Giving Meaning to GI Web Service Descriptions

Florian Probst and Michael Lutz

Institute for Geoinformatics, Robert Koch Str. 26-28,
48149 Münster, Germany

Abstract. Composition of web services based on currently available descriptions such as WSDL are error-prone because the meaning (or semantics) of the labels used in these syntactic descriptions is unclear. We identify three types of problems that can result from semantically heterogeneous descriptions during service composition. These problems call for a *Semantic Reference System* for semantically annotating symbols, i.e. referencing symbols to concepts and semantically grounding these concepts. We present a three-level architecture for such a Semantic Reference System and illustrate how it can be used for solving the problems identified. Our example for illustration purposes is taken from the domain of disaster management and focuses on the composition of geographic information services.

1 Introduction

Composability is often seen as one of the main strengths of web services. In the geospatial domain, after focusing mainly on services providing data and maps, the first (geo) processing services are currently emerging.

To enable meaningful composability of web services not only syntactic descriptions such as WSDL [1] are needed. A further step has to be taken in providing the user with descriptions telling him what the labels of data types and operations used in a syntactic service description actually *mean*. This calls for a Semantic Reference System [2] allowing for semantic annotation of symbols, referencing of symbols to concepts and semantic grounding of concepts with image schemata (Fig. 1).

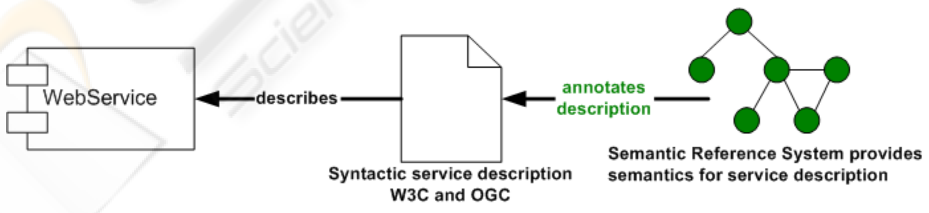

WebService
describes
annotates description
Syntactic service description W3C and OGC
Semantic Reference System provides semantics for service description

Fig. 1. A Semantic Reference System provides (grounded) semantics for standard service descriptions

In this paper we introduce the architecture for a three-level Semantic Reference System consisting of application ontology, domain ontology and semantic grounding levels. We focus on data types that are used as operation input and output, rather than on the functionality of the operation. We illustrate the conceptual results with a real world example. We show that, by evaluating references of service descriptions to a Semantic Reference System, meaningful interoperability between services can be ensured during service discovery.

The following short scenario[1] is used to explain in which context we encountered and solved semantic problems: A service provider is about to build a composite service for the management of accidents involving toxic gas releases from a chemical plant. He builds the composite service starting with the most specialized service[2] and subsequently adding further services until the composite service meets the user's requirements. The user already has the central, most specialized service of the composite service available. This is a service for calculating a toxic plume (subsequently called *CalculateGasDispersionService*). The first step is to check the input and output of the plume service. It requires information about the wind speed, the wind direction, the location of the gas emission and the emission rate as input. The output is a polygon indicating the dispersion of the gas. The user chooses as next step to find an additional service providing information on the wind direction and therefore searches for services providing weather information in a UDDI registry [3]. He discovers two services: the *GlobalWeatherService* and the *AirportWeatherService* provided by CapeScience (*http://www.capescience.com*). The user now has to determine whether these services actually match (syntactically and semantically) the requirements of the *CalculateGasDispersionService*. The semantic problems he encounters are explained in the following chapter.

2 Types of Semantic Problems

Currently service discovery relies on the labels of input and output descriptions and on the labels of data types given in WSDL (or similar service metadata) descriptions. It is generally assumed that if the labels are the same, the transported information is, too. However, this is not always the case. Different types of semantic heterogeneity have been identified for GIS [4] and GI web services in general [5]. In the following, we present three types of heterogeneity problems that play a role during GI web service composition. Each problem type is illustrated by relating it to the scenario given above (Fig. 2).

1 This scenario was developed in conjunction with the ACE-GIS e-Emergency composite service (*http://www.acegis.net/*).

2 To keep things simple, we assume that each web service has only one operation. We therefore use the terms web service and operation interchangeably.

2.1 Problem Type I (Naming Heterogeneity)

The output of a web service and the input of a second web service are represented with the same data type and refer to the same domain concept, but have different *labels (names).*

The *AirportWeatherService* and the *GlobalWeatherService* both provide information about the wind direction. However, the labels of the data types containing the required information are different. The *GlobalWeatherService* refers to the information as *prevailing_direction* while the *AirportWeatherService* labels the information as *wind*. However, both elements represent the same (domain) concept of *wind direction.*

This problem can be solved by annotating the elements used in the WSDL descriptions with concepts from an application ontology. The application concepts are always referenced to (or derived from) more general domain concepts. The reference includes additional restrictions on the properties of the domain concept, thus constraining the meaning of the application concept. If two application concepts refer to the same domain concept and their restrictions are identical, the meaning of the annotated symbols is the same.

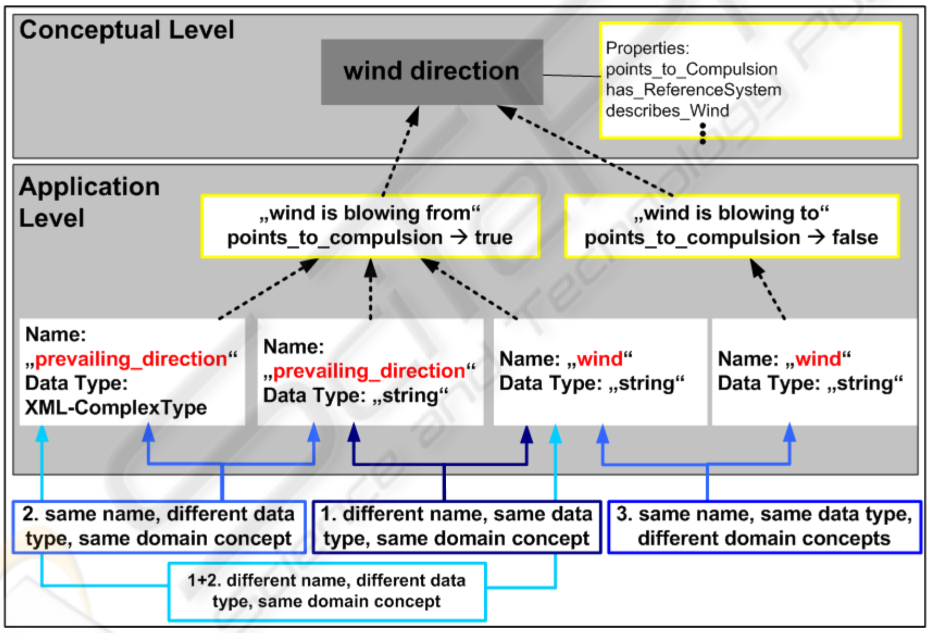

Conceptual Level
wind direction
Properties:
points_to_Compulsion
has_ReferenceSystem
describes_Wind
Application Level
„wind is blowing from“
points_to_compulsion → true
„wind is blowing to“
points_to_compulsion → false
Name: „prevailing_direction“
Data Type: XML-ComplexType
Name: „prevailing_direction“
Data Type: „string“
Name: „wind“
Data Type: „string“
Name: „wind“
Data Type: „string“
2. same name, different data type, same domain concept
1. different name, same data type, same domain concept
3. same name, same data type, different domain concepts
1+2. different name, different data type, same domain concept

Fig. 2. Types of semantic problems. Restrictions on a domain concept change its meaning

2.2 Problem Type II (Data Type Heterogeneity)

The output of a web service and the input of a second web service have the same labels (names) and refer to the same domain concept, but are represented with different *data types.*

The example given above is further complicated by a second source of heterogeneity. The *GlobalWeatherService* provides the wind direction information represented as a complex type labeled *Direction.* The *AirportWeatherService* provides this information contained in a *String*. However, both elements represent the same domain concept[3].

This problem is not simply to be considered syntactical heterogeneity since the meaning of the information contained in a complex type is not explicit to the user. When dealing with complex data types, a semantic description of their structure and content can help in supplying rules for transforming them or for extracting the required information. How to semantically describe complex types such that these rules can be generated automatically from the descriptions will be a topic for future research. In our current approach we assume that appropriate parsers are available for transforming the information into the required data type of the preceding service.

2.3 Problem Type III (Conceptual Heterogeneity)

The output of a web service and the input of a second web service have the same labels (names), are represented with the same data type, but refer to different *domain concepts.*

Consider now the wind information represented as a *String* provided by the *AirportWeatherService* as described above. And consider further that the *CalculateGasDispersionService* also requires wind information represented as a *String.* However, the *CalculateGasDispersionService* interprets the provided *String* not as degrees characterizing the direction the wind is blowing from. Instead it interprets the *String* as characterizing the direction the wind is blowing to. This misinterpretation introduces a 180-degree mismatch.

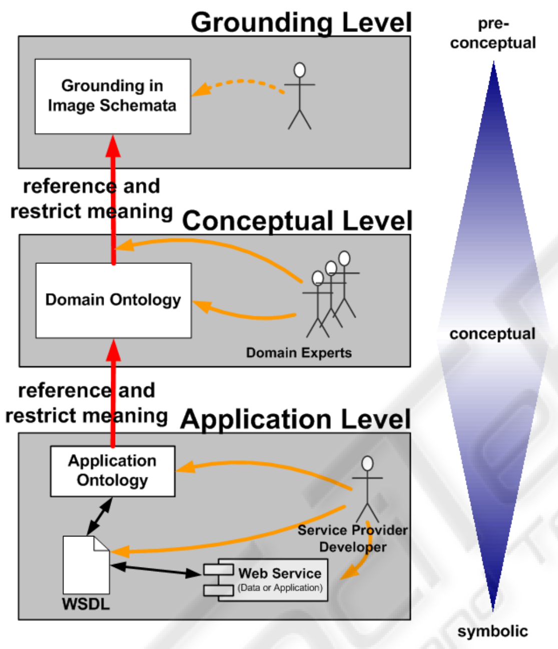
This can lead to the creation of composite services that produce results not intended by the user. This is due to the fact that currently most composite services are built manually using WSDL-based service descriptions. If inputs and outputs of operations have the same name and data type, the user cannot tell that these operations refer to different domain concepts. The underlying assumption causing this problem is that if something is described identically, it must have the same meaning.

By referencing the application level concepts to different domain concepts, or by using different restrictions on the same domain concepts, it can be prevented that services whose inputs and outputs do not match (on the conceptual level) are combined in a service chain.

[3] Note that in the example which is taken from existing web services, problem types I and II occur simultaneously. For clarification, the example problem is split into two problems types.

3 Semantic Reference System

This chapter gives a brief introduction of the three levels of the Semantic Reference System and how they are related to each other. The application and domain levels consist of ontologies, i.e. “explicit specifications of a conceptualization” [6]. The necessity of introducing three levels is explained in the following section, which relates the user’s tasks during service composition to the semantic problems identified in the previous section. For better comprehension we start with describing the middle section on the architecture, the Conceptual Level. Fig. 3 gives an overview of the three-leveled architecture of a Semantic Reference System.


Grounding Level
Grounding in Image Schemata
reference and restrict meaning
Conceptual Level
Domain Ontology
Domain Experts
reference and restrict meaning
Application Level
Application Ontology
Service Provider Developer
Web Service (Data or Application)
WSDL
pre-conceptual
conceptual
symbolic

GOAL	Number of Ontologies	Changes	Ontology Engineering Approach
Semantic datum	one (?)	none	Image schamata based
Stable meaning	few	rare	Blending based domain theory
dynamic symbols	many	frequent	Axiomatic model

Fig. 3. Three-level Semantic Reference System; left: the levels meet different requirements

3.1 Conceptual Level

The Conceptual Level is the level of human concepts. A domain ontology provides an agreed-upon conceptualization of a certain part of the world; in our example a small part of the domain of meteorology. Such a domain ontology will never be complete since the meteorological knowledge evolves and the terms used change. But it will serve to describe a particular worldview on this domain and the vocabulary humans use to communicate about it.

The domain ontology makes a world view explicit and presents it in a machine- interpretable way. It is important to note, that the task of connecting a symbol with a concept in the human mind, in other words assigning meaning to a symbol, remains partly with the user. However, the important contribution of a domain ontology is that the many possible conceptualizations an English speaking user could assign to symbols such as *wind* are restricted via a set of relations to other symbols and by classifying the symbols in a subsumption hierarchy. In this way, the domain ontology restricts the meaning of a known symbol (term).

Thus, while the domain ontology assigns meaning to a symbol by providing a conceptualization for it, this conceptualization is based on the use of other symbols. The meaning of these symbols is either given by yet other symbols or by assuming that the user *knows* the meaning of these symbols. The latter may work within a small user community, but for larger communities this assumption will not hold.

3.2 Semantic Grounding Level

To escape the vicious circle of defining concepts with other concepts on an ever higher level of abstraction or by assuming the user knows the meaning of symbols a *Semantic Grounding* Level is required. Semantic grounding is the process of referencing a concept described on the conceptual level to concepts form which is assumed that their meaning is common to the user and thus needs no further definition.

We propose image schemata to semantically ground the concepts used on the Domain Ontology Level. Image schemas fall between abstract propositional structures (such as predicates) and concrete images (such as the spatial mental images in [7]). They are developed through bodily experiences and influence our reasoning through the recurrence of form and function. They stand in a long tradition of representing elements of knowledge into patterns or *schemas*. An image schema can be seen as a generic and abstract structure that helps people establish a connection between different experiences that have this same recurring structure. Therefore, meaning involves image-schematic structures [8, 9].

After defining the meaning of a symbol via the restrictions posed on it on the Conceptual Level, references to image schemata which are represented using similar methods as employed for the domain ontologies will further reduce semantic ambiguity. These references will be based on non-taxonomic relations, avoiding the subsumption of domain concepts under image schemata. We intend to restrict the meaning of domain concepts by using the shared and thus common understanding of image schemata. Image schemata are seen as *semantic datum* in a *Semantic Reference System*, in analogy to the datum of spatial reference systems.

3.3 Application Level

On the application level, meaning is assigned to web service descriptions, i.e. the symbols (labels) used in the WSDL-based service description are captured in an application ontology. The method to build an application ontology is similar of such for building domain ontologies. However, in the domain ontology the meaning for the *general* vocabulary of a certain domain is captured. The application ontology now makes use of the symbols to which meaning is already assigned and connects these to the semantics-free symbols of the WSDL descriptions.

The reason for introducing an application level is that the domain ontology may define concepts in a way which is to general for a certain application. Therefore the application level provides a possibility for further restricting the meaning of domain ontology concepts. It is also possible to introduce concepts not contained in the domain ontology by using non-taxonomic relations to domain ontology concepts. This is why application ontologies should not be understood as specializations of domain ontologies as could be inferred from [10; figure 4].

Since meaning is passed from the Groundling Level via the Conceptual Level to the Application Level consistency issues are to be considered on the Grounding and Conceptual Level. We argue that domain and grounding ontology provider are responsible for keeping updates of their ontologies consistent with prior versions. Extending domain or grounding ontologies with new concepts will cause no problems on application level. However extending existing concepts with new restrictions or changing existing restrictions can cause consistency problems on the application level. Since domain and grounding ontologies are considered to be relatively stable constructs with only few changes after an initial development phase, comparable to agreed upon meta data standards. Therefore we consider the three-leveled architecture as good solution for guaranteeing consistency and at the same time allowing for evolving semantics over time.

4 Semantic Problem Types Related to User Tasks during Service Composition

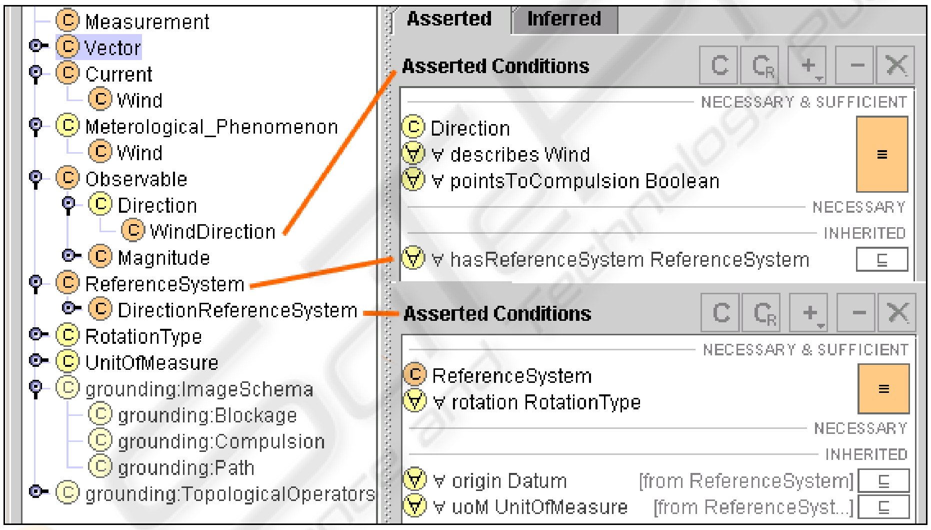
The types of semantic problems identified in the previous section will now be related to the user's tasks during service composition in order to explain the need of a three-level semantic reference system. Each of the levels and the references between them will be illustrated by providing formal definitions of concepts for the example scenario. The ontologies containing these definitions can be found at *http://musil.uni-muenster.de/onto/*.

For enhanced readability these concepts are imported from the different ontologies (e.g. *grounding.owl* or *domain1.owl*) into one ontology. The prefixes of the concepts (e.g. *grounding:*) shown indicate from which ontology they are taken (Fig. 4 and 5). The prefixes, e.g. *domain1* can be extended to the full XML namespace.

The ontology language employed is the Web Ontology Language OWL [11]. To build the ontologies we employed the ontology editor Protégé 2.0 [12] in combination with an plug-in supporting OWL [13].

Defining the concept of “wind direction”. As starting point of service discovery, the user needs the possibility to specify his concept of wind direction. This is performed by querying a domain ontology (for meteorology). In the example, the user will find the concept *WindDirection* with all properties relevant in the domain of meteorology (Fig. 4). The property *hasReferenceSystem* relates *WindDirection* to the concept *ReferenceSystem*. Its sub-concept *DirectionReferenceSystem* in turn has the properties *rotation*, *origin*, and *unit of measure*. Via the concept *DirectionReferenceSystem* the user could, for example, specify that the concept of *WindDirection* he is looking for takes north as origin, turns clockwise and uses *degrees* as a unit of measure.

To further decrease the ambiguity of domain ontology concepts, properties using or based on image schemata are employed. For example, *compulsion* is identified as an image schema by [8]. The domain ontology property *pointsToCompulsion* is therefore referenced to the grounding level. The possibility of referencing domain ontology concepts to image schemata decreases the likelihood of ambiguously interpreted domain ontology concepts. The (yet incomplete) set of image schemata, their internal structure and how they can serve as semantic grounding level need further investigation. However, the possibility to break the vicious circle of defining concepts with other (undefined) concepts is appealing.



```
C Measurement
C Vector
C Current
    C Wind
C Meterological_Phenomenon
    C Wind
C Observable
    C Direction
        C WindDirection
    C Magnitude
C ReferenceSystem
    C DirectionReferenceSystem
C RotationType
C UnitOfMeasure
C grounding:ImageSchema
    C grounding:Blockage
    C grounding:Compulsion
    C grounding:Path
C grounding:TopologicalOperators
```

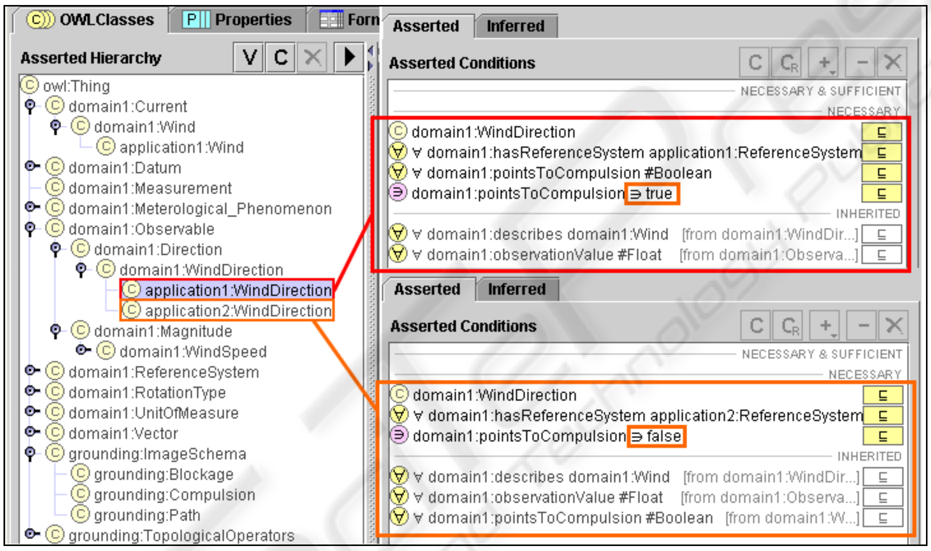
```
Asserted   Inferred

Asserted Conditions
                                  NECESSARY & SUFFICIENT
C Direction                                             ≡
∀ describes Wind
∀ pointsToCompulsion Boolean
                                               NECESSARY
                                               INHERITED
∀ hasReferenceSystem ReferenceSystem                    ⊑

Asserted Conditions
                                  NECESSARY & SUFFICIENT
C ReferenceSystem                                       ≡
∀ rotation RotationType
                                               NECESSARY
                                               INHERITED
∀ origin Datum          [from ReferenceSystem]          ⊑
∀ uoM UnitOfMeasure     [from ReferenceSyst...]         ⊑
```

Fig. 4. Definition of the domain ontology concept *wind direction* in the ontology editor Protégé using the OWL plug-in (modified screenshot).

Finding services dealing with the identified concept. With the chosen concept “wind direction” the user discovers all three services in an application ontology registry since the labels used in their WSDL service descriptions refer to application ontology concepts. The WSDL labels wind (from AirportWeatherService) and prevailing_direction (from GlobalWeatherService) both refer to the application ontology concept *application1:WindDirection*. The WSDL label wind (from *CalculateGasDispersionService*) refers to *application2:WindDirection*.

Although both application ontology concepts are defined using the same domain ontology concept, the restrictions put on their references reveal that the two concepts are different. They represent the direction *in which* respectively *from which* the wind is blowing. Discovering whether two concept definitions are equivalent or similar and whether concepts are related in a subsumption hierarchy is easy in the simple example we use for illustration. However, when using more complex definitions, it becomes much more difficult. In such cases a reasoning engine such as RACER [14] can be used to compute a subsumption hierarchy and to identify equivalent concepts.

With the currently existing WSDL descriptions the user has to judge whether the application concepts differ in such a way that chaining the services will produce wrong results. In this example, chaining the *AirportWeatherService* or the *Global-WeatherService* to the *CalculateGasDispersionService* both would result in a 180-degree mismatch.



owl:Thing
domain1:Current
domain1:Wind
application1:Wind
domain1:Datum
domain1:Measurement
domain1:Meterological_Phenomenon
domain1:Observable
domain1:Direction
domain1:WindDirection
application1:WindDirection
application2:WindDirection
domain1:Magnitude
domain1:WindSpeed
domain1:ReferenceSystem
domain1:RotationType
domain1:UnitOfMeasure
domain1:Vector
grounding:ImageSchema
grounding:Blockage
grounding:Compulsion
grounding:Path
grounding:TopologicalOperators

domain1:WindDirection ⊑
∀ domain1:hasReferenceSystem application1:ReferenceSystem ⊑
∀ domain1:pointsToCompulsion #Boolean ⊑
domain1:pointsToCompulsion ∋ true ⊑
∀ domain1:describes domain1:Wind [from domain1:WindDir...] ⊑
∀ domain1:observationValue #Float [from domain1:Observa...] ⊑

domain1:WindDirection ⊑
∀ domain1:hasReferenceSystem application2:ReferenceSystem ⊑
domain1:pointsToCompulsion ∋ false ⊑
∀ domain1:describes domain1:Wind [from domain1:WindDir...] ⊑
∀ domain1:observationValue #Float [from domain1:Observa...] ⊑
∀ domain1:pointsToCompulsion #Boolean [from domain1:W...] ⊑

Fig. 5. Definition of two application ontology concepts representing two different conceptualization of *wind direction*

The domain ontology concept plus the restrictions on its interpretation provide *meaning* to the labels used in the WSDL service descriptions. Here it becomes obvious why domain ontology and application ontology need to form separate levels. The domain ontology provides a stable source of broadly defined (domain) concepts. The application ontology provides the service developer with a flexible means to restrict the meaning of the domain ontology concepts. Finding services using the same restrictions on domain concepts solves naming problems (problem type I). The possibility for the user to be aware of different restrictions solves conceptual problems (problem type III).

Alternative Search. Consider a user which belongs to a user community different form that which built the domain ontology. It is likely that such a user has difficulties in “finding the right words” to start a search on a domain ontology which is based on unfamiliar vocabulary. Instead he specifies a query “find concepts which describes wind”, by using the *describes* property and the *wind* concept from the domain ontology. This approach follows the user-defined query concept introduced by [15]. The reasoning engine RACER returns application ontology concepts which implement such property. Our ontology architecture application ontologies contain references to the services they describe. Finding an application ontology concept which is a sub-concept of the user-defined query immediately indicates a web service potentially interesting to the user. The two application ontologies in our example can be distinguished by a user-defined query which is asking for a service which understands a wind direction as pointing to the compulsion the wind creates. This is achieved by searching for concepts implementing the *pointsToCompulsion* property with the value *true*.

Learn how the service represents the needed information. Consider the user searches for services dealing with wind direction where the property *domain1:pointsToCompulsion* is *true*. Such a query would result in finding the *AirportWeather-Service* and the *GlobalWeatherService* which reference to the application ontology called *application1*. Now the user needs to know which data types are used to represent the information he requires. For this purpose the application ontology of the services is queried. The resulting human-readable description of the data types employed in the service can be used to deal with problems of type II. How to automatically derive rules for transforming complex types or for extracting the required information *automatically* from the descriptions will be a topic for future research.

5 Related Work

With the maturation of service-oriented computing, several approaches to service discovery that are based on service descriptions referring to ontologies have been proposed (e.g. [16-18]). We share with these approaches the use of inference engines for evaluating subsumption relationships between the ontology concepts that are used for annotating syntactic service descriptions. However, we differ from these approaches in the ontology architecture applied.

In [19] a classification of different ontology architectures used for making the semantics of information sources explicit is introduced (Fig. 6).

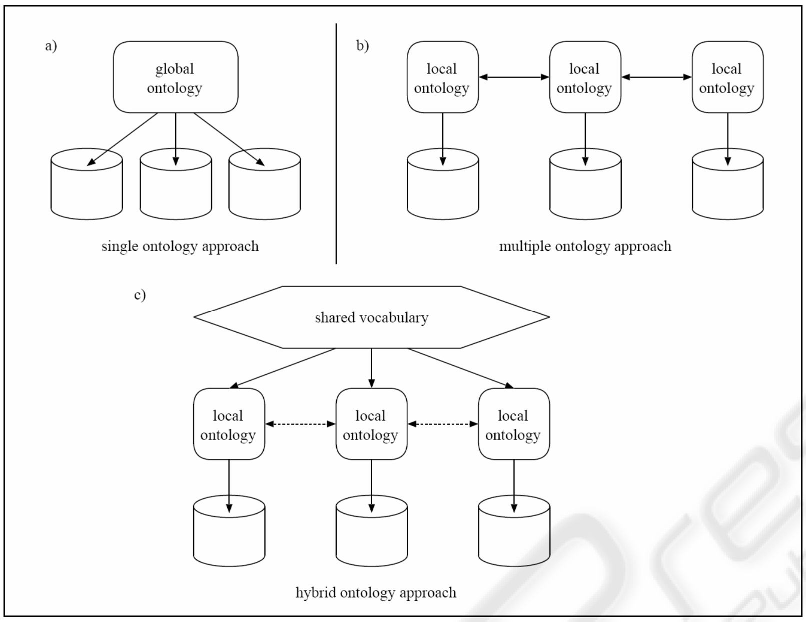


Fig. 6. Different ontology architectures for information integration [19]

( ) *Single Ontology approaches* use one global ontology providing a shared vocabulary for the specification of the semantics. All information sources are related to the global ontology. Such approaches can be applied to problems where all information sources to be integrated provide a very similar view on a domain.

( ) In *multiple ontology approaches*, each information source is described by its own ontology. In principle, each of these application ontologies can be a combination of several other ontologies. However, it can not be assumed that several application ontologies share the same vocabulary. While each application ontology can be developed independently, the lack of a common vocabulary makes it difficult to compare different application ontologies.

( ) *Hybrid approaches* are similar to multiple ontology approaches in that the semantics of each source is described by its own ontology. But in order to make the local ontologies comparable to each other they are built from a global shared vocabulary. The shared vocabulary contains basic terms (the primitives) of a domain which are combined in the local ontologies in order to describe more complex semantics. Sometimes the shared vocabulary is also an ontology.

The approaches currently proposed for semantic service discovery employ simple ontology architectures that can be classified as single ontology approaches. The use of ontologies on multiple levels is not intended. In contrast, the ontology architecture in this paper can be seen refinement of the hybrid ontology approach. Our domain ontology corresponds to a shared vocabulary as shown in Fig. 6c. In addition, the image-schemata-based ontology on the grounding level provides a means for comparing concepts form different domains on the one hand, and for grounding domain concepts in human cognition on the other hand.

6 Conclusion and Future Work

We have identified three types of semantic problems that may occur during web service composition: Naming heterogeneity, conceptual heterogeneity and data type heterogeneity. We have illustrated these types by relating them to a real world example based on services employed in a composite service for the management of accidents involving toxic gas releases from a chemical plant. Meaningful composability of web services needs semantic interoperability between web services. We argue that semantic interoperability needs a sound theory of semantic grounding.

Therefore we have introduced a three-level Semantic Reference System for tackling the identified problem types. It allows the user to

- discover appropriate services, even when they are named differently than expected,
- identify data type heterogeneity and take further syntactical integration steps, and
- Discover conceptual heterogeneity problems and thus avoid the construction of composite services producing unintended results.

We have shown in an example ontology how to give meaning to the symbols (labels of data types) used in WSDL descriptions. This is done by referencing the symbols to an application ontology, which in turn derives meaning for its concepts from a domain ontology, which in turn grounds its concepts on an image schemata-based grounding level.

We argue that the introduced Semantic Reference System allows for the different degrees of flexibility, unambiguity and stability such a system has to provide. Apart from the grounding of meaning with image schemata on the grounding level, the innovation of this approach lies in the flexibility of the application ontology level. On this level, a service developer can model virtually anything (including totally fictive worlds) using restrictions on domain ontology concepts. Likewise, a user can find a certain concept based on a domain ontology vocabulary and learn how its meaning is restricted on the application ontology level. While this avoids that statements that are valid in a certain application ontology need to be valid in another one, they can still be traced back to their common domain ontology concept.

Future work includes the exploration of the reliability and usability of image schemata for grounding the semantics of domain ontology concepts. Also, we aim to establish a working prototype of the described Semantic Reference System. This involves the implementation of an environment for testing the semantics of inputs and outputs of composite services.

Acknowledgements

The work presented in this paper has been supported by the European Commission through the ACE-GIS project (grant number IST-2002-37724) and the German Federal Ministry for Education and Research as part of the GEOTECHNOLOGIEN program (grant number 03F0369A). It can be referenced as publication no. GEOTECH-52.

References

1. Chinnici, R., Moreau, J.-J., Gudgin, M., Schlimmer, J., Weerawarana, S. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.[Online]. Available: http://www.w3.org/TR/2003/WD-wsdl20-20031110/
2. Kuhn, W.: Semantic Reference Systems. International Journal of Geographical Information Science (2003)
3. Bellwood, T., Clément, L., von Riegen, C. (2003) UDDI Version 3.0.1.[Online]. Available: http://uddi.org/pubs/uddi-v3.0.1-20031014.htm
4. Bishr, Y.: Overcoming the semantic and other barriers to GIS interoperability. International Journal of Geographical Information Science 12 (1998) 299-314
5. Lutz, M., Riedemann, C., Probst, F.: A Classification Framework for Approaches to Achieving Semantic Interoperability. In: S. Timpf, (ed.) COSIT 2003 (Conference on Spatial Information Theory). Springer (2003) 200-217
6. Gruber, T.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5 (1993) 199-220
7. Barkowsky, T.: Mental Processing of Geographic Knowledge. In: D. Montello, (ed.) Spatial Information Theory - Foundations of Geographic Information Science, Proceedings of COSIT 2001, Morro Bay, CA, USA, September 2001. Springer (2001) 371-386
8. Johnson, M.: The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason. The University of Chicago Press, Chicago (1987)
9. Gärdenfors, P.: Conceptual Spaces - The Geometry of Thought. Bradford Books, MIT Press, Cambridge, MA (2000)
10. Guarino, N.: Formal Ontology and Information Systems. In: Proc. Formal Ontology in Information Systems (1998) 3-15
11. McGuinness, D. L., Van Harmelen, F. (2004) OWL Web Ontology Language, W3C Candidate Recommendation.[Online]. Available: http://www.w3.org/TR/2003/CR-owl-features-20030818/
12. Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R. W., Musen, M. A.: Creating Semantic Web Contents with Protege-2000. IEEE Intelligent Systems 16 (2001) 60-70
13. Knublauch, H. (2004) Editing OWL Ontologies with Protégé.[Online]. Available: http://protege.stanford.edu/plugins/owl/tutorial/index.html
14. Haarslev, V., Möller, R. (2003) RACER User's Guide and Reference Manual Version 1.7.7.[Online]. Available: http://www.sts.tu-harburg.de/~r.f.moeller/racer/racer-manual-1-7-7.pdf
15. Visser, U., Stuckenschmidt, H.: Interoperability in GIS - Enabling Technologies. In: Proc. 5th AGILE Conference on Geographic Information Science (2002) 291-297
16. Kawamura, T., De Blasio, J.-A., Hasegawa, T., Paolucci, M., Sycara, K.: Preliminary Report of Public Experiment of Semantic Service Matchmaker with UDDI Business Registry. In: Proc. First International Conference on Service-Oriented Computing (ICSOC 2003) (2003)
17. Paolucci, M., Kawamura, T., Payne, T. R., Sycara, K.: Semantic Matching of Web Service Capabilities. In: Proc. 1st International Semantic Web Conference (ISWC2002) (2002) 333-347
18. Sirin, E., Hendler, J., Parsia, B.: Semi-automatic Composition of Web Services using Semantic Descriptions. In: Proc. "Web Services: Modeling, Architecture and Infrastructure" Workshop (in Conjunction with ICEIS2003) (2003)
19. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-Based Integration of Information — A Survey of Existing Approaches. In: Proc. IJCAI-01 Workshop: Ontologies and Information Sharing (2001) 108-117