allows creating machine-processable service descriptions in order to automatically discover, compose, and execute Web services. To realize a proper semantic service matching a thorough *semantic description* is essential. The process of building this description requires usually a lot of human involvement. It is not possible to automatically derive a semantic service description from interface descriptions like WSDL or even from the service's source code due to the lack of detailed semantic meta-information. We propose a mostly automatic annotation framework supporting the service developer in the annotation process. The framework allows to gather all necessary semantic meta-data during the service development process directly from the developer. The proposed new WSDL-to-OWL-S algorithm is then able to create a corresponding semantic service description.

Real-world Web services live on their global availability and accessibility. The problem is usually to advertise a Web service properly, so its intentions and function can be understood and the service itself can be found easily. In traditional service discovery a request fails instantly if keywords in the request do not match the manual created service description. With semantic annotated services the meaning of the service is machine-processable and can be understood. In many cases the precise inputs and outputs of a service do not meet a request exactly. Therefore, this thesis considered the issue of *service composition* in order to enhance the quality of service matchings. The semantic annotation allows a much more advanced and precise service composition than in traditional approaches. The amount of semantic information concerning all elements towards a semantic service composition is enormous. Thus, we developed a matching-cut method to reduce this amount to only relevant information for a successful service composition. It is integrated in an OWL-S matchmaker which also considers all participating ontologies, the OWL-S service profiles, service models, and WSDL descriptions. The matchmaker is able to check for composeability of two semantic Web services from different domains. It considers all inputs, outputs, preconditions, and effects by reducing the available information to the essential parts with the matching-cut procedure and thus improves the quality of the service matching.

With growing numbers of services in registries and possible service compositions it gets harder to successfully match the appropriate service meeting a user's request. Huge result sets are the consequence. However, user's and service's contexts contain descriptions of their individual environment. Service requests depend highly on the current situation of the requestor. Thus, the contribution of *context information* in the matchmaking process can improve the matching result significantly. The semantic service contextualization in this thesis takes this argument and integrates context descriptions into the OWL-S matchmaker. With the help of a new developed upper context ontology and the concept of context providers, requestors, and service provider from different domains can use their context information. This improves, once again, the quality of semantic service matching.

Finally, this thesis includes details of an implemented *prototype*, the CoBaSS system and an *evaluation* of the matchmaking component in a large Web service test set. CoBaSS is a context-based service system intentionally started to give indoor location-based services a try. Here, we enhanced a positioning technique using WLAN signal strength measurements in combination with a fingerprinting algorithm. This results in a sufficient and accurate indoor positioning system delivering positions of Web service users. CoBaSS was then extended to a semantic and context-aware service system processing semantic service descriptions and context information such as identity, time, location and activity.

The evaluation of the matchmaking approach is accomplished by the usage of the OWLS-TC2 test set. It contains over 500 semantic annotated Web services including sample queries and relevancy service sets. The analysis showed that although the test set does not integrate any context information the performance of the matchmaker is good. The matching-cut method with the aid of ontology matching and service model matching improves the service matching quality significantly. Recall-precision analysis showed also a good performance in all aspects of the matching results compared to other matchmaking approaches. Therefore, the goal of this thesis to improve the quality of semantic service matching is succeeded.

8.2 Outlook

We have designed a system for high quality semantic Web service matchmaking and made a sufficient contribution to this area. However, the field of semantic matchmaking is broad and many aspects needs still to be investigated. We have concentrated on service annotation, context integration, and composition aspects. A number of issues were left out, but are interesting to examine in the future.

Possible directions for future research consider new approaches in the still young Semantic Web area. The new WSDL 2.0, SAWSDL, or SWRL languages are most promising contributions and worthwhile to investigate. Some aspects such as privacy concerns especially in the service composition area were ignored in this thesis. Here, much work can be done and various improvements can be achieved.

The used service test set to evaluate the performance of the matchmaker does not include any context information. This is needed to unfold the full potential of our matchmaking approach. Thus, any support in this direction will further expose the benefits of contextualized service matchmaking. The semantic Web service community recently announced the need

for a semantic Web service matchmaker contest. Future efforts are surely to compete at such an event with our approaches.

Finally, the implementations of CoBaSS are still in a prototype stage. Overall system performance and scalability aspects were so far only secondary goals. Here, further improvements are planned.

List of Figures

List of Tables

Bibliography

[1] ALONSO, G. ; CASATI, F. ; KUNO, H.: *Web Services – Concepts, Architectures and Applications*. Springer, 2003. – ISBN 3–54–044008–9

[2] ANDREWS, T. ; CURBERA, F. ; DHOLAKIA, H. ; GOLAND, Y. ; KLEIN, J. ; LEYMANN, F. ; LIU, K. ; ROLLER, D. ; SMITH, D. ; TRICKOVIC, I. ; WEERAWARANA, S.: *Business Process Execution Language for Web Services (BEPL4WS)*. – `ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf`

[3] ANTONIOU, G. ; HARMELEN, F. v.: *A Semantic Web Primer*. MIT Press, 2004. – ISBN 0–26–201210–3

[4] APACHE JAKARTA PROJECT: *The Apache eXtensible Interaction System (AXIS)*. – `http://ws.apache.org/axis/`

[5] APACHE JAKARTA PROJECT: *Apache Tomcat*. – `http://jakarta.apache.org/tomcat/`

[6] APACHE JAKARTA PROJECT: *The Axis C++ WSDL2Ws Tool*. – `http://ws.apache.org/axis/cpp/arch/WSDL2Ws.html`

[7] ARKIN, Assaf ; ASKARY, Sid ; FORDIN, Scott ; JEKELI, Wolfgang ; KAWAGUCHI, Kohsuke ; ORCHARD, David ; POGLIANI, Stefano ; RIEMER, Karsten ; STRUBLE, Susan ; TAKACSI-NAGY, Pal ; TRICKOVIC, Ivana ; ZIMEK, Sinisa: *Web Service Choreography Interface (WSCI) 1.0*. – `http://www.w3.org/TR/2002/NOTE-wsci-20020808`

[8] Ascension Technologies Inc.: *MotionStar User Manual*. – `http://www.ascension-tech.com`

[9] AT&T: *Find Friend Application*. – `http://www.attws.com/mmode/features/findit/FindFriends/`

[10] BAHL, P. ; PADMANABHAN, V. N.: RADAR: An In-Building RF-based User Location and Tracking System. In: *Proceedings of the IEEE Conference on Information and Communication (InfoComm'00)*, 2000, S. 775–784

[11] BAHL, P. ; PADMANABHAN, V. N. ; BALACHANDRAN, A.: Enhancements to the RADAR User Location and Tracking System / Microsoft Research. 2001 (MSR-TR-2000-12). – Forschungsbericht

[12] BARNES, Stuart J.: Location-Based Services: The State-of-the-Art. In: *e-service Journal* 2 (2003), Nr. 3, S. 59–70

[13] BARTEL, M. ; BOYER, J. ; FOX, B. ; LAMACCHIA, B. ; SIMON, E.: *XML-Signature Syntax and Processing*. – http://www.w3.org/TR/xmldsig-core/

[14] BAUER, J.: *Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic*, Technical University of Berlin, Diplomarbeit, 2003

[15] BECHHOFER, S. ; HARMELEN, F. v. ; HENDLER, J. ; HORROCKS, I. ; MCGUINNESS, D. L. ; PATEL-SCHNEIDER, P. F. ; STEIN, L. A.: *Web Ontology Language (OWL)*. 2004. – http://www.w3.org/TR/owl-ref/

[16] BERESFORD, A.R. ; STAJANO, F.: Location Privacy in Pervasive Computing. In: *IEEE International Magazine on Pervasive Computing* 2 (2003), Nr. 1, S. 46–55

[17] BERNERS-LEE, T. ; HANDLER, J. ; LASSILA, O.: The Semantic Web. In: *Scientific American* 284 (2001), Nr. 5, S. 28–37

[18] BIRKENHEUER, Georg: *A Framework for Semantic Web Service Development*, University of Paderborn, Diplomarbeit, 2006

[19] BIRMAN, Ken: The Untrustworthy Web Service Revolution. In: *IEEE Computer Magazine* 1 (2006), Nr. 1, S. 98–100

[20] BLUETOOTH SPECIAL INTEREST GROUP: *Bluetooth*. – http://www.bluetooth.org

[21] BOHN, Jürgen ; VOGT, Harald: Robust Probabilistic Positioning Based on High-Level Sensor-Fusion and Map Knowledge / ETH Zürich. 2003 (ETH-TR-421). – Forschungsbericht

[22] BOOTH, D. ; HAAS, H. ; MCCABE, F. ; NEWCOMER, E. ; CHAMPION, M. ; FERRIS, C. ; ORCHARD, D.: *Web Services Architecture*. – http://www.w3.org/TR/ws-arch/

[23] BOOTH, D. ; LIU, C. K.: *Web Services Description Language (WSDL) Version 2.0 Primer*. – http://www.w3.org/TR/wsdl20-primer/

[24] BRAY, T. ; PAOLI, J. ; SPERBERG-MCQUEEN, C. M. ; MALER, E. ; YERGEAU, F.: *Extensible Markup Language (XML) 1.0 Specification*. – http://www.w3.org/TR/2004/REC-xml-20040204/

[25] BRICKLEY, D. ; GUHA, R.V.: *RDF Schema Specification 1.0*. – http://www.w3.org/TR/2000/CR-rdf-schema-20000327/

[26] BROWN, P. J. ; BOVEY, J. D. ; CHEN, X.: Context-aware Applications: From the Laboratory to the Marketplace. In: *IEEE International Journal on Personal Communications* 4 (1997), Nr. 5, S. 58–64

[27] BRUIJN, J. de ; LAUSEN, H. ; KRUMMENACHER, R. ; POLLERES, A. ; PREDOIU, L. ; KIFER, M. ; FENSEL, D.: *Web Service Modeling Language (WSML)*. – `http://www.wsmo.org/TR/d16/d16.1/v0.21/`

[28] CASATI, Fabio ; SHAN, Eric ; DAYAL, Umeshwar ; SHAN, Ming-Chien: Business-oriented Management of Web services. In: *International Journal Communications of the ACM* 46 (2003), Nr. 10, S. 55–60

[29] CASTRO, P. ; CHIU, P. ; KREMENEK, T. ; MUNTZ, R.: A Probabilistic Room Location Service for Wireless Networked Environments. In: *Proceedings of the IEEE Conference on Ubiquitous Computing (UbiComp'01)*, 2001, S. 18–34

[30] CHAN, A. T. S. ; LEONG, H. V. ; CHAN, J. ; HON, A. ; LAU, L. ; LI, L.: BluePoint: A Bluetooth-based Architecture for Location-Positioning Services. In: *Proceedings of the ACM International Symposium on Applied Computing (SAC'03)*, 2003, S. 990–995

[31] CHARTERS, Stuart M. ; HOLLIMAN, Nicolas S. ; MUNRO, Malcolm: Visualisation on the Grid: A Web Service Approach. In: *Proceedings of the UK e-Science All Hands Meeting*, 2004, S. 103–111

[32] CHEN, H. ; FININ, T. ; JOSHI, A.: Using OWL in a Pervasive Computing Broker. In: *Proceedings of the Workshop on Ontologies in Open Agent Systems (AAMAS'03)*, 2004, S. 97–104

[33] COGNITIVE SCIENCE LABORATORY PRINCETON UNIVERSITY: *Wordnet Project*. – `http://wordnet.princeton.edu`

[34] CONNOLLY, D. ; HARMELEN, F. v. ; HORROCKS, I. ; MCGUINNESS, D. L. ; PATEL-SCHNEIDER, P. R. ; STEIN, L. A.: *DAML+OIL Reference Description*. – `http://www.w3.org/TR/daml+oil-reference/`

[35] CROW, B. P. ; WIDJAJA, I. ; KIM, J. ; SAKAI, P.: IEEE 802.11 Wireless Local Area Networks. In: *IEEE Communications Magazine* 35 (1997), Nr. 9, S. 116–126

[36] DACONTA, Michael C.: *The Semantic Web*. John Wiley and Sons, 2003. – ISBN 0–47143–257–1

[37] DEARMAN, David ; HAWKEY, Kirstie ; INKPEN, Kori M.: Effect of Location-awareness on Rendezvous Behaviour. In: *Proceedings of the Conference on Human Factors in Computing Systems (CHI'05)*, 2005, S. 1929–1932

[38] DEY, A. K.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. In: *International Journal on Human-Computer Interaction* 16 (2001), Nr. 2, S. 97–166

[39] DEY, A. K. ; ABOWD, G. D.: Towards a Better Understanding of Context and Context-Awareness / Georgia Institue of Technology. 1999 (GIT-GVU-99-22). – Forschungsbericht

[40] DEY, A. K. ; ABOWD, G. D. ; WOOD, A.: CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services. In: *International Journal on Knowledge-Based Systems* 11 (1999), Nr. 1, S. 3–13

[41] DEY, Anind K. ; MANKOFF, Jen ; ABOWD, Gregory D.: Distributed Mediation of Imperfectly Sensed Context in Aware Environments. In: *Proceedings of the Thirdteenth ACM Symposium on User Interface Software and Technology (UIST'00)*, 2000, S. 76–91

[42] DOSTAL, W. ; JECKLE, M. ; MELZER, I.: *Service-orientierte Architekturen mit Web Services. Konzepte - Standards - Praxis.* Spektrum Akademischer Verlag, 2005. – ISBN 3–82–741457–1

[43] DUSTDAR, Schahram ; TREIBER, Martin: A View Based Analysis on Web Service Registries. In: *International Journal on Distributed and Parallel Databases* 18 (2005), Nr. 2, S. 147–171

[44] EHRIG, Marc ; SURE, York: Ontology Mapping - An Integrated Approach. In: *Proceedings of the European Semantic Web Symposium (ESWS'04)*, 2004, S. 76–91

[45] ELLIS, C. A. ; GIBBS, J. ; REIN, G. L.: Groupware: Some Issues and Experiences. In: *ACM International Journal on Communications* 34 (1991), Nr. 1, S. 38–59

[46] FALLSIDE, D. C. ; WALMSLEY, P.: *XML Schema Primer*. – `http://www.w3.org/TR/xmlschema-0/`

[47] FENSEL, Dieter: *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce.* Springer, 2003. – ISBN 3–54000–302–9

[48] FINKENZELLER, K.: *RFID-Handbuch – Grundlagen und praktische Anwendungen induktiver Funkanlagen, Transponder und kontaktloser Chipkarten.* Hanser Verlag, 2002. – ISBN 3–446–22071–2

[49] FONTANA, R. J. ; GUNDERSON, S. J.: Ultra-Wideband Precision Asset Location System. In: *Proceedings of the IEEE Conference on Ultra Wideband Systems and Technologies*, 2002

[50] FOX, Dieter ; BURGARD, Wolfram ; THRUN, Sebastian: Markov Localization for Mobile Robots in Dynamic Environments. In: *Journal of Artificial Intelligence Research* 11 (1999), Nr. 1, S. 391–427

[51] FRANKLIN, D. ; FLACHSBART, J.: All Gadget and No Representation Makes Jack a Dull Environment / AAAI Spring Symposium on Intelligent Environments. 1998 (SS-98-02). – Forschungsbericht

[52] GALILEO: EUROPEAN SATELLITE NAVIGATION SYSTEM: *GALILEO.* – `http://ec.europa.eu/dgs/energy_transport/galileo/index_en.htm`

[53] GANNON, D. ; ALAMEDA, J. ; AL., O. C.: Building Grid Portal Applications From a Web Service Component Architecture. In: *International Journal Proceedings of the IEEE* 93 (2005), Nr. 3, S. 551–563

[54] GIAGLIS, G. M.: *Towards a Classification Framework for Mobile Location Services*. Idea Group Publishing, 2002, S. 67–85. – ISBN 1–591–40044–9

[55] GLONASS: GLOBAL NAVIGATION SATELLITE SYSTEM: *GLONASS*. – `http://www.glonass-center.ru/`

[56] GOMEZ-PEREZ, A.: *Ontological Engineering*. Springer, 2003. – ISBN 1–85233–551–3

[57] GPS: GLOBAL POSITIONING SYSTEM: *GPS*. – `http://www.navcen.uscg.gov/default.htm`

[58] GRAY, P. ; SALBER, D.: Modelling and Using Sensed Context Information in the Design of Interactive Applications. In: *Proceedings of the Eighth International Conference on Engineering for Human-Computer Interaction (EHCI'01)*, 2001, S. 317–325

[59] GRUBER, T.: Towards Priniciples for the Design of Ontologies Used for Knowledge Sharing. In: *International Journal of Human-Computer Studies* 43 (1995), Nr. 5, S. 907–928

[60] GRUDIN, J.: Computer-supported cooperative work: Its history and participation. In: *International IEEE Journal of Computers* 27 (1994), Nr. 5, S. 19–26

[61] GUARINO, N. ; GIARETTA, P.: Ontologies and Knowledge Bases: Towards a Terminological Clarification. In: *Towards Very Large Knowledge Base - Knowledge Building and Knowledge Sharing*, 1995, S. 25–32

[62] HAARSLEV, V. ; MÖLLER, R.: Description of the RACER System and its Applications. In: *Proceedings of the International Workshop on Description Logics (DL'01)*, 2001, S. 131–141

[63] HAASE, Peter ; HARMELEN, Frank van ; HUANG, Zhisheng ; STUCKENSCHMIDT, Heiner ; SURE, York: A Framework for Handling Inconsistency in Ontologies. In: *Proceedings of the Fourth International Semantic Web Conference (ISWC'05)*, 2005, S. 353–367

[64] HAGELWEIDE, Wilke: *SAM4OWL-S: Semi-Automatischer Matchmaker für OWL-S*, University of Paderborn, Diplomarbeit, 2006

[65] HAMMER, Katherine: Web Services and Enterprise Integration. In: *eAI Journal* 3 (2001), Nr. 1, S. 12–15

[66] HARTER, A. ; HOPPER, A. ; STEGGLES, P. ; WARD, A. ; WEBSTER, P.: The Anatomy a Context-aware Application. In: *Proceedings of the ACM/IEEE Conference on Mobile Computing and Networking (Mobicom'99)*, 1999, S. 59–68

[67] HARTMAN, B. ; FLINN, D. J. ; BEZNOSOV, K.: *Mastering Web Services Security*. Wiley, 2003. – ISBN 0–47–126716–3

[68] HAUSER, T. ; LÖWER, U. M.: *Web Services - Die Standards*. Galileo Computing, 2003. – ISBN 3–89–842393–X

[69] HAZAS, M. ; SCOTT, J. ; KRUMM, J.: Location-aware Computing Comes of Age. In: *IEEE International Magazine on Computer* 37 (2004), Nr. 2, S. 95–97

[70] HEFLIN, J.: *Web Ontology Language (OWL) Use Cases and Requirements*. – `http://www.w3.org/TR/webont-req/`

[71] HELD, A. ; BUCHHOLZ, S. ; SCHILL, A.: Modeling of Context Information for Pervasive Computing Applications. In: *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics (SCI'02)*, 2002, S. 97–104

[72] HENRICKSEN, Karen: Modeling Context Information in Pervasive Computer Systems. In: *Proceedings of the International Conference on Pervasive Computing (Pervasive'02)*, 2002, S. 97–104

[73] HIGHTOWER, J. ; BORRIELLO, G.: Location Systems for Ubiquitous Computing. In: *IEEE International Magazine on Computers* 34 (2001), Nr. 8, S. 57–66

[74] HILDEBRANDT, M.: *Sicherheitsrisiken und Sicherheitskonzepte von XML Web Services dargestellt an einem E-Business-Szenario*, Fachhochschule Bonn-Rhein-Sieg, Diplomarbeit, 2003

[75] HORROCKS, I. ; FENSEL, D. ; BROEKSTRA, J. ; DECKER, S. ; ERDMANNA, M. ; GOBLE, C. ; HARMELEN, F. v. ; KLEIN, M. ; STAAB, S. ; STUDER, R. ; MOTTA, E.: *Ontology Inference Layer (OIL) White Paper*. 2000. – `http://www.ontoknowledge.org/oil/downl/oil-whitepaper.pdf`

[76] HUANG, S.-X. ; FAN, Y.-S. ; ZHAO, D.-Z. ; MEI, C.-Y. ; ZHANG, L.: Webservice-based Enterprise Application Integration. In: *Computer Integrated Manufacturing Systems Journal* 9 (2003), Nr. 10, S. 864–867

[77] HULL, R. ; NEAVES, P. ; BEDFORD-ROBERTS, J.: Towards Situated Computing. In: *Proceedings of the first International Symposium on Wearable Computers*, 1997, S. 146–153

[78] IEEE 802.11 WORKING GROUP: *IEEE 802.11*. – `http://grouper.ieee.org/groups/802/11/`

[79] IMAMURA, T. ; DILLAWAY, B. ; SIMON, E.: *XML-Encryption Syntax and Processing*. – `http://www.w3.org/TR/xmlenc-core/`

[80] IRDA: INFRARED DATA ASSOCIATION: *IrDA*. – `http://www.irda.org`

[81] JAEGER, Michael C. ; ENGEL, Lars ; GEIHS, Kurt: A Methodology for Developing OWL-S Descriptions. In: *Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'05)*, 2005, S. 153–166

[82] JANACIK, Peter ; KAO, Odej ; RERRER, Ulf: An Approach Combining Routing and Resource Sharing in Wireless Ad Hoc Networks Using Swam-Intelligence. In: *Proceedings of the Seventh ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04)*, 2004, S. 31–40

[83] JANACIK, Peter ; KAO, Odej ; RERRER, Ulf: A Routing Approach Using Swam-Intelligence for Resource Sharing in Wireless Ad Hoc Networks. In: *Proceedings of the International Symposium on Trends in Communications (SYMPOTIC'04)*, 2005, S. 170–174

[84] JASPER, R. ; USCHOLD, M.: A Framework for Understanding and Classifying Ontology Applications. In: *Proceedings of the IJCAI'99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*, 1999, S. 1101–1115

[85] JOHNSTON, Eddie ; HESS, Andreas ; KUSHMERICK, Nicholas: ASSAM: A Tool for Semi-automatically Annotating Semantic Web Services. In: *Proceedings of the Third International Semantic Web Conferernce (ISWC'04)*, 2004, S. 320–334

[86] KAASINEN, Eija: User Needs for Location-aware Mobile Services. In: *Springer International Magazine on Personal and Ubiquitous Computing* 7 (2003), Nr. 1, S. 70–79

[87] KALYANPUR, Aditya ; PASTOR, Daniel J. ; BATTLE, Steve ; PADGET, Julian: Automatic mapping of OWL ontologies into Java. In: *Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'04)*, 2004, S. 151–157

[88] KAO, Odej ; RERRER, Ulf: Multimedia Services for Location-Aware, Ad-hoc Collaboration in Wireless Networks. In: *Proceedings of the International Conference on Wireless Networks (ICWN'04)*, 2004, S. 549–555

[89] KAO, Odej ; RERRER, Ulf: A Platform for Location-Aware, Ad-hoc Collaboration in Wireless Networks. In: *Proceedings of the International Workshop on Positioning, Navigation and Communication (WPNC'04)*, 2004, S. 171–178

[90] KAO, Odej ; RERRER, Ulf: *Peer-to-Peer Based Collaboration for Virtual Communities*. IDEA Group Inc., 2005, S. 27–39. – ISBN 1–59140–563–7

[91] KINDBERG, Tim: People, Places, Things: Web Presence for The Real World / HP Labs. 2000 (HPL-2000-16). – Forschungsbericht

[92] KLEIN, Michel ; NOY, Natalya F.: A Component-Based Framework For Ontology Evolution. In: *Proceedings of the International Workshop on Ontologies and Distributed Systems (IJCAI'03)*, 2003, S. 153–166

[93] KLUSCH, Matthias: *OWL-S Service Retrieval Test Collection Version 2*. – `http://projects.semwebcentral.org/projects/owls-tc/`

[94] KLUSCH, Matthias ; FRIES, Benedikt ; SYCARA, Katia: Automated Semantic Web Service Discovery with OWLS-MX. In: *Proceedings of the ACM International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'06)*, 2006, S. 226–235

[95] KNUBLAUCH, Holger ; FERGERSON, Ray W. ; NOY, Natalya F. ; MUSEN, Mark A.: The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In: *Proceedings of the Third International Semantic Web Conference (ISWC'04)*, 2004, S. 229–237

[96] KONTOGIANNIS, Kostas ; SMITH, Dennis ; O'BRIEN, Liam: On the Role of Services in Enterprise Application Integration. In: *Proceedings of the 10th International Workshop on Software Technology and Engineering Practice*, 2002, S. 103–111

[97] KRILL, Paul: *Overcoming Information overload*. 2000. – http://archive.infoworld.com/articles/ca/xml/00/01/10/000110caoverload.xml

[98] LEYMANN, F. ; ROLLER, D. ; SCHMIDT, M.-T.: Web Services and Business Process Management. In: *IBM Systems Journal* 41 (2002), Nr. 2, S. 198–211

[99] MANOLA, F. ; MILLER, E.: *Resource Description Framework (RDF) Specification*. – http://www.w3.org/TR/rdf-primer/

[100] MARTIN, D. ; BURSTEIN, M. ; HOBBS, J. ; LASSILA, O. ; MCDERMOTT, D. ; MCILRAITH, S. ; NARAYANAN, S. ; PAOLUCCI, M. ; PARSIA, B. ; PAYNE, T. ; SIRIN, E. ; SRINIVASAN, N. ; SYCARA, K.: *OWL-S: Semantic Markup for Web Services*. – http://www.w3.org/submissions/OWL-S

[101] MCCARTHY, J.: Notes on Formalizing Contexts. In: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1997, S. 555–560

[102] MCILRAITH, Sheila ; NARAYANAN, Srini ; PAOLUCCI, Massimo ; PARSIA, Bijan ; SIRIN, Evren ; SRINIVASAN, Naveen ; SYCARA, Katia: *DAML-S Draft Release*. – http://www.daml.org/services/daml-s/0.9/

[103] MCILRAITH, Sheila ; SON, Tran C.: Adapting Golog for Composition of Semantic Web Services. In: *Proceedings of the Eights International Conference on Principles of Knowledge Respresentation and Reasoning (KRR'02)*, 2002, S. 482–493

[104] MICROSOFT CORPORATION: *Microsoft Web Service Enhancements (WSE)*. – http://msdn.microsoft.com/webservices/webservices/building/wse/default.aspx

[105] MILLER, B. A. ; BISDIKIAN, C.: *Bluetooth Revealed - The Insider's Guide to an Open Specification for Global Wireless Communications*. Prentice Hall, 2001. – ISBN 0–13–090294–2

[106] MISHRA, P. ; LOCKHART, H.: *Security Assertion Markup Language (SAML)*. – http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[107] MITRA, N.: *SOAP Version 1.2 Primer*. – http://www.w3.org/TR/soap12-part0/

[108] MUSEN, Mark ; NOY, Natasha ; O'CONNOR, Martin ; REDMOND, Timothy ; RUBIN, Daniel ; TU, Samson ; TUDORACHE, Tania ; VENDETTI, Jennifer: *Protégé*. – http://protege.stanford.edu/

[109] NI, L. M. ; LIU, Y. ; LAU, Y. C. ; PATIL, A. P.: LANDMARC: Indoor Location Sensing Using Active RFID. In: *Proceedings of the IEEE Conference on Pervasive Computing and Communications (PerCom'03)*, 2003, S. 407–415

[110] NICKULL, D.: *Service-oriented Architecture Reference Model*. – http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

[111] NILES, Ian ; PEASE, Adam: Towards a Standard Upper Ontology. In: *Proceedings of the ACM International Conference on Formal Ontology in Information Systems (FOIS'01)*, 2001, S. 2–9

[112] OBJECT MANAGEMENT GROUP (OMG): *Common Object Request Broker Architecture (CORBA/IIOP)*. 2004. – `http://www.omg.org/technology/documents/corba_spec_catalog.htm`

[113] OPENCYC: *Opencyc Project*. – `http://www.opencyc.com`

[114] PAOLUCCI, Massimo ; KAWAMURA, Takahiro ; PAYNE, Terry R. ; SYCARA, Katia: Semantic Matching of Web Services Capabilities. In: *Proceedings of the First International Semantic Web Conference (ISWC'02)*, 2002, S. 333–341

[115] PAPAZAFEIROPOULOS, G. ; PRIGOURIS, N. ; MARIAS, G. F. ; HADJIEFTHYMIADES, S. ; MERAKOS, L.: Retrieving Position From Indoor WLANs Through GWLC. In: *Proceedings of IST Conference on Mobile and Wireless Communications*, 2003, S. 97–104

[116] PASCOE, J.: Adding Generic Contextual Capabilities to Wearable Computers. In: *Proceedings of the Second International Symposium on Wearable Computers*, 1998, S. 92–99

[117] PATIL, Abhijit A. ; OUNDHAKAR, Swapna A. ; SHETH, Amit P. ; VERMA, Kunal: Meteor-S Web Service Annotation Framework. In: *Proceedings of the Thirdteenth International Conference on the World Wide Web (WWW'04)*, 2004, S. 553–562

[118] PRIYANTHA, N. ; CHAKRABORTY, A. ; BALAKRISHNAN, H.: The Cricket Location Support System. In: *Proceedings of the ACM/IEEE Conference on Mobile Computing and Networking (Mobicom'00)*, 2000, S. 32–43

[119] PRIYANTHA, N. ; MIU, A. ; BALAKRISHNAN, H. ; TELLER, S.: The Cricket Compass for Context-aware Mobile Applications. In: *Proceedings of the ACM/IEEE Conference on Mobile Computing and Networking (Mobicom'01)*, 2001, S. 1–14

[120] RAJASEKARAN, Preeda ; MILLER, John ; VERMA, Kunal ; SHETH, Amit: Enhancing Web Services Description and Discovery to Facilitate Composition. In: *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'04)*, 2004, S. 55–68

[121] RANDALL, Julian ; AMFT, Oliver ; BOHN, Jürgen ; BURRI, Martin: LuxTrace - Indoor Positioning Using Building Illumination. In: *Personal and Ubiquitous Computing Journal* 1 (2006), Nr. 1, S. 391–427

[122] RANGANATHAN, Anand: A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. In: *Proceedings of the International Conference on Middleware (USENIX'02)*, 2002, S. 97–104

[123] RANGANATHAN, Anand ; AL-MUHTADI, Jalal ; CHETAN, Shiva ; CAMPBELL, Roy ; MICKUNAS, M. D.: MiddleWhere: A Middleware for Location Awareness in Ubiquitous Computing Applications. In: *Proceedings of the ACM/IFIP/USENIX International Conference on Middleware (Middleware'04)*, 2004, S. 397–401

[124] RECH, J.: *Wireless LANs - 802.11-WLAN-Technologie und praktische Umsetzung im Detail*. Heise Verlag, 2004. – ISBN 3–936931–04–6

[125] REED, J. H. ; KRIZMAN, K. J. ; WOERNER, B. D. ; RAPPAPORT, T. S.: An Overview of the Challenges and Progress in Meeting the E-911 Requirement for Location Service. In: *IEEE International Magazine on Communications* 36 (1998), S. 30–37

[126] REKIMOTO, J. ; AYATSUKA, Y.: CyberCode: Designing Augmented Reality Environments with Visual Tags. In: *Proceedings of the ACM Conference on Designing Augmented Reality Environments*, 2000, S. 1–10

[127] RERRER, Ulf: Location-aware Web Service Architecture using WLAN Positioning. In: *Proceedings of the International Workshop on Context-Aware Mobile Systems (CAMS'05)*, 2005, S. 237–245

[128] RERRER, Ulf: Location-awareness for a Service-oriented Architecture using WLAN Positioning. In: *Proceedings of the Third European Conference on Web Services (ECOWS'05)*, 2005, S. 106–112

[129] RERRER, Ulf ; KAO, Odej: Suitability of Positioning Techniques for Location-based Services in wireless LANs. In: *Proceedings of the Workshop on Positioning, Navigation and Communication (WPNC'05)*, 2005, S. 51–56

[130] ROBINSON, J. A. ; VORONKOV, Andrei: *Handbook of Automated Reasoning*. MIT Press, 2001. – ISBN 0–26218–223–8

[131] RODDEN, T. ; CHEVERST, K. ; DAVIS, K. ; DIX, A.: Exploiting Context in HCI Design for Mobile Systems. In: *Proceedings of the Workshop on Human Computer Interaction with Mobile Devices*, 1998, S. 97–104

[132] ROOS, T. ; MYLLYMAEKI, P. ; TIRRI, H. ; MISIKANGAS, P. ; SIEVAENEN, J.: A Probabilistic Approach to WLAN User Location Estimation. In: *International Journal of Wireless Information Networks* 9 (2002), Nr. 3, S. 155–164

[133] ROTH, J.: *Mobile Computing - Grundlagen, Technik, Konzepte*. dpunkt Verlag, 2002. – ISBN 3–89864–165–1

[134] ROWSTRON, Antony ; DRUSCHEL, Peter: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: *Lecture Notes in Computer Science* 2218 (2001), Nr. 1, S. 329+

[135] RYAN, N. S. ; PASCOE, J. ; MORSE, D. R.: Enhanced Reality Fieldwork: The Context-aware Archaeological Assistant. In: *Proceedings of the Comference of Computer Applications in Archaeology*, 1997, S. 97–104

[136] SALBER, D. ; DEY, A. K. ; ABOWD, G. D.: Ubiquitous Computing: Defining an HCI Research Agenda for an Emerging Interaction Paradigm / Georgia Institue of Technology. 1998 (GIT-GVU-98-01). – Forschungsbericht

[137] SCHILIT, B. ; ADAMS, N. ; WANT, R.: Disseminating Active Map Information to Mobile Hosts. In: *IEEE International Journal on Networks* 8 (1994), Nr. 5, S. 22–32

[138] SCHILIT, B. ; THEIMER, M.: Context-Aware Computing Applications. In: *Proceedings of the First International Workshop on Mobile Computing Systems and Applications*, 1994, S. 85–90

[139] SCHILLER, J.: *Mobilkommunikation*. Pearson Studium, 2003. – ISBN 3–8273–7060–4

[140] SCHILLER, J. ; VOISARD, A.: *Location-based Services*. Morgan Kaufmann, Elsevier, 2004. – ISBN 1–55860–929–6

[141] SCHMIDT, A. ; LADERHOVEN, K. V.: There is More to Context Than Location. In: *International Journal on Computers and Graphics* 23 (1999), Nr. 6, S. 893–901

[142] SHESHAGIRI, Mithun: *Automatic Composition and Invocation of Semantic Web Services*, University of Maryland, Diplomarbeit, 2004

[143] SIRIN, Evren: *OWL-S API Java Documentation*. – `http://www.mindswap.org/2004/owl-s/api/doc/javadoc/`

[144] SIRIN, Evren ; PARSIA, Bijan: Planning for Semantic Web Services. In: *Proceedings of the Third International Semantic Web Conference (ISWC'04)*, 2004, S. 237–246

[145] SRIVASTAVA, Biplav ; KOEHLER, Jana: Web Service Composition-Current Solutions and Open Problems. In: *Proceedings of the International Workshop on Planning for Web Services (ICAPS'03)*, 2003, S. 28–35

[146] STEIN, L. A. ; CONNOLLY, D. ; MCGUINNESS, D.: *DAML-ONT Intitial Release*. – `http://www.daml.org/2000/10/daml-ont.html`

[147] STOICA, Ion ; MORRIS, Robert ; KARGER, David ; KAASHOEK, Frans ; BALAKRISHNAN, Hari: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: *Proceedings of the ACM Special Interest Group on Data Communications Conference (SIGCOMM'01)*, 2001, S. 149–160

[148] STRANG, T. ; LINNHOFF-POPIEN, C.: A Context Modeling Survey. In: *Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management*, 2004, S. 97–104

[149] STRASSMANN, M. ; COLLIER, C.: *Case Study: Development of the Find Driend Application*. Morgan Kaufmann, Elsevier, 2004, S. 27–39. – ISBN 1–55860–929–6

[150] STUCKENSCHMIDT, H. ; HARMELEN, F. v.: *Information Sharing on the Semantic Web*. Springer, 2004. – ISBN 3–54–020594–2

[151] SUN: *Sun Java Web Service Tutorial*. – `http://java.sun.com/webservices/tutorial.html`

[152] SYCARA, Katia ; WIDOFF, Seth ; KLUSCH, Matthias ; LU, Jianguo: Larks: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. In: *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAAS'02)*, 2002, S. 173–203

[153] TREVISANI, Emiliano ; VITALETTI, Andrea: Cell-ID Location Technique, Limits and Benefits: An Experimental Study. In: *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*, 2004, S. 51–60

[154] UDDI.ORG: *UDDI Technical White Paper*. 2000. – `http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf`

[155] UNIVERSITY OF MANCHESTER: *Bio Ontologies Community*. – `http://bio-ontologies.man.ac.uk/`

[156] UNIVERSITY OF MARYLAND, BALTIMORE COUNTY: *Swoogle: The Semantic Web Search Engine*. – `http://swoogle.umbc.edu`

[157] VONHOEGEN, H.: *Einstieg in XML*. Galileo Computing, 2004. – ISBN 3–89–842–488–X

[158] WANG, X. H. ; ZHANG, D. Q. ; GU, T. ; PUNG, H. K.: Ontology-based Context Modeling and Reasoning Using OWL. In: *Proceedings of the Second IEEE Conference on Pervasive Computing and Communications (PerComm'04)*, 2004, S. 18–22

[159] WANT, R. ; HOPPER, A. ; FALCAO, V. ; GIBBONS, J.: The Active Badge Location System. In: *ACM Transactions International Journal on Information Systems* 10 (1992), Nr. 1, S. 91–102

[160] WARD, A. ; JONES, A. ; HOPPER, A.: A New Location Technique for the Active Office. In: *IEEE International Journal on Personal Communications* 4 (1997), Nr. 5, S. 42–47

[161] WILSON, P.: *Computer-supported cooperative work: An Introduction*. Kluwer Academic Publishers, 1991. – ISBN 0–79–231–446–8

[162] YOUSSEF, M. A. ; AGRAWALA, A. ; SHANKAR, A. U.: WLAN Location Determination via Clustering and Probability Distributions / University of Maryland. 2002 (CS-TR-4350 and UMIACS-TR-20002-30). – Forschungsbericht

[163] YOUSSEF, Moustafa ; YOUSSEF, Adel ; RIEGER, Chuck ; SHANKAR, Udaya ; AGRAWALA, Ashok: PinPoint: An Asynchronous Time-Based Location Determination System. In: *Proceedings of the Fourth International Conference on Mobile Systems, Applications, and Services (MobiSys'06)*, 2006, S. 97–104

[164] ZHAO, Yilin: Standardization of Mobile Phone Positioning for 3G Systems. In: *IEEE International Magazine on Communications* 40 (2002), Nr. 7, S. 108–116

[165] ZHU, F. ; ZHU, W. ; MUTKA, M. W. ; NI, L.: Expose or Not? A Progressive Exposure Approach for Service Discovery in Pervasive Computing Environments. In: *Proceedings of the Third International IEEE Conference on Pervasive Computing and Communications (PerCom'05)*, 2005, S. 225–234

[166] ZHU, Feng ; MUTKA, Matt ; NI, Lionel: Classification of Service Discovery in Pervasive Computing Environments / Michigan State University, East Lansing. 2002 (MSU-CSE-02-24). – Forschungsbericht