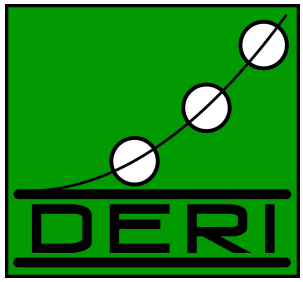

Towards an Ontology Mapping Specification Language for the Semantic Web

Jos de Bruijn Axel Polleres

DERI Technical Report 2004-06-30
June 2004

DERI Galway
University Road
Galway
IRELAND
www.deri.ie

DERI Innsbruck
Technikerstrasse 13
A-6020 Innsbruck
AUSTRIA
www.deri.ie

DERI Technical Report
DERI Technical Report 2004-06-30, June 2004

Towards an Ontology Mapping Specification Language for the Semantic Web

Jos de Bruijn[1], Axel Polleres[1]

Abstract. This paper addresses the requirements for an Ontology Mapping Specification Language for the Semantic Web. We present a set of generic technical use cases and a number of application scenarios, which are used as a basis for a list of requirements for such a mapping language. Furthermore, we discuss further steps to be taken towards a useful language fulfilling the requirements we identify in terms of flexibility and expressiveness.

Keywords: ontology mapping, semantic web, mapping language, ontology alignment, ontology mediation.

[1]Digital Enterprise Research Institute, University of Innsbruck,
Technikerstraße 13, A-6020 Innsbruck, Austria. E-mail: {jos.de-bruijn, axel.polleres}@deri.org

Acknowledgements: The research presented in paper was partially funded by the European Commission in the context of the SEKT project (http://sekt.semanticweb.org/), under contract number IST-2003-506826 and the DIP project (http://dip.semanticweb.org/), under contract number FP6 - 507483.

Contents

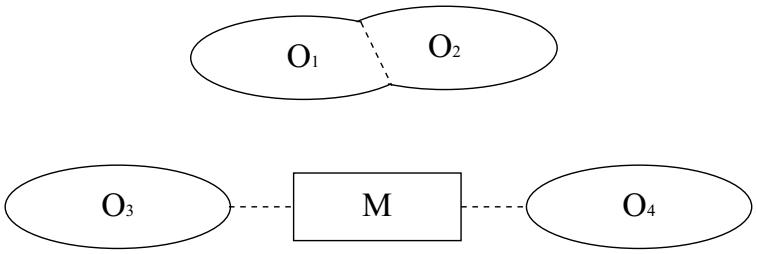

Figure 1: Illustrating tightly coupled (O_1 and O_2) and loosely coupled (O_3 and O_4) ontologies

1 Introduction

With the establishment of the Web Ontology Language OWL [8] as recommendation for the representation of ontologies in the Web, an important step has been taken towards the realization of the Semantic Web. Ontologies represented in OWL can be used to achieve inter-operability between various applications in the Semantic Web. However, in the context of an open environment such as the Web it is very unlikely that there will be very few ontologies shared by many parties; we will have to deal with many different heterogeneous ontologies with overlapping domains.

In order to cope with this heterogeneity between ontologies in the Semantic Web and the relations between ontologies there is a need for means to formally and explicitly specify *ontology mappings* in order to achieve inter-operability. OWL offers limited support for such mappings through the `owl:import` construct which can be used to import one ontology into another one. After importing an ontology, relationships between concepts in the different ontologies can be specified using equivalence and subsumption axioms [26].

We believe that these mechanisms provided by OWL are not sufficient for the specification of ontology mappings in the general case. We see the following pitfalls and shortcomings in specifying ontology mappings directly in OWL:

Tight coupling between ontologies When an ontology is explicitly imported in another ontology, the mapping is already explicitly present in one of the ontologies. This results in a tight coupling between ontologies. Such a tight coupling is undesirable, because it makes one ontology dependent on another in the sense that axioms and definitions in one ontology use classes and properties from the other ontology. This can result in such things as the necessity to use the other, externally specified, ontology in order to perform certain local reasoning tasks. Such strong dependency is undesirable in an open environment as the Semantic Web. In fact, the import of an ontology into another ontology, results in a merged version of the two ontologies. This might be desirable for an ontology merging scenario, but not for the general ontology mapping use case.

Figure 1 illustrates the two situations of tightly coupled and of loosely coupled ontologies. In the Figure, O_1 and O_2 are tightly coupled. It is no longer possible to use one ontology independent of the other. On the other hand, O_3 and O_4 are loosely coupled. Both ontologies can be used independent of each other; the mapping M enables inter-operation between the two.

Inadequate mapping constructs It turns out that certain desired mappings cannot be ex-

pressed using the constructs provided by OWL. Value transformations, for example, cannot be expressed using OWL, see Section 2. In fact, it is not possible to specify certain conditions on data values in OWL, which is often necessary in order to describe a mapping.

We argue that the mapping constructs provided by OWL are also *epistemologically inadequate*, mainly because the OWL language was not conceived as a mapping language, but as an ontology modeling language. Some of the constructs in OWL might be useful to describe relations between ontologies. However, the Description Logic modeling constructs in OWL seem most useful (and mostly used) for the description of merged ontologies and not general ontology mappings.

In order to understand exactly what we mean with ontologies and ontology mapping, we provide definitions of these terms below:

Definition 1.1 *We define an ontology O as a tuple $\langle C, R, I, A\rangle$ where C is a set of concepts, R is a set of relations, I is a set of instances and A is a set of axioms. All concepts, relations, instances and axioms are specified in some logical language. This notion of an ontology coincides with the notion of an ontology described in [24, Chapter 2] and is similar to the notion of an ontology in OKBC [4].* Concepts *correspond with classes in OKBC,* relations *correspond with slots, facets in OKBC are a kind of* axiom *and individuals in OKBC are what we call* instances*.*

Because all concepts, relations, instances and axioms, also referred to a *entities* in the remainder, are specified using a logical language, an ontology can be seen as a logical theory. If we assume an ontology language with a standard model-theoretic semantics[1], the *meaning* of the ontology is then defined by the collection of models of the logical theory. Examples of ontology languages with a standard model-theoretic semantics are OWL [22] and F-Logic [14].

Although instances are logically part of an ontology, it is often useful to separate between an ontology describing a collection of instances and the collection of instances described by the ontology. We refer to this collection of instances as the *Instance Base*. Instance bases are sometimes used to discover similarities between concepts in different ontologies (e.g. [27], [9]). An instance base can be any collection of data, such as a relational database or a collection of web pages. Note that this not rule out the situation where instances use several ontologies for their description. Note also that when a separate instance base is distinguished, this does not mean that there are no more instances in the ontology.

Because ontology modeling is a community activity, the concepts, relations, instances and axioms play an important role in the specification process of the ontology. These entities also play an important role in sharing ontologies and especially in sharing the intended meaning of the concepts in an ontology among humans. It turns out that these entities also play an important role in the discovery and specification of mappings between ontologies.

In order to further clarify the terminology used in this paper, we describe the notion of ontology mediation and the notion of an ontology mapping.

Ontology mediation is the process of reconciling differences between heterogeneous ontologies in order to achieve inter-operation between data sources annotated with and applications using

[1] If a language has a model-theoretic semantics, the meaning (semantics) of any sentence in the language is defined by all the models, which make the sentence true.

these ontologies. This includes the discovery and specification of *ontology mappings*, as well as the use of these mappings for certain tasks, such as query rewriting and instance transformation. Furthermore, merging ontologies also falls under the term ontology mediation.

An *ontology mapping* M is a (declarative) specification of the semantic overlap between two ontologies O_S and O_T . This mapping can be one-way (injective) or two-way (bijective). In an injective mapping we specify how to terms in O_T using terms from O_S in a way that is not easily invertible. A bijective mapping works both ways, i.e. a term in O_T is expressed using terms of O_S and the other way around.

When creating mappings between ontologies, several factors need to be taken into account in choosing the appropriate solution for an Ontology Mapping problem. In particular, we identify the following factors:

- *Expressiveness of the Ontologies.* Ontologies can range in expressiveness from simple taxomonies to very expressive ontologies with arbitrary logic formulas (cf. [6, 18]). The expressiveness of an ontology is of course limited by the expressivity of the ontology representation language. OWL, for example, does not allow arbitrary logic formulas; however, this does not stop ontologies from being built using more expressive languages. The *DOLCE* foundational ontology [11], for example, is specified using full first-order logic with modality; a subset of the ontology has also been specified using OWL.

  Expressive ontologies typically require more complex mappings. With different types of entities in the ontology, also different types of mappings can occur. Another factor, which makes the mapping more complex, is the size of the ontology. Mapping large ontologies typically requires a large number of mapping rules.

- *Precision and coverage of the mapping.* Different scenarios impose different requirements on the mapping. An e-commerce application which requires the translation of purchase orders requires a very precise and complete translation of instances. A semantically-enabled search-engine could live with a simple probabilistic identification of common instances and usually does not require an exact transformation.

- *Desired automation in the mapping process.* Several approaches for (semi-)automation in the mapping process have been proposed (e.g. [23, 20, 1]). The applicability of these automation proposals depends on the complexity of the ontologies[2] and to some extent on the desired precision[3] of the mappings.

  Furthermore, many automation methods depend on the existence of thesauri (e.g. WordNet [10]) which contain the vocabulary used in the ontologies. We argue that a general thesaurus such as WordNet is often inappropriate, because most ontologies are very domain-specific, so that terms used in the vocabulary often do not occur in the thesaurus, which makes the thesaurus less useful for the discovery of similarities.

- *Inconsistencies in Mappings* Mappings between ontologies can create inconsistencies. Especially when translating instances from one representation to another, inconsistencies can

[2]We argue that typically there exists a trade-off between the possible degree of automation in the mapping process and the complexity of the ontologies.

[3]Some applications may require 100% precision in the mappings, whereas other applications can live with lower precision.

arise. We see two types of inconsistencies, which can arise when translating instances from one representation to another: (1) the instance can violate some integrity constraint in the target ontology or (2) the instance has a conflict with another instance of the target ontology, e.g. they might have some conflicting value for a particular attribute. It could be the case that a particular mapping can be used to correctly translate a number of instances, but would introduce inconsistencies with the translation of other instances. This is related to the notion of precision introduced earlier.

The remainder of this paper is organized as follows. In Section 2 we present a selection of generic technical use cases and application scenarios which shall serve as a basis for identifying the requirements for an Ontology Mapping Specification Language. In Section 3 we present requirements for the Ontology Mapping Specification. In Section 4 we propose a conceptual model for a mapping specification language. We present a discussion and related works in Section **??**. We conclude with an outlook to future work and research goals in Section 5.

2 Use Cases and Scenarios for Ontology Mediation

In this section we describe a number of motivating use cases and application scenarios in order to identify the requirements for a language for the specification of ontology mappings.

2.1 Generic Use Cases

This section describes the core technical use cases which need to be supported by a generic Ontology Mediation framework. We distinguish three use cases, which are detailed in the remainder of this section:

- Instance Mediation
- Ontology Merging
- Creating Ontology Mappings

The first use case, Instance Mediation, addresses the tasks of instance transformation, unification and query rewriting. The second use case, Ontology Merging, addresses the way two source ontologies can be merged into one target ontology. The third use case, Creating Ontology Mappings, is about actually finding similarities between ontologies and, on the basis of these similarities, creating mappings between the ontologies.

These three use cases are all inter-related, as we shall see in the following.

2.1.1 Instance Mediation

The following use cases are the typical use cases for instance mediation, where the emphasis is on query rewriting and instance transformation and unification.

Instance mediation is the process of reconciling differences between two instance bases, each described by an ontology. This includes the discovery and specification of ontology mappings, as well

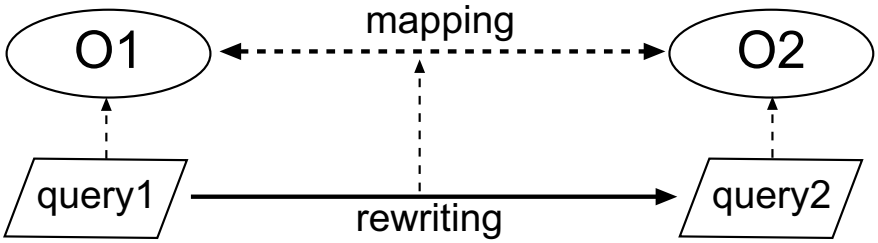

Figure 2: Query Rewriting

as the use of these mappings for certain tasks, such as query[4] rewriting and instance transformation, both described below.

As we can see in the definition, instance mediation also requires the discovery and specification of ontology mappings. This makes apparent the inter-dependencies between the different use cases. We do not describe the discovery and specification of ontology mappings here; instead, these use cases are discussed later, because of the use in different other areas of ontology mediation.

Query Rewriting An operation occurring very frequently in Knowledge Management and data integration applications is querying of information sources. We want to allow an application to query different heterogeneous information sources without actually knowing about all the ontologies. In order to achieve this, a query written in terms of the application's ontology, needs to be rewritten using the terms in the target data source's ontology.

Say, we have an application A, which uses an ontology $\mathcal{O}_A$ for its information representation. Say now that this applications want to query a different data source, which uses ontology $\mathcal{O}_B$, but A does not know about the structure of this ontology. The application A now formulates a query $\mathcal{Q}_A$ in terms of ontology $\mathcal{O}_A$. In order to execute this query on the target data source, it needs to be rewritten into query $\mathcal{Q}_B$, which is formulated in terms of ontology $\mathcal{O}_B$. This rewriting process is illustrated in figure 2. Ideally, the rewritten query can be completely derived from the original query and the mapping.

After execution of the query, the results are transformed back to the $\mathcal{O}_A$ representation and unified with the local instances using the techniques for instance transformation and unification described above.

Instance Transformation For the instance transformation use case we assume two separate applications with separate instance stores both described by ontologies. The task to be performed is the transformation of an instance of a source ontology, say $\mathcal{O}_S$, to an instance of the target ontology $\mathcal{O}_T$. Figure 3 illustrates the process of instance transformation. An instance $i1$, which refers to ontology $O1$, is transformed into instance $i2$, which refers to ontology $O2$. What is important to note here is that the transformation itself is derived from the mapping between the two ontology and that both the original and the transformed instance provide information about the same real-world object. Therefore, the mapping needs to be expressive enough for the specification of this transformation.

[4]A query in this context is a request for information from a user formulated in terms of a particular ontology.

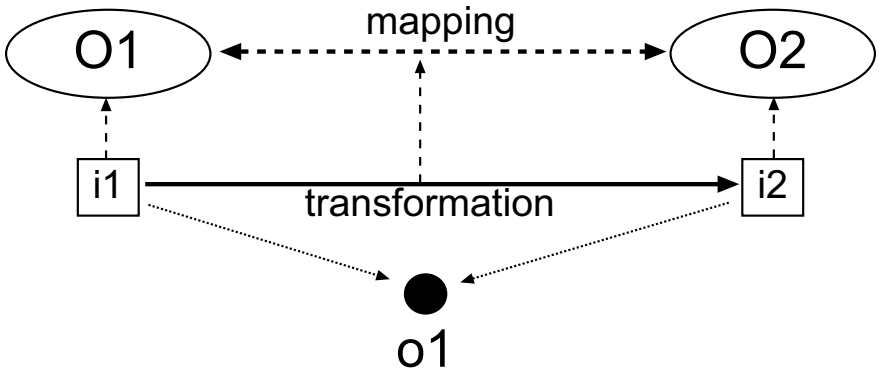

Figure 3: Instance Transformation

This kind of transformation needs to be supported by the ontology mapping in the sense that the ontology mapping specifies the relationship between instances of the source ontology $\mathcal{O}_S$ and instances of the target ontology $\mathcal{O}_T$.

Different application scenarios have different requirements on the precision and coverage of the transformation. With *precision* in this context we mean the degree to which the intended meaning of the instances is preserved in the transformation. In other words, the precision is the fraction of instance data, which is translated correctly. With *coverage* we mean the fraction of instances that are intended to be transformed, which are actually transformed. In other words, the coverage is the fraction of instance data, which should the translated, that is actually translated. The requirements of the application determine the lower bounds for these measures.

When an instance has been translated from $\mathcal{O}_S$ to $\mathcal{O}_T$, it is often necessary to detect whether the transformed instance corresponds to an existing instance in the instance store of the target application in order to avoid duplication of information and in order to find out more about the instances in the knowledge base. We discuss this issue below in the context of instance unification.

Instance Unification The instance unification problem can be summarized as follows:

Say, we have an ontology $\mathcal{O}$ and two instances I_1 and I_2 of that ontology. We want to check whether I_1 and I_2 refer to the same real-world object. In this case we need to unify I_1 and I_2 into a newly created instance I_0, which is the union of I_1 and I_2. Therefore, the instance unification task can be decomposed into (1) the identification of instances referring to the same real-world object and (2) taking the union of the two instances in order to obtain the unified instance. For example, one can use the property Name of a class Person to identify two instances of Person, which describe the same person in the real world. After these two instances of Person have been identified as being the same, the attribute values need to be merged. For example, I_1 could have a value for the property Age, whereas I_2 does not. The merged instance need to inherit the property value of Age from I_1.

If the instances I_1 and I_2 have been identified as referring to the same real-world object, but contain contradictory information[5] with respect to the constraints in the ontology, it is not possible

[5]Notice that not all ontology languages allow contradictory information in the ontology. RDF Schema [2], for example, does not allow contradictory definitions, whereas in OWL this is possible. Therefore, the problem of contradictory instance information is highly interrelated with the expressivity of the ontology language.

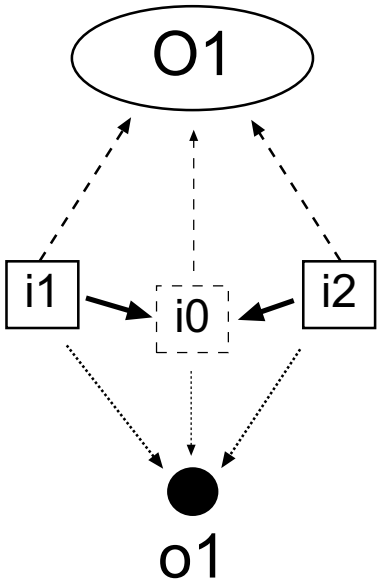

Figure 4: Instance Unification

to create a unified instance and the user should be informed of the inconsistency. For example, if a class Person has a functional property Name and two instances of this class referring to the same person have different names, the user should be informed of this inconsistency.

Figure 4 illustrates the process of instance unification. Two instances ($i1$ and $i2$) of the same ontology $O1$, which refer to the same real-world object $o1$, are unified into one new instance, $i0$, which is the union of both instances, is also an instance of the ontology $O1$ and also describes to the same real-world object $o1$.

We identify two general means of detecting whether two instances refer to the same real-world object:

- In the 'exact' case, the ontology mapping specifies precise, exact conditions which unambiguously specify in which cases two instances refer to the same object and in which cases they refer to different objects, in other words, in which cases the instances are unifiable. These conditions are similar to primary keys in relational databases, where the value of the primary key uniquely identifies a tuple in a relation. This is also similar to the notion of InverseFunctional properties in OWL, where a property value uniquely identifies an instance.

  We see an example of this exact case of instance unification in InfoSleuth [19], where a variant of the SQL query language is used to express the conditions on the fusion of query answers from different data sources. Interestingly, InfoSleuth uses two different decompositions of the global query. The first decomposition is used to query local data sources. The second decomposition is used to fuse query answers.

- In the 'probabilistic' case, a similarity measure is created on the basis of the ontology mapping. The similarity measure expresses the probability that both instances refer to the same object. A threshold could be used to decide whether to unify the instances.

The Bibster bibliographic peer-to-peer system [3] uses a combined similarity measure, taking into account things like string similarity, structural similarity and background knowledge, together with a threshold to identify duplicates in query answers. Duplicates are then visualized as one merged resources, which consists of the union of the properties of all individuals identified as duplicates. In the case of conflicting property values, heuristics are used to select the appropriate value for the property.

Instance transformation and instance unification are often required in a querying scenario where an application A queries another application B and the query results (consisting of instances) are transformed to the representation of A and unified with instances in the instance base of A.

In order to be able to query a data source which uses a different (unknown) ontology, the query originally formulated in terms of the application's ontology needs to be rewritten in terms of the other ontology. The next section describes the generic query rewriting use case.

2.1.2 Ontology Merging

Besides instance transformation, instance unification and query rewriting, we see another major use case for ontology mediation: Ontology Merging.

In the case of Ontology Merging [20], two source ontologies shall be merged into one target ontology[6] based on the source ontologies. In the general case, the source ontologies would disappear and only the target (merged) ontology remains. A special case is when the source ontologies remain, along with mappings to the merged ontology.

In the case where the source ontologies disappear after the merge, the complete instance stores of the source ontologies have to be merged. If the source ontologies remain, the source ontologies can maintain their instance stores and query rewriting, instance transformation and instance unification (see above) are necessary in order to allow querying of the source ontologies through the merged ontology. We can compare these two distinct cases with notions developed in the field of database integration, namely, the notions of *materialised* and *virtual* views [13] respectively.

Of course, when the source ontologies do not have instance stores associated with them, these problems do not occur. However, in the general case an ontology will have one or more instance stores associated with it. In special cases, such as the (distributed) development of ontologies, there will not be instance stores.

2.1.3 Creating Ontology Mappings

In order to be able to support any of the previously mentioned use cases, a mapping needs to be created between the source and the target ontology. Note that this does not apply to the case of ontology merging where the source ontologies do not remain. Because the source ontologies disappear, there needs to be no ontology mapping between these sources and the new merged ontology. However, the techniques for finding entities to be merged in different ontologies and

[6]Note that we do not say here how the merged ontology relates to the original ontologies. This is intentionally left open because not all approaches merge ontologies in the same way. The most prominent approaches are the union and the intersection approaches. In the *union* approach, the merged ontologies is the union of all entities in both source ontologies, where differences in representation of similar concepts have been resolved (c.f. the *union* operator in ontology algebra [32]). In the *intersection* approach, the merged ontology consists only of the parts of the source ontology, which overlap (c.f. the *intersection* operator in ontology algebra [32]).

finding mappings between entities in different ontologies are the same, since they are both based on the similarity of entities. In fact, a mapping between two ontologies can be used as a basis for the merged ontology. In the case of Ontology Merging where the source ontologies remain, a mapping needs to be created between every source ontology and the merged ontology.

We split the "Creating Ontology Mappings" use case into two distinct use cases: finding similarities between ontologies and specifying mappings between ontologies.

Finding Similarities In order to find out which mappings need to be created, similarity between concepts, relations, etc... needs to be established. The similarity between ontologies can either be established manually, or automatically using the so-called *Match* operator (cf. [23]). The *Match* operator takes as input two ontologies and returns as output the similarities between entities[7] in the two source ontologies. These similarities can now be used as a starting point to semi-automatically create a mapping between the ontologies or to merge the two ontologies (cf. [20]).

Specifying Mappings After having defined the similarities between entities in the different ontologies, a mapping needs to be specified between the similar entities of the ontologies. The requirements on this mapping depend on the application scenario (cf. the various scenarios described in the next section) and in general the requirements on ontology mediation, as mentioned in the introduction.

2.2 Scenarios for Ontology Mediation

This section describes a number of typical scenarios for ontology mediation.

2.2.1 Data Integration

Data Integration is concerned with the use of data from different sources in one application. The data from the different sources needs to be presented to the user in a unified way.

Using a Relational Database in a Semantic Web application Relational databases are currently the most popular data storage paradigm in enterprises. As was shown in [30], a large amount of the information currently available over the Web is actually stored in relational databases. This clearly demonstrates the necessity of the ability to use a relational database source in a Semantic Web application.

Typically, a Semantic Web application would not want to deal with the peculiarities of a specific database schema. Especially since legacy database schemas are often specified using incomprehensible, organization- or application-specific relation and attribute names.

In order for the application to use a Relational Database, the database schema has to be "lifted" to the ontology level[8], after which an ontology mapping can be created between the ontology used by the application and the ontology based on the database schema. Examples of relating relational database schemata to ontologies can be found in [5, 17, 31].

[7] Note that current matchers can typically only discover one-to-one similarities, where one entity in O_1 matches one entity in O_2. Matches of other cardinalities (e.g. one-to-many) are often hard to discover for matchers.

[8] This lifting can be done either directly by rewriting the database schema into an ontology (cf. [31]) or indirectly by relating the database schema to an existing ontology [5].

Once a relational data schema has been lifted to the ontology level and provided that the connector lifting the ontology[9] performs a two-way translation, i.e. it translates both instances from the relational representation to the ontology representation and queries from the ontology to the relational representation, the relational data source can be used in a Semantic Web application. In fact, the relational source can then be treated as an ontology with a corresponding instance store.

Using different heterogeneous Ontologies in a Semantic Web application Larger applications typically make use of multiple data sources to fulfill the information needs of its users. For instance, in an organization it could be the case that customer information and employee information are stored in separate sources. An application which wants to provide a search facility to search through all people known to the organization would have to integrate these separate sources.

We distinguish two cases for the use of different ontologies by one application. In the first case, the application uses a global ontology, where all specific local ontologies are mapped to the global ontology. In the second case, we assume a peer-to-peer like setting, where each application has it's own ontology and mappings exist between these different applications.

Using a Global Ontology Because one-to-one mappings between all involved ontologies do not scale in the general case (cf. [29, 28]), it is often preferred to have a central, global ontology to which all local ontologies are mapped.

Using only Local Ontologies In the case of using only local ontologies, an ontology mapping needs to exist between every pair of ontologies, if one wants to mediate between. This approach is not very scalable in general, because it requires $O(n^2)$ ontology mappings, where n is the number of ontologies.

We do not think of one global upper-level ontology for the Semantic Web, but rather different islands, consisting of a central ontology and local ontologies, along with the mappings between them, with mappings between the islands where appropriate. This layering could again define a hierarchy, where other central ontologies can combine several of these islands and so forth.

One could think of creating an ontology for a specific integration problem in an organization or take the (more preferred) approach of relating the local ontologies to a domain ontology, which is a conceptual description of the domain and thus can be shared with others. A clear advantage of this approach is that each application can use its own terminology, while still being able to communicate with other applications, which use different terminologies.

Another interesting approach is the one presented in [26]. This approach assumes a shared light-weight ontology. The concepts in each local ontology are defined in terms of the concepts in the shared ontology. This allows for a certain degree of automation in creating mappings between the local ontologies, because similarities immediately apparent through the use of common terms in the definitions. Limitations of this approach are the fact that the authors of the ontologies need to agree on the use of the shared ontology, which is not always possible, and, because the shared ontology is so light-weight, the concepts in the shared ontology could easily be interpreted differently by authors of different local ontologies.

[9] This lifting involves mediation again which can be viewed as another ontology mediation problem as such using different "ontology languages", for example OWL and SQL. However, in this paper we restrict ourselves to mediation between ontologies in the same language.

2.2.2 Ontology Evolution

One can expect ontologies to evolve over time. This holds also for ontologies that participate in mappings to other ontologies.

If either of the two ontologies involved in an ontology mapping changes (evolves), the mapping might become invalid. Therefore, the mapping between the ontologies needs to evolve together with the ontologies and versioning of both the ontologies and the mapping is required, as we explain below.

Evolving Ontology Mappings Not only ontologies evolve, but also mappings evolve, especially in early stages of the mapping design.

For this scenario we assume static (non-evolving) ontologies and a changing (evolving) mappings between the ontologies. There are various reasons why a mapping may evolve. Examples are evolving insights into the source and target ontology and their similarities, new requirements on the mapping (e.g. a new subpart of the ontologies needs to be mapped, which was not considered before) and inadequate or faulty specification of the mapping.

This scenario does not only indicate the need for evolution support for ontology mappings, but also for versioning. Each change of the mapping requires a new uniquely identifiable and accessible version of the mapping, so that applications which use a certain version do not break because of changes in the mapping.

Mapping different versions of Evolving Ontologies The evolution of ontology mappings is relatively simple compared to the mapping of evolving ontologies. In this case, both the source and the target ontology evolve over time and the mapping between the ontologies needs to evolve accordingly. The mapping between an evolving source ontology and a target ontology might even make apparent the need for the target ontology to evolve along with the source ontology in order to enable inter-operation.

The evolution of ontologies has the following implications for the mappings between ontologies:

- *Versioning* of the ontologies is required and the ontology mapping needs to be specified between *specific versions* of the source and target ontology. The mapping needs to refer to specific versions of the ontology.

- Evolution of ontologies indicates the need for *evolution of ontology mappings*. In many cases, when a new version of an ontology is created, a new version of each of the mappings in which the ontology is involved needs to be created. If the changes in the ontologies are formally and explicitly documented, these changes can be used as the basis for changes to be made to the mapping. We believe that in many cases, the evolution of the mapping can be done semi-automatically.

- Evolution of an ontology $\mathcal{O}_B$ which is mapped to an ontology $\mathcal{O}_A$ is mapped, might *indicate the need for subsequent evolution of* $\mathcal{O}_A$. An example of this is a case of database integration, using the local-as-view (cf. [15]) paradigm, where $\mathcal{O}_B$ is the global ontology and $\mathcal{O}_A$ is the local ontology.

Mappings between different Ontology Versions When an ontology on the Semantic Web evolves, some mappings to the old ontology might become invalid. This problem can be solved by either evolving the mapping or by mapping to a specific version of the ontology.

When mapping to a specific version, problems may occur, for example because the instance base of the ontology evolves with the ontology. It is not feasible to maintain an instance base for each version of an ontology, because this can easily lead to inconsistencies and would actually be a maintenance nightmare. A way to deal with this problem is by providing a mapping between different versions of one ontology.

Say, $\mathcal{O}_S$ evolves from $\mathcal{O}_{Sv1}$ into $\mathcal{O}_{Sv2}$ and a (two-way) mapping M_S is created between $\mathcal{O}_{Sv1}$ and $\mathcal{O}_{Sv2}$. Now, when there exists a mapping M_{S-T} between $\mathcal{O}_{Sv1}$ and $\mathcal{O}_T$, this mapping can ideally be automatically combined with M_S in order to create a new mapping between $\mathcal{O}_{Sv2}$ and $\mathcal{O}_T$.

3 Requirements for an Ontology Mapping Specification Language

This section shall present a number of requirements on the Ontology Mapping Specification Language, partly derived from the use cases and scenarios presented in the previous section. Furthermore, we present a number of additional requirements that follow from our setting.

Particularly, we identify the following requirements on an Ontology Mapping Specification Language:

Mapping on the Semantic Web Our goal is to develop an ontology mapping language for the Semantic Web. Therefore, we must be able to specify mappings between ontologies on the Web and the ontology mapping itself must also be available on the Web. The current standard for specifying ontologies on the web is the Web Ontology Language OWL [8]. We must therefore support mapping between ontologies written in OWL.

Mapping between Description Logic Ontologies An important species of OWL is OWL DL, which is a syntactical variant of the $\mathcal{SHOIN}(\mathbf{D})$ Description Logic language [12]. Therefore, mappings between OWL ontologies can be reduced to mappings between Description Logic ontologies.

Specify Instance Transformations It follows from the generic use cases presented in the previous section that the ontology mapping language must support transformations between instances of the different ontologies. In fact, [23] defines the mapping process as the set of activities required to transform instances of the source ontology into instances of the target ontology. Also MAFRA [16] explicitly addresses the transformation of instances on the basis of a mapping between two ontologies.

In instance transformation, we identify two dimensions: structural transformation and value transformation:

- A *structural* transformation is a change in the structure of an instance. This means that an instance might be split into two instances, two instances might be merged into one, properties might be transformed to instances, etc. For example, an instance of the concept PhD-Student in one ontology might need to be split into two instances, one of Student and one of Researcher, in the target ontology. A different example is the use

of aggregate functions. An ontology O_S might have a concept Parent with a property hasChild, whereas the ontology O_T might also have a class Parent, but in this case only with the property nrOfChildren. An aggregate function is required to count the number of children of a specific parent in O_S in order to come up with a suitable property filler for nrOfChildren.

- A *value* transformation is a simple transformation from one value into another. An example of such a value transformation is the transformation from kilograms into pounds

An example of a transformation, which requires both a structural and two value transformations is the transformation from a full name to separate first and last names. Splitting the full name instance into both the first and the last name requires structural transformation. After the structural transformation, two value transformations are required; on for the first and one for the last name.

Specify Instance Similarity Conditions One of the generic use cases presented in section 2 is the instance unification use case. It turned out in this use case that when instances need to be unified, first the similarity between the instances must be established. In order to detect the similarity, one can compare the values of all properties and describe the similarity as a function over all the individual property similarities. The other extreme is to designate one property as the identifying property (cf. primary keys in relational databases) and designate instances that have equivalent values for these designated properties as equivalent and unify them[10].

We shall take a hybrid approach, where it is possible to specify equality of instances as a logical condition over its property values. We call this the *exact* approach for instance unification. In the second case, the *probabilistic* approach, it is possible to specify a matching function over the property values, which yields a probability between 0 and 1, specifying the similarity between the instances. When combined with a threshold, this function also becomes a condition for similarity.

We are currently not aware of any existing approach which addresses the specification of instance similarity in the same sense we do here.

Query Rewriting and Ontology Merging The Query Rewriting and Ontology Merging use cases presented in the previous chapter indicate the need for the ontology mapping to not only map instances of the ontologies but to also map concepts and relations in the source and target ontologies. This is necessary for the case when a query written in terms of ontology $\mathcal{O}_S$ needs to be executed on an instance base, which is described by ontology $\mathcal{O}_T$. The mapping needs to specify exactly how concepts and relations in $\mathcal{O}_S$ relate to concepts in relations in $\mathcal{O}_T$ in order to enable the rewriting.

After execution of the query, the result instances need to be transformed back to $\mathcal{O}_S$ which involves all requirements for instance transformation described above. The querying use case scenario does, however, indicate the need for a mapping which supports query rewriting in one direction and instance transformation in the other direction.

[10]Note that, as with primary keys in relational databases, it is possible to designate several properties as the unique identification for instances

One mapping for all tasks It is clearly advantageous to have one common declarative mapping language, which suffices for the different use cases of instance transformation, instance unification, query rewriting and ontology merging.

MAFRA [16] combines relating entities (such as concepts and relations) in ontologies with instance transformations. So-called *semantic bridges* specify the relationship between entities in different ontologies. Each instance of a semantic bridge has a transformation attached to it, which specify the instance transformations. The semantic bridges can be used for query rewriting and ontology merging, whereas the attached transformations can be used for instance transformation.

Use of Mapping patterns It is our expectation that many similar ontologies will appear on the Semantic Web. When many similar ontologies exist in a specific domain, the mappings between the ontologies will also be similar. In order to capture these similarities and to reuse existing mapping specification we aim to identify recurring mapping patterns. A mapping pattern can be seen as a template for mappings between classes of ontologies, which can be instantiated to create specific mappings between specific ontologies (cf. [21]).

Mapping patterns furthermore reduce the complexity of a mapping for the user and can be used as a way to modularize a mapping.

Versioning support The ontology mapping language must support constructs for the versioning of the mapping and for referring to specific version of the source and target ontologies.

The latter of course depends on the scheme chosen for ontology versioning. In the case of a new name for each new version of the ontology, no additional provisions have to be taken in the mapping language. This is currently the only way to do versioning in the Web Ontology Language OWL. Therefore, we will assume this situation.

Treating classes as instances Different ontologies might be modeled within slightly different domains with different granularity. What is seen as a class in one ontology might be seen as an instance of a different class in another ontology [25]. In order to support inter-operation between two ontologies with such differences, classes need to be mapped to instances and vice versa.

In fact this can be seen more general. The mapping language should support the mapping of any entity in the source ontology, whether it is a class, instance, relation, to any entity in the target ontology. For example, it should be possible to have a relation-instance mapping, a class-relation mapping, etc.

Mappings of different cardinalities It might be necessary to map a class in one ontology to a number of different classes in the other ontology. It might also be necessary to map a class in one ontology to a class and a relation in the other ontology. In other words, the language needs to support mappings of arbitrary cardinalities. One-to-one mappings are not enough.

4 Mappings and Mapping Patterns

In order to specify ontology mappings in an epistemologically adequate way, we plan to use elementary mapping patterns (similar to the semantic bridges presented in [16]) as the building blocks for the mapping language.

Elementary mapping patterns can also serve as building blocks for creating complex mapping patterns which capture a recurring pattern of ontology mappings. A mapping pattern is instantiated for a particular pair of ontologies into an ontology mapping. In fact, a complex mapping pattern in *composed of* other (elementary of complex) mapping patterns and a mapping is *composed of* a number of (instantiated) mapping patterns.

An ontology mapping can be *uni-directional* or *bi-directional*.

- A *uni-directional* mapping specifies the relationship between two ontologies in only one direction. This means for the task of instance transformation that the mapping can be used to transform an instance from the $\mathcal{O}_S$ to the $\mathcal{O}_T$ representation, but not the other way around. For the query rewriting task this means that a query can only be rewritten from the $\mathcal{O}_S$ to the $\mathcal{O}_T$ vocabulary, which means a mapping in the other direction is necessary for the transformation of the query results.
- A *bi-directional* mapping specifies the relation between two ontologies in both directions. This means that the same mapping can be used to transform instances both from $\mathcal{O}_S$ to $\mathcal{O}_T$ and the other way around. The same holds for the query rewriting.

  Note that in order to specify the merge of two ontologies, a bi-directional mapping is required.

Bi-directional mappings are in general favorable over uni-directional mappings, because there is more flexibility. However, specifying the bi-directional mapping might require more effort than is actually necessary for the task at hand. If, for example, it is only required to transform instances from $\mathcal{O}_S$ to $\mathcal{O}_T$, it would be a waste of effort to specify both directions. Furthermore, a bi-directional mapping might be infeasible or even impossible, especially if aggregate functions come into play.

In the remainder, when we talk about an ontology mapping, we mean a *uni-directional* ontology mapping. A bi-directional mapping can always be decomposed into two uni-directional mappings.

One requirement stated in the previous section is the support for mapping patterns. We see two types of mapping patterns, namely, *elementary* mapping patterns, also called "Semantic Bridge" [16] or "Mapping Type" [21] and *complex* mapping patterns, simply referred to as "Mapping Patterns" in [21].

The Ontology Mapping Specification Language will support the use of elementary mapping patterns as constructs of the language itself. An elementary pattern specifies the relation between one or more ontology constructs of the source ontology and one or more ontology constructs of the target ontology. The mapping pattern describes both the relation and the necessary instance transformation in a declarative way. When the mapping pattern is instantiated to a mapping relation, the abstract ontology constructs in the patterns are filled-in with the actual ontology constructs from the source and target ontologies and the relation and the transformation can be further refined.

An additional benefit that these mapping patterns could bring is when mapping patterns would be associated with similarity detection methods. A *Match* operator could then detect mapping patterns based on certain similarities between the different ontologies.

5 Future Work

The work presented in the paper is the first step towards as Ontology Mapping Specification Language for the Semantic Web. The requirements formulated in section 3 will be taken as a

starting point for this ongoing work on the language.

From the requirements outlined in Section 3 it becomes apparent that a mapping language for the Semantic Web needs to inter-operate with current Semantic Web standards. On the other hand, the Web Ontology Language OWL is not enough to express all mappings we require and a rule language is required for such mappings. As we have argued in [7], rule extension of OWL is not straightforward, because straightforward combination of Description Logic and Horn Logic can easily lead to undecidability. Therefore, we propose an approach based on the fragment of OWL, which is expressible in Datalog, as a starting point for a rule language for the Semantic Web [7]. This rule language will be further developed in the context of the Web Service Modeling Language Working Group WSML[11].

An additional benefit of using LP as a formal basis for ontology mapping is the well-known and well-understood number of extensions of Logic Programming (LP) formalisms like Datalog, such as default negation, classical negation, function symbols, etc. Furthermore, there is well-established research on the use of LP for query answering.

In order to develop an epistemologically adequate mapping language we will develop a library of elementary and complex mapping patterns, which will form the basis for ontology mappings.

Our future work will furthermore consist of the following:

Relating axioms We have not yet considered relating axioms in different ontologies to each other. We believe that this would not be common in an ontology mapping scenario, but it would be common in an ontology merging scenario, where certain constraints need to be merged.

Handling semi-structured data In this paper, we have considered only instances of one particular ontology, which have to be transformed to a different representation. One could imagine that data could be annotated with several different ontologies or even data could not be annotated at all but be self-describing. In future work we will consider how to handle such semi-structured data.

Static vs. Dynamic mappings Besides the evolution of mappings, which we mentioned in Section 2, which can be seen as a dynamic aspect of ontology mappings, we see other issues arising with dynamic aspects of ontology mappings. For example, a mapping function which translates a Euro currency representation to a Dollar representation depends on the exchange rate between the Euro and Dollar currency, which changes over time.

There are different ways of handling such dynamicity. One obvious way is to assume an oracle beyond the mapping language, which provides the required information at the appropriate time. In the example, the oracle would provide the current exchange rate at the time an instance needs to be translated from one representation to the other. Another possibility is to use a Web Service to retrieve the appropriate information. A transformation function could be associated with a Web Service, which provides the required functionality, or with a goal (cf. [24]), which describes the required functionality and can be used to automatically discover a server, which offers the functionality.

[11] http://www.wsmo.org/wsml/

References

[1] Sonia Bergamaschi, Silvana Castano, Maurizio Vincini, and Domenico Beneventano. Semantic integration of heterogeneous information sources. *Special Issue on Intelligent Information Integration, Data & Knowledge Engineering*, 36(1):215–249, 2001.

[2] Dan Brickley and Ramanathan V. Guha. RDF vocabulary description language 1.0: RDF schema. Recommendation 10 February 2004, W3C, 2004. Available from http://www.w3.org/TR/rdf-schema/.

[3] Jeen Broekstra, Marc Ehrig, Peter Haase, Frank van Harmelen, Maarten Menken, Peter Mika, Björn Schnizler, and Ronny Siebes. Bibster – a semantics-based bibliographic peer-to-peer system. In *Proceedings of SemPGRID '04, 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing*, pages 3–22, New York, USA, May 2004.

[4] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. Rice. OKBC: A programmatic foundation for knowledge base interoperability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 600–607, Madison, Wisconsin, USA, 1998. MIT Press.

[5] Jos de Bruijn. Semantic information integration within and across organizational boundaries. Master Thesis, TU Delft, The Netherlands, 2003.

[6] Jos de Bruijn. Using ontologies - enabling knowledge sharing and reuse on the semantic web. Technical Report DERI-2003-10-29, DERI, 2003. Available from http://homepage.uibk.ac.at/∼c703239/publications/DERI-TR-2003-10-29.pdf.

[7] Jos de Bruijn, Axel Polleres, and Dieter Fensel. OWL lite$^-$. Deliverable d20v0.1, WSML, 2004.
Available from http://www.wsmo.org/2004/d20/v0.1/.

[8] Mike Dean and Guus Schreiber, editors. *OWL Web Ontology Language Reference*. 2004. W3C Recommendation 10 February 2004.

[9] AnHai Doan, Jazant Madhaven, Pedro Domingos, and Alon Halevy. Ontology matching: A machine learning approach. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies in Information Systems*, pages 397–416. Springer-Verlag, 2004.

[10] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1999.

[11] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. Sweetening ontologies with DOLCE. In *Proceedings of EKAW 2002*, Siguenza, Spain, 2002.

[12] Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, Sanibel Island, Florida, 2003.

[13] Richard Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *ACM Symposium on Principles of Database Systems*, pages 51–61, Tuscon, Arizona, USA, 1997.

[14] Michael Kifer, Geord Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *JACM*, 42(4):741–843, 1995.

[15] Alon Y. Levy. *Logic-Based Techniques in Data Integration*, pages 575–595. Kluwer Publishers, 2000.

[16] Alexander Maedche, Boris Motik, Nu no Silva, and Raphael Volz. Mafra a mapping framework for distributed ontologies. In *Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW-2002*, Madrid, Spain, 2002.

[17] Andreas Maier, Hans-Peter Schnurr, and York Sure. Ontology-based information integration in the automotive industry. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proceedings of the Second International Semantic Web Conference (ISWC2003)*, number LNCS 2870, pages 897 – 912. Springer-Verlag, 2003.

[18] Deborah L. McGuinness. Ontologies come of age. In Dieter Fensel, James Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, chapter 6, pages 171–194. MIT Press, 2003.

[19] Marian H. Nodine, Jerry Fowler, Tomasz Ksiezyk, Brad Perry, Malcolm Taylor, and Amy Unruh. Active information gathering in infosleuth. *International Journal of Cooperative Information Systems*, 9(1-2):3–28, 2000.

[20] Natalya F. Noy and Mark A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proc. 17th Natl. Conf. On Artificial Intelligence (AAAI2000)*, Austin, Texas, USA, July/August 2000.

[21] John Y. Park, John H. Gennari, and Mark A. Musen. Mappings for reuse in knowledge-based systems. In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modelling and Management (KAW 98)*, Banff, Canada, 1998.

[22] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. Recommendation 10 February 2004, W3C, 2004.

[23] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.

[24] Dumitru Roman, Holger Lausen, and Uwe Keller. Web service modeling ontology standard (WSMO-standard). Working Draft D2v0.2, WSMO, 2004.

[25] Guus Schreiber. The web is not well-formed. *IEEE Intelligent Systems*, 17(2), 2002. Contribution to the section Trends and Controversies: Ontologies KISSES in Standardization.

[26] Heiner Stuckenschmidt and Frank van Harmelen. *Information Sharing on the Semantic Web*. Springer, 2004. to appear.

[27] Gerd Stumme and Alexander Maedche. Fca-merge: Bottom-up merging of ontologies. In *7th Intl. Conf. on Artificial Intelligence (IJCAI '01)*, pages 225–230, Seattle, WA, USA, 2001.

[28] Mike Uschold. Creating, integrating, and maintaining local and global ontologies. In *Proceedings of the First Workshop on Ontology Learning (OL-2000) in conjunction with the 14th European Conference on Artificial Intelligence (ECAI-2000)*, Berlin, Germany, August 2000.

[29] Pepijn R. S. Visser and Zhan Cui. On accepting heterogeneous ontologies in distributed architectures. In *Proceedings of the ECAI98 workshop on applications of ontologies and problem-solving methods*, Brighton, UK, 1998.

[30] Raphael Volz, Siegfried Handschuh, Steffen Staab, Ljiljana Stojanovic, and Nenad Stojanovic. Unveiling the hidden bride: deep annotation for mapping and migrating legacy data to the semantic web. *Journal of Web Semantics*, 2004. to appear.

[31] Raphael Volz, Daniel Oberle, Steffen Staab, and Rudi Studer. Ontolift prototype. Deliverable 11, WonderWeb, 2003.

[32] Gio Wiederhold. An algebra for ontology composition. In *Proceedings of 1994 Monterey Workshop on formal Methods*, pages 56–61, U.S. Naval Postgraduate School, Monterey CA, 1994.