MAFRA — A MApping FRAmework for Distributed Ontologies in the Semantic Web

Alexander Maedche[1], Boris Motik[1], Nuno Silva[12], and Raphael Volz[1]

[1] FZI Research Center for Information Technologies at the University of Karlsruhe, D-76131 Karlsruhe, Germany
{maedche,motik,silva,volz}@fzi.de
[2] ISEP Instituto Superior de Engenharia, Instituto Politecnico do Porto, Portugal

Abstract. Ontologies as means for conceptualizing and structuring domain knowledge within a community of interest are seen as a key to realize the Semantic Web vision. However, the decentralized nature of the Web makes achieving this consensus across communities difficult, thus, hampering efficient knowledge sharing between them. In order to balance the autonomy of each community with the need for interoperability, mapping mechanisms between distributed ontologies in the Semantic Web are required. In this paper we present MAFRA, an interactive, incremental and dynamic framework for mapping distributed ontologies in the Semantic Web.

1 Introduction

The current WWW is a grea1t success with respect to the amount of stored documents and the number of users. However, the ever-increasing amount information on the Web places a heavy burden of accessing, extracting, interpreting and maintaining information on the human users of Web. Tim Berners-Lee, the inventor of the WWW, coined the vision of Semantic Web, providing means for annotation of Web resources with machine-processable metadata providing them with background knowledge and meaning (see [2]). Ontologies as means for conceptualizing and structuring domain knowledge are seen as the key to enabling the fulfillment of the Semantic Web vision.

However, the de-centralized nature of the Web makes indeed inevitable that communities will use their own ontologies to describe their data. In this vision, ontologies are themselves distributed and the key point is the mediation between distributed data using mappings between ontologies. Thus, complex mappings and reasoning about those mappings are necessary for comparing and combining ontologies, and for integrating data described using different ontologies [15]. Existing information integration systems and approaches (e.g., TSIMMIS [6], Information Manifold [8], Infomaster[3], MOMIS[4], Xyleme [5]) are "centralized" systems of mediation between users and distributed data sources, which exploit mappings between a single mediated schema and schemas of data sources. Those mappings are typically modeled as views (over the mediated schema in the local-as-view approach, or over the sources schemas in the global-as-view approach) which are expressed using languages having a formal semantics. For scaling up to the Web, the "centralized" approach of mediation is probably not flexible enough, and distributed systems of mediation are more appropriate.

Building on this idea and on existing work, we introduce in this paper MAFRA, an Ontology MApping FRAmework (MAFRA) for distributed ontologies in the Semantic Web. Within MAFRA we provide an approach and conceptual framework that provides a generic view onto the overall distributed mapping process. The distributed nature of Semantic Web entails significant degrees of information redundancy, incoherence and constant evolution, thus changing the nature of the ontology mapping problem: instead of creating a static specification document relating entities in two ontologies, a continuous, incremental, interactive and highly dynamic process supporting mapping evolution is required to scale up to the ever-changing nature of ontologies being mapped. Establishing a mapping between two ontologies is an engineering process of consensus building between two communities already agreeing on common ontologies for their own respective domains. This task implies negotiation, so attention is paid to providing means for cooperative mapping. Thus, proposed framework offers support in all parts of the ontology mapping life-cycle.

Organization of this paper. In section 2 we motivate our work by introducing an application for whose success a solution to the distributed ontology mapping problem is required. Based on this application, we have collected the requirements for developing MAFRA. In section 3 we introduce the underlying conceptual architecture of MAFRA. In section 4 we focus on mapping representation and present the current status of our semantic bridging ontology and discuss its features. Section 5 presents the realized mapping implementation within KAON - an ontology and Semantic Web application framework[6]. Before we conclude a short discussion of related and future work is given in section 6.

2 MAFRA – Application Scenarios

Design of MAFRA recognizes specific requirements of several concrete application scenarios. In this section we present one of these scenarios and discuss its respective requirements.

"People can't share knowledge if they do not speak a common language". This simple insight accurately characterizes what makes knowledge management a challenging task. Its goal to reach global knowledge access within different departments of an enterprise is usually difficult due to the fact that different departments usually encompass different vocabularies, which hinders communication. Large companies

[3] http://infomaster.stanford.edu/infomaster-info.html
[4] http://sparc20.ing.unimo.it/Momis/
[5] http://www.xyleme.com
[6] http://kaon.semanticweb.org

typically consist of departments such as Human Resources, Production, Sales, Marketing and Finance. By using ontologies, the task of collecting, organizing, and distributing the knowledge within one department may be solved – ontologies provide a sound semantic basis for the definition of meaning that can be understood by both humans and machines. Also, a single department is typically small enough so that achieving consensus among interested parties is feasible. However, designing a large-scale ontology covering the needs of all departments has shown to be a very difficult task due to effort, scale and maintainability. Interoperability between departments can then be achieved by mapping of ontologies of each department. It is anticipated that mapping existing ontologies will be easier than creating common ontology because a smaller community is involved in the process. It is important to emphasize that we do not consider a closed world and centralized information integration system as a possible solution for the problem introduced above.

Ontologging[7] is an ontology-based environment tackling this problem. It builds on Semantic Web standards with the goal of enabling next generation knowledge management applications allowing management and usage of multiple ontologies. An important requirement within the development of the Ontologging multi-ontology system is that there exists extensive tool support for supporting the overall mapping process. A specific requirement was the support of automatic detection of similarities of entities contained in the two different department ontologies.

3 Conceptual Framework

An ontology mapping process, as defined in [14], is the set of activities required to transform instances of a source ontology into instances of a target ontology. By studying the process and analyzing different approaches from the literature [14] we observed a set of commonalities and assembled them into the MAFRA conceptual framework, outlined in Figure 1. The framework consists of five horizontal modules describing the phases that we consider fundamental and distinct in a mapping process. Four vertical components run along the entire mapping process, interacting with horizontal modules.

3.1 Horizontal Dimension of MAFRA

Within the horizontal dimension, we identified following five modules:

Lift & Normalization. This module focuses on raising all data to be mapped onto the same representation level, coping with syntactical, structural and language heterogeneity [16]. Both ontologies must be normalized to a uniform representation, in our case RDF(S), thus eliminating syntax differences and making semantics differences between the source and the target ontology more apparent [14]. To facilitate that, we developed a LIFT tool providing means to bring DTDs, XML-Schema, and relational databases to the structural level of the ontology. Lift is not further elaborated in this paper - we shall simply assume that the source and target ontologies are already represented in RDF-Schema with their instances in RDF.

[7] http://www.ontologging.com

Similarity. This module establishes similarities between entities from the source and target ontology. Similarity between conceptual models is hard to measure and often establishing a suitable similarity measure is a very subjective task. Several different similarity measures have been proposed in literature [14, 3, 5, 10, 1], focusing on different aspects of ontology entities. We don't further elaborate on this issue, as it is not in scope of this paper.

Semantic Bridging. Based on the similarities computed in the previously described phase, the semantic bridging module is responsible for establishing correspondence between entities from the source and target ontology. Technically, this is accomplished by establishing semantic bridges - entities reflecting correspondence between two ontology entities. Apart from the semantic correspondence, additional "procedural" information is needed to further specify the transformation to be performed, e.g. translation of measures like currencies. Semantic bridging is further discussed in section 4.

Execution. This module actually transforms instances from the source ontology into target ontology by evaluating the semantic bridges defined earlier. In general two distinct modes of operation are possible, namely offline (static, one-time transformation) and online (dynamic, continuous mapping between source and the target) execution. Execution issues further discussed in section 5.

Post-processing. The post-processing module takes the results of the execution module to check and improve the quality of the transformation results. The most challenging task of post-processing is establishing object identity - recognizing that two instances represent the same real-world object [7]. Furthermore, by computing statistical properties of transformed instances, it is possible to check whether semantic bridges were underspecified.

3.2 Vertical Dimension of MAFRA

The vertical dimension of MAFRA contains modules that interact with horizontal modules during the overall mapping process. Following four modules have been identified and will be only shortly mentioned in this paper:

Evolution. This modules focuses on keeping semantic bridges obtained by the "Semantic Bridge" module, which must be kept in synchrony with the changes in the source and target ontologies. Evolving ontologies on the Semantic Web result in an update requirement of the corresponding semantic bridges. Although this may be achieved by reapplying the mapping process, this is probably not the most efficient or accurate way. Thus, the mapping process must have an evolution component that will reuse the existing semantic bridges in adapting them to new requirements.

Cooperative Consensus Building. The cooperative Consensus Building module is responsible for establishing a consensus on semantic bridges between two communities participating in the mapping process. This is a requirement as one has to choose frequently from multiple, alternatively possible mappings .The amount of human involvement required to achieve consensus may be reduced by automating the mapping process as much as possible.

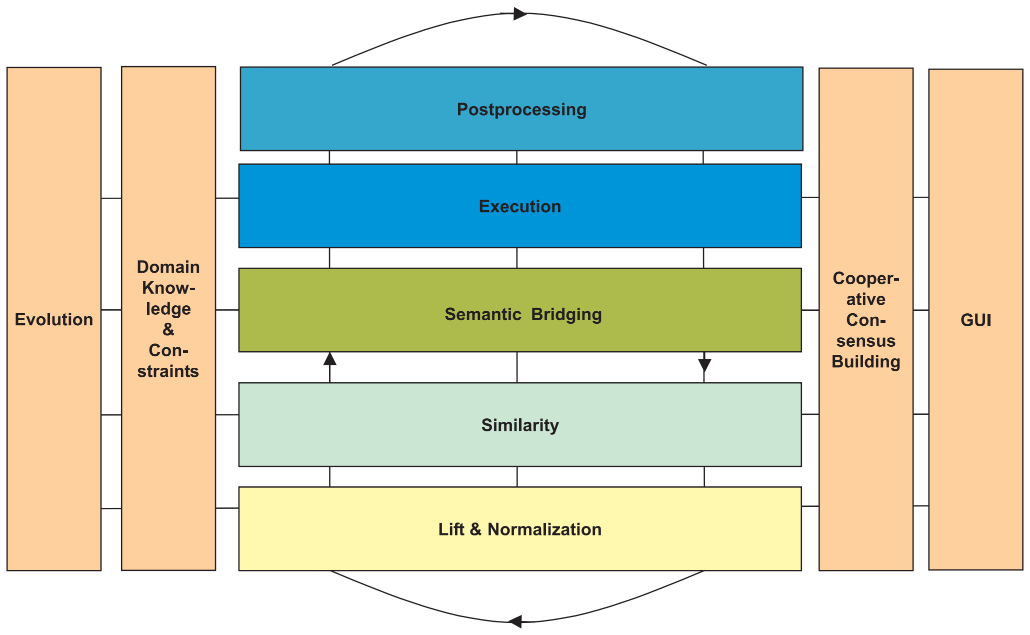


Fig. 1. Conceptual Architecture

Domain Constraints and Background Knowledge. The quality of similarity computation and semantic bridging may be dramatically improved by introducing background knowledge and domain constraints, e.g. by using glossaries to help identify synonyms or by using lexical ontologies, such as WordNet or domain-specific thesauri, to identify similar concepts.

Graphical User Interface. Mapping is a difficult and time consuming process, which is not less difficult than building an ontology itself, i.e. deep understanding of both conceptualizations required on human side, thus extensive graphical support must be given and it is a separate issue how this can be achieved in an optimal way. The graphical user interfaces (GUI) modules allows the users drive the mapping process, provide domain constraint and background knowledge, create semantic bridges, refine bridges according to the results of the execution module, etc. Some aspects of the GUI are further elaborated in section 5.

4 Semantic Bridging

As mentioned in subsection 3.1, the role of the semantic bridging component is to semantically relate entities from the source and target ontologies. This is achieved by creating so-called semantic bridges. A role of a semantic bridge is to encapsulate all necessary information to transform instances of one source ontology entity to instances of one target ontology entity. In the rest of this section we first explore the nature of semantic bridges by analyzing their different dimensions determining each bridge. Next we discuss our approach of using a meta-ontology to enable the specification of semantic bridges. At last we give an example of how semantic bridges can be defined between two domain ontologies.

4.1 Dimensions of Semantic Bridges

The nature of semantic bridges may be understood by considering different dimensions, each describing one particular aspect of a semantic bridge. By analyzing ontologies used on the Semantic Web, we identified following five dimensions of semantic bridges:

1. **Entity dimension** reflects the type of ontology entities being bridged,
2. **Cardinality dimension** reflects the number of ontology entities being bridged,
3. **Structural dimension** reflects the way how elementary bridges may be combined into more complex bridges,
4. **Constraint dimension** reflects constraints applied during the execution phase to instances from the source ontology,
5. **Transformation dimension** reflects how instances of the source ontology are transformed during the mapping process.

Entity dimension. Semantic bridges may relate the ontology entities *(i)* concepts (modeling classes of objects from the real world), *(ii)* relations (modeling relationships between objects in the real world), and, *(iii)* attributes (modeling simple properties of objects in the real world) and *(iv)* extensional patterns (modeling the content of the instances).

Cardinality dimension. This dimension determines the number of ontology entities at both sides of the semantic bridge, ranging from $1:1$ to $m:n$. However, we have found that in most cases $m:n$ is not a common requirement, so $1:n$ and $m:1$ suffice. Even when $m:n$ are encountered, often they may be decomposed into m $1:n$ bridges.

Structural dimension. This dimension reflects the way how elementary bridges may be combined into more complex bridges. We distinguish between the following different relations that may hold between bridges:

– **Specialization** allows a bridge to reuse definitions from another bridge and provide additional information (e.g. a bridge relating Employee concepts from two ontologies may be a specialization of a more general bridge relating Person concepts),

- **Abstraction** is a variation of the type of the superclasses. When this attribute is set, the specified bridge should not be executed independently, but only as superclass of another.
- **Composition** relation between to bridges specifies that a bridge is composed of other bridges,
- **Alternatives** relation between bridges specifies a set of mutually exclusive bridges.

Constraint dimension. The constraint dimension permits to control the execution of a semantic bridge. It reflects relevant constraints applied during the execution phase to instances from the source ontology. Constraints act as conditions that must hold in order the transformation procedures is applied onto the instances of the source ontology, e.g. the bridge evaluate only if the value of the source instance matches a certain pattern.

Transformation dimension. This dimension reflects how instances of the source ontology are transformed during the mapping process. Transformations assume different complexity and variety depending on the ontologies being bridged.

4.2 Semantic Bridging Ontology (SBO)

Within our approach four different types of relations between entities, a particular semantic bridge exists. A specification of all available semantic bridges, organized in a taxonomy, is a semantic bridging ontology (SBO). To actually relate the source and target ontology, the mapping process creates an instance of SBO containing semantic bridge instances, each encapsulating all necessary information to transform instances of one source entity to instances of the target entity. In the following sections we will describe the semantic bridging ontology in more detail.

Figure 2 describes the most important entities of the semantic bridging ontology. We refer to the five, previously described semantic bridge dimensions:

- Three basic types of entities are considered: Concepts, Relations and Attributes,
- The class SEMANTIC BRIDGE is the most generic bridge, it defines the relations to source and target entities. It is specialized according to the entity type and according to cardinality. Though, there are many combinations of entity types and cardinality bridges that are not explicitly specified, it is important to mention that they can be easily specialized from more general bridges.
- The class SERVICE represents a class used to reference resources that are responsible to connect to, or describe transformations. This class is intended to be used to describe these transformations resources. Because services are normally external to the execution engine, it is required to describe some fundamental characteristics like name, interface (number and type of arguments) and location. Argument and its sub classes Arg and ArgArray permits to describes these characteristics in a simple and direct form.
- RULE is the general class for constraints and transformation-relevant information, which provides a relation to the service class.
- The class TRANSFORMATION is mandatory in each semantic bridge except if the semantic bridge is set as abstract. It uses the inService relation to link to the transformation procedure, and any execution engine and function specific attributes in order to specify extra requirements;
- The class CONDITION represents the conditions that should be verified in order to execute the semantic bridge. Condition is operationally similar to transformation in the sense that it must specify all the extra requirements for the function that test the conditions. Because any semantic bridge may have a condition, it allows to control complex transformations according to both the schema and instances data, specially in combination with SemanticBridgeAlt and the Composition constructs.
- The COMPOSITION modelling primitive identified above is supported by the hasBridge relation in the SEMANTICBRIDGE class. It has no cardinality limit nor type constraint which allows any semantic bridge to aggregate many different bridges. Those semantic bridges are then called one by one, and processed in the context of the former.
- The ALTERNATIVE modelling primitive is supported by the SemanticBridgeAlt class. It groups several mutual exclusive semantic bridges. The execution parser checks each of the bridges condition rules and the first bridge which conditions hold is executed while the others are discarded.

In the following, we will describe how the semantic bridging ontology has been represented so it may be used within Semantic Web applications.

SBO represented in DAML+OIL. DAML+OIL [8] has been choosen to represent the semantic bridge ontology. DAML+OIL builds on and extends RDF-Schema and provides a formal semantics for it. One of the goals in specifying the semantic bridge ontology was to maintain and exploit the existent constructs and minimize extra constructs, which would maximize as much as possible the acceptance and understanding by general Semantic Web tools. The SBO ontology is available online at http://kaon.semanticweb.org/2002/04/SBO.daml.

4.3 Example

Let us consider Figure 3 where a small part of two different ontologies are represented. The ontology on the left side (o1) describes the structure of royal families and associated individuals. These concepts are combined with events, both individual events (birth date and death date) and families events (marriages and divorces). The ontology on the right side (o2), characterizes individuals using a very simple approach. It is mainly restricted in representing if the individual is either a Man or a Woman. Unlike o1 that extensively enumerates marriages and divorces, o2 is concerned just with the number of marriages. The goal of this example is to specify a mapping between the source and target ontology (o1 and o2 respectively), using the developed semantic bridge ontology

[8] http://www.daml.org/2001/03/daml+oil-index.html

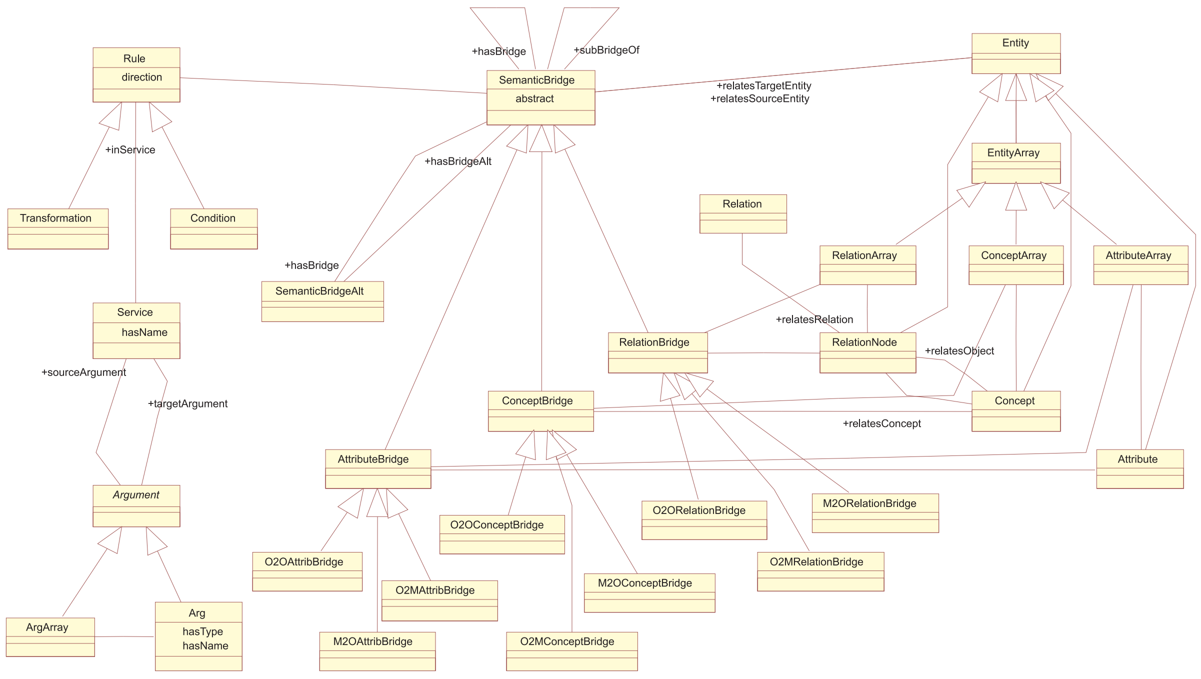


Fig. 2. Bridging Ontology view in UML

(SBO). In order to exploit the SBO potentialities, the mapping specification follows the structure of the ontologies being mapped, normally in the form of a taxonomy. Therefore, a mapping structure represented according to SBO tends to arrange bridges in a hierarchical way.

First, the mapping must define the two ontologies being mapped. Additionally, one may specify top-level semantic bridges which serve as entry points for the translation, even if there are not mandatory. In this case the translation engine starts executing the ”Individual-Individual” bridge.

```
<Mapping rdf:ID="mapping">
    <relatesSourceOntology rdf:resource="&o1;"/>
    <relatesTargetOntology rdf:resource="&o2;"/>
    <hasBridge rdf:resource="#Individual-Individual"/>
</Mapping>
```

Notice that the target ontology intends to create instances of either ”o2:Woman” or ”o2:Man”, but not ”o2:Individual”. In object oriented terminology ”o2:Individual” class is said to be abstract. It is therefore required to state that this concept bridge should not be used to create instances, but serve just as support to sub bridges, like it happens in object oriented paradigm. SBO uses the abstract property in these circumstances. If no abstract property is specified or if it is set to FALSE, then the concept bridge is considered as non-abstract.

It is now necessary to set the alternative between ”o1:Individual” and either ”o2:Woman” or ”o2:Man”. This situation is specified by a SemanticBridgeAlt. In this case the alternatives are two ConceptBridge’s: ”Individual-Woman” and ”Individual-Man”. Bridges may be numerically ordered which can useful if the last bridge has no specified condition. Both rdf:_n like syntax and the one presented are allowed to specify the order.

```
<SemanticBridgeAlt rdf:ID="ManOrWoman">
    <hasBridge>
        <Seq ordinal="1">
            <bridge rdf:resource="#Individual-Woman"/>
        </Seq>
    </hasBridge>
    <hasBridge>
        <Seq ordinal="2">
            <bridge rdf:resource="#Individual-Man"/>
        </Seq>
    </hasBridge>
</SemanticBridgeAlt>
```

The alternative ConceptBridge’s are presented next: ”Individual-Woman” and ”Individual-Man”.

```
<ConceptBridge rdf:ID="Individual-Woman">
    <subBridgeOf rdf:resource="#Individual-Individual"/>
    <relatesSourceEntity rdf:resource="#Individual"/>
    <relatesTargetEntity rdf:resource="#Woman"/>
    <whenVerifiedCondition rdf:resource="#isFemale"/>
</ConceptBridge>

<ConceptBridge rdf:ID="Individual-Man">
    <subBridgeOf rdf:resource="#Individual-Individual"/>
    <relatesSourceEntity rdf:resource="#Individual"/>
    <relatesTargetEntity rdf:resource="#Man"/>
</ConceptBridge>
```

Both bridges rely on the ”Individual-Individual” bridge to translate ”o2:Man” and ”o2:Woman” inherited attributes from ”o2:Individual”. Hence, both are specified as sub-bridges of ”Individual-Individual” concept bridge. Additionally, ”Individual-Woman” concept bridge specifies the whenVerifiedCondition property to ”isFemale”. As remarked bellow, this condition is responsible to test if the individual is of feminine sex. If the condition is verified the bridge is executed. Otherwise, and because the condition is tested in the context of a SemanticBridgeAlt, the next concept bridge in the alternative is processed. The next concept bridge in the alternative is ”Individual-Man” which has no associated condition, and therefore it is unconditionally executed.

Respecting the translation process, consider that an ”o1:Individual” instance is to be translated. The translation engine seeks for bridges relating ”o1:Individual” to any o2 entity. Three are found, but one of them is abstract and is therefore rejected. The other two are both defined in the context of a SemanticBridgeAlt. The SemanticBridgeAlt choosing/exclusion process starts. One of the bridges (or eventually none if none of the associated conditions is verified) is selected. The concept bridge must then create a tar-

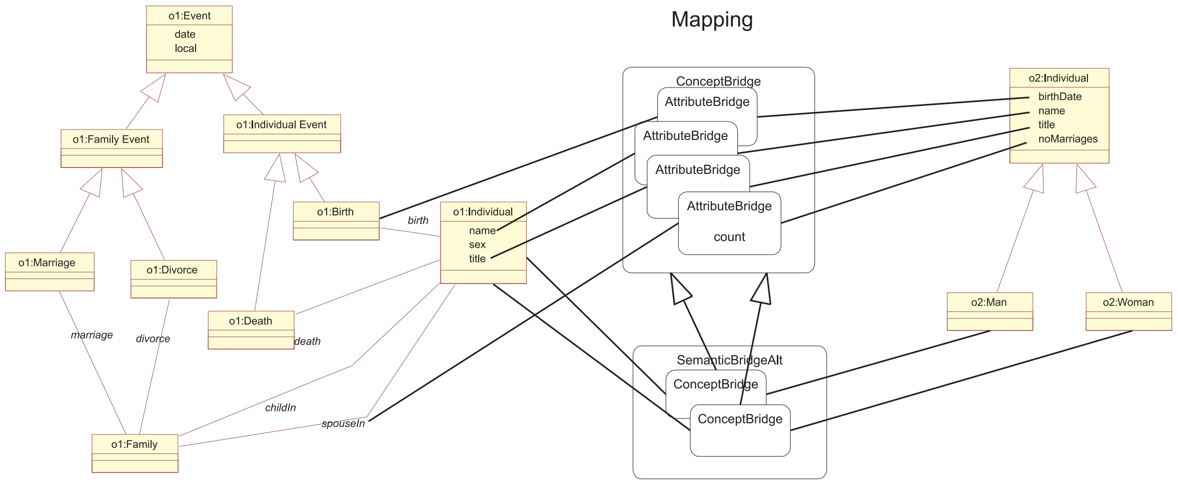


Fig. 3. UML representation of two small ontologies

get instance which will serve as context for complementary bridges.

Complementary attribute bridges are in this example simple 1:1 attribute bridges, relating one attribute from o1 to an attribute in o2, through the associated transformation.

```
<AttributeBridge rdf:ID="name-name">
    <relatesSourceEntity rdf:resource="#name"/>
    <relatesTargetEntity rdf:resource="#name"/>
    <accordingToTransformation rdf:resource="#copyName"/>
</AttributeBridge>

<Transformation rdf:ID="copyName">
    <mapSourceArgument>
        <MapArg>
            <from rdf:resource="#name"/>
            <to>sourceString</to>
        </MapArg>
    </mapSourceArgument>
    <mapTargetArgument>
        <MapArg>
            <from>targetString</from>
            <to rdf:resource="#name"/>
        </MapArg>
    </mapTargetArgument>
    <inService>CopyString</inService>
</Transformation>
```

The ”name-name” attribute bridge for example, bridges ”o2:Individual.name” to ”o2:Individual.name”. The associated transformation in this bridge has the responsibility to copy/create the attribute and assign it to the concept instance. Remember that the concept instance has been created by the concept bridge previously.

Concerning the transformation, it intends to map between the bridge entities and the transformation service arguments. This mapping specification varies according to the service be requested, either in type, cardinality and used tags. For example, the ”copyName” transformation specifies the ”CopyString” service to be called. This service expects to receive a source argument called ”sourceString” and the output is named ”targetString”. The transformation maps ”sourceString” with the attribute ”o1:Individual.name” and ”targetString” to the ”o2:Individual.name”. ”title-title” attribute bridge is very similar to the previous and is not be presented.

In contrast, ”marriages” attribute bridges for example, are slightly different from previous ones. Notice that the source entity is not an attribute but a relation to another concept. Normally an AttributeBridge would not be correctly applied. However, since this is a very common mapping pattern the translation engine allows to process the relation as an attribute. That could eventually be a problem if the translation service expects an attribute. However, the ”CountRelations” service expects a relation which is the case of ”spouseIn” and therefore no problem occurs. A similar situation occurs with ”birth-birth” AttributeBridge. Once again there is no problem because the source entity is accepted as an attribute and the rest is up to the transformation and its associated service.

```
<AttributeBridge rdf:ID="mariages">
    <relatesSourceEntity rdf:resource="#spouseIn"/>
    <relatesTargetEntity rdf:resource="#noMariages"/>
    <accordingToTransformation rdf:resource="#countSpouses"/>
</AttributeBridge>

<Transformation rdf:ID="countSpouses"> <putServiceArgument>
        <MapArg>
            <from>relation</from>
            <to rdf:resource="#spouseIn"/>
        </MapArg>
    </putServiceArgument>
    <mapTargetArgument>
        <MapArg>
            <from>count</from>
            <to rdf:resource="#noMariages"/>
        </MapArg>
    </mapTargetArgument>
    <inService>CountRelations</inService>
</Transformation>

<AttributeBridge rdf:ID="birth-birthDate">
    <relatesSourceEntity rdf:resource="#birth"/>
    <relatesTargetEntity rdf:resource="#birthDate"/>
    <accordingToTransformation rdf:resource="#Birth"/>
</AttributeBridge>


<Transformation rdf:ID="Birth">
    <putServiceArgument>
        <MapArg>
            <from>1</from>
            <to rdf:resource="#birth"/>
        </MapArg>
    </putServiceArgument>
    <putServiceArgument>
        <MapArg>
            <from>2</from>
            <to rdf:resource="#date"/>
        </MapArg>
    </putServiceArgument>
    <mapTargetArgument>
        <MapArg>
            <from>targetString</from>
            <to rdf:resource="#birthDate"/>
        </MapArg>
    </mapTargetArgument>
    <inService>RoyalDate</inService>
</Transformation>
```

Finally, the ”isFemale” condition is considered. This condition is responsible to verify if an instance of an individual is of feminine sex. In this case the pattern refers to the fact that the value of sex attribute has value ”F”. Normally, the services applied in a condition return a boolean value. However, this constraint would depend on the translation engine once it is possible to create a table of correspondences between boolean types and other types. For example, it would be reasonable to consider a true result if the service returns a set of entities or false if it return a empty set.

```
<Condition rdf:ID="isFemale">
```

```
    <putServiceArgument>
        <MapArg>
            <from>1</from>
            <to rdf:resource="#sex"/>
        </MapArg>
    </putServiceArgument>
    <putServiceArgument>
        <MapArg>
            <from>pattern</from>
            <to>F</to>
        </MapArg>
    </putServiceArgument>
    <inService>CascadeAndMatch</inService>
</Condition>
```

5 Implementation

MAFRA is currently under development within the KAON Ontology and Semantic Web Framework[9]. KAON offers a framework and a common set of tools for realizing scalable and reliable ontology-enabled Semantic Web applications. The architecture underlying KAON is depicted in Figure 4, with elements split into three layers as described next.

– The **Application and Services Layer** contains components providing interface to KAON. Human agents typically use one of user interface applications realized within OntoMat - a UI framework for ontology applications offering easy integration and interoperability of different tools for managing ontologies and metadata. KAON-Portal is a framework for the generation of Web portals from ontology-based data. Interoperability with non-Java platforms is realized through a Web-service interface. Furthermore, machine agents use the Web Service interface to access KAON methods and functionality.

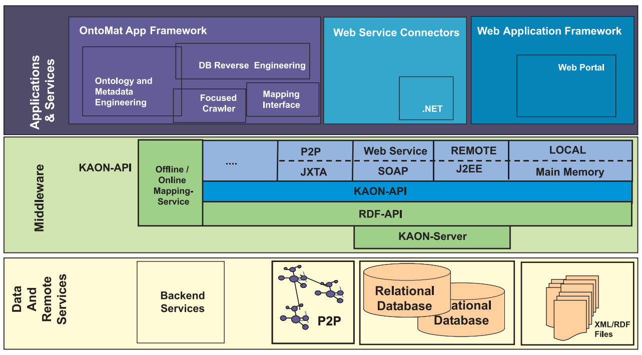
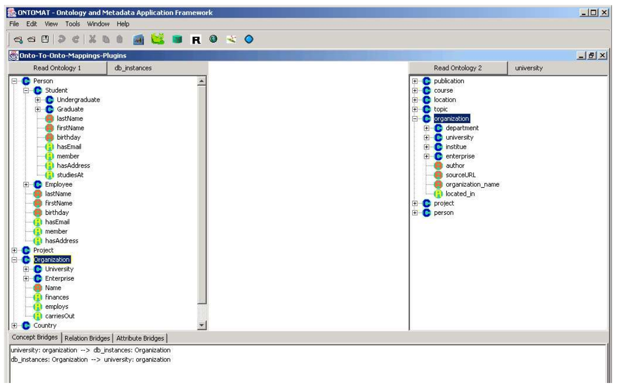


Fig. 4. KAON Architecture

– The **Middleware Layer** is focused around KAON API, providing primitives for ontology access and manipulation via different means. The middleware layer also includes a remote, J2EE-based implementation of KAON-API which allows to work multiple user on the same mapping task. Mapping execution is realized within this layer.

– The **Data and Remote Services Layer** is the back-end part of KAON. For local, in-memory operation storage based on a modified version of RDF API may be used. As mentioned above for enabling scalable and concurrent applications, KAON RDF Server is realized within the J2EE framework. Atomicity of updates is ensured by using transactions.

As specified in section 4, mapping is specified and represented as an instance of a bridging ontology. Therefore, mappings can be created using OntoMat-SOEP – a tool for ontology and metadata management. However to simplify the task of establishing mappings, a plug-in for OntoMat has been implemented. A screen-shot of the user interface for mapping specification is presented in Figure 5. In this example two ontologies have been opened side by side, and in between an instance of the semantic bridging ontology is created using a simplified user interface.



university: organization --> db_instances: Organization
db_instances: Organization --> university: organization

Fig. 5. Creating Mappings Using KAON Tools

As mentioned earlier mapping execution is implemented within KAON API – the focal point of the KAON architecture. KAON API models the domain of ontology applications, providing classes such as Concept, Relation, Instance etc. and means for their creation and manipulation. Consequently, instances of the semantic bridging ontology can be expressed using KAON API constructs and processed and stored in the desired storage systems available within KAON.

The KAON mapping service supports the mapping execution phase under two distinct modes of operation:

– **Offline** (static) execution, transforming source ontology instances into target ontology instances once, without later synchronization,

– **Online** (dynamic) execution, where a connection between source and target ontology instances is constantly maintained.

Execution in either of the two modes is currently developed on the basis of the run-time structure model depicted in Figure 6, taking the upper half of the KAON mapping service box for offline execution and the lower half for online execution.

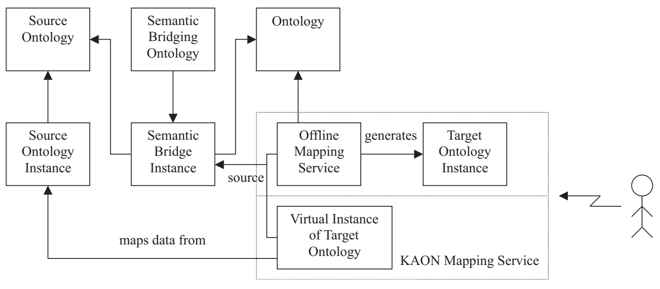


Fig. 6. Mapping Run-time Structure

Offline execution is supported as a batch process. To execute, a previously generated instance of the semantic bridging ontology and an instance of the ontology to be transformed are passed to the offline mapping service. It can then

[9] http://kaon.semanticweb.org

perform the execution and generate an instance of the target ontology by applying transformations from the semantic bridging ontology. The offline mapping service as been implemented in Java.

Online execution is more complex, since the connection between instances of the source and target ontology is constantly maintained - notifications of changes to source ontology instance are mapped to changes in the target ontology instance and propagated to the user. The user has no means to detect that mapping is going on. To achieve this, as in the offline case, first an instance of semantic bridging ontology must be created. It and the source ontology instance are used to create a virtual instance of the target ontology handling online mapping. Execution occurs dynamically - e.g., when the user queries for all instances of a concept of the target ontology, the query is mapped into a query for the source ontology instance and executed there. Upon execution, the list of all instances obtained is then mapped into all instances of the target ontology and reported to the user.

6 Related Work

Much research has been done in the area of information integration. Existing information integration systems and approaches (e.g., TSIMMIS [6], Information Manifold [8], Infomaster[10], MOMIS[11], Xyleme [12]) are “centralized” systems of mediation between users and distributed data sources, which exploit mappings between a single mediated schema and schemas of data sources. Those mappings are typically modeled as views (over the mediated schema in the local-as-view approach, or over the sources schemas in the global-as-view approach) which are expressed using languages having a formal semantics. For scaling up to the Web, the “centralized” approach of mediation is probably not flexible enough, and distributed systems of mediation are more appropriate.

Furthermore, mapping approaches can mainly be distinguished along the following three categories: discovery, [14, 3, 5, 10, 1], mapping representation [9, 1, 11, 13] and execution [4, 11]. However, none of the proposed solutions has really encompassed the overall mapping process specially considering the evolution and consensus building of semantic bridges. Having this in mind, we have introduced the Ontology MApping FRAmework (MAFRA) as a basis for managing and executing mapping between distributed ontologies in the Semantic Web. Within MAFRA we provide an approach and conceptual framework that provides a generic view and figure onto the overall mapping process. In this paper we have set a specific focus on the semantic bridging phase corresponding to the mapping representation category. The approaches which resemble our approach more closely are [13] and [12]. Basically, our work has been motivated by the work done in [13], where an ontology has been specified for the translation between the domain-knowledge-base components and problem-solving-method components. The approach that comes nearest to ours has been described in [12]. They describe an approach for integrating vocabularies including means for mapping discovery and representing mappings with a focus on B2B applications (product catalogues) has been described. In contrast to our work, the RDFT ontology describes a set of core bridges to *(i)* lift XML tags to the RDF model and *(ii)* to define bridges between RDF(S) classes and properties and to *(iii)* translate transformation results back to XML. In the paper [12] it remains unclear, how execution specific information in the form of the constraint and transformation dimension is attached to the bridges. Furthermore, it is also not discussed if the overall process is executed statically or dynamically, where we offer both solutions.

[10] http://infomaster.stanford.edu/infomaster-info.html
[11] http://sparc20.ing.unimo.it/Momis/
[12] http://www.xyleme.com

7 Conclusion and Future Work

Ontologies may used for achieving a common consensus within a user community about conceptualizing, structuring and sharing domain knowledge. Based on the application scenario provided by Ontologging we have motivated that it is unrealistic to assume that one single ontology for different communities of users is realistic in real-world applications. We argue that decentralization has been one of the key elements for the scalability of the World Wide Web and its underlying applications. In order to balance the autonomy of each community with the need for interoperability, mapping mechanisms between ontologies have been proposed. In this paper we presented the Ontology Mapping Framework (MAFRA) supporting the interactive, incremental and dynamic ontology mapping process in the context of the Semantic Web. In this paper a specific focus has been set on the semantic bridging phase where we have provided a detailed description of a semantic bridge meta-ontology, that is instantiated when mapping between two domain ontologies.

In the future much work remains to be done. First, depending on the domain ontologies, data sources, application scenarios, user participation, capabilities and other factors further semantic bridges may be necessary. For example, procedural mechanisms may complement the taxonomy of semantic bridges. Thus, we consider the semantic bridging ontology as evolving. Second, considering the mapping process as a consensus building process of two communities, we will on the basis of our technological infrastructure KAON, perform an experiment how multi-user mapping may be efficiently supported. Third, we will develop an integrated LIFT tool that allows to lift several existing data representations including relational databases, XML-Schema, DTDs onto the same data model. Executing a dynamic mapping process keeping the autonomy of the different input data will be a challenging task.

Acknowledgements. Research for this paper was financed by European Commission, IST, project ”Ontologging” (IST-2000-28293) and by Marie Curie Fellowship on Semantic Web Technologies. Special thanks to Gabor Nagypal for fruitful discussions on defining the semantic bridging ontology and Oliver Fodor for stimulating discussions on the lift component and cooperative mapping. Thanks to the students Frank Westerhausen and Zoltan Varady who did the implementation work for the graphical user interface and the static transformation engine.

References

1. S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration of heterogeneous information sources. In *Special Issue on Intelligent Information Integration, Data & Knowledge Engineering*, volume 36, pages 215–249. Elsevier Science B.V., 2001.
2. T. Berners-Lee. *Weaving the Web*. Harper, San Francisco, 1999.
3. W. Cohen. The whirl approach to data integration. *IEEE Intelligent Systems*, pages 1320–1324, 1998.
4. T. Critchlow, M. Ganesh, and R. Musick. Automatic generation of warehouse mediators using an ontology engine. In *Proceedings of the 5 th International Workshop on Knowledge Representation meets Databases (KRDB'98)*, 1998.
5. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the World-Wide Web Conference (WWW-2002)*, 2002.
6. J. Hammer, H. Garcia-Molina, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. Information Translation, Mediation, and Mosaic-Based Browsing in the TSIMMIS System. In *Exhibits Program of the Proceedings of the ACM SIGMOD International Conference on Management of Data, page 483, San Jose, California, June 1995.*, 1995.
7. S. Khoshafian and G. Copeland. Object identity. In *Proceedings of the 1st ACM OOPSLA conference, Portland, Oregon, September 1986.*, 1985.
8. Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proceedings of VLDB-96, 1996*, 1996.
9. J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proceedings of the 27th International Conferences on Very Large Databases*, pages 49–58, 2001.
10. A. Maedche and S. Staab. Measuring similarity between ontologies. In *Technical Report, E0448, University of Karlsruhe*, 2001.
11. P. Mitra, G. Wiederhold, and M. Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Proceedings of Conference on Extending Database Technology (EDBT 2000)*. Konstanz, Germany, 2000.
12. B. Omelayenko. Integrating Vocabularies: Discovering and Representing Vocabulary Maps. In *Proceedings of the First International Semantic Web Conference (ISWC-2002), Sardinia, Italy, June 9-12, 2002.*, 2002.
13. J. Y. Park, J. H. Gennari, and M. A. Musen. Mappings for reuse in knowledge-based systems. In *Technical Report, SMI-97-0697, Stanford University*, 1997.
14. E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
15. M.C. Rousset. Standardization of a web ontology language. *IEEE Intelligent Systems, March/April 2002*, 2002.
16. P.R.S. Visser, D.M. Jones, T.J.M. Bench-Capon, and M.J.R. Shave. An analysis of ontology mismatches: Heterogeneity versus interoperability. In *AAAI 1997 Spring Symposium on Ontological Engineering, Stanford CA., USA*, pages 164–72, 1997.