

OWL Ontology Translation for the Semantic Web

Luís Mota and Luís Botelho *We, the Body and the Mind* Research Lab
ADETTI/ISCTE
Av. das Forças Armadas, 1649-026 Lisboa, Portugal
luis.mota@iscte.pt,luis.botelho@we-b-mind.org

Abstract

This paper presents an ontology translation Web Service based on the O_3F ontology representation framework that translates between ontologies expressed in OWL. In the O_3F framework, ontology translation is based on a mapping from the basic concepts of the object ontology to basic or compound concepts of the target ontology. Using publicly available mappings, it is possible to recursively translate compound concepts of the object ontology into the target ontology.

We describe the algorithms used by and the interface of the Ontology Translation Web Service. The FIPA Ontology Service specification[4] was used as a starting point for the definition of relationships between ontologies.

Finally, the paper presents the definition of two ontologies in OWL, and the source of mappings from the object ontology into the target ontology. In an hypothetical scenario, two programs using different ontologies interact with the ontology translation web service in order to obtain interchangeability between information that is heterogeneously organised. The translation web service was implemented using Tomcat¹, which implements the Servlet 2.3 specification[13, v2.3].

Keywords: Ontology translation, semantic interoperability, Web Service

1. Introduction

The key issue in the Semantic Web is the creation of a new Web of content, where information is annotated in a semantically organised way, using well defined ontologies. However, information creators will certainly use a wide set of different ontologies, according to their needs, background and convenience. It will therefore be possible to annotate the same kind of information in very different ways. Without dedicated support, the use of different ontologies

for shared application domains impairs information interoperability, compromising the access to the Semantic Web.

In order to overcome that difficulty, several authors have proposed methodologies and repositories for ontology management and access [12, 6, 10]. These initiatives did not, however, address the question of ontology translation or integration.

Ontology translation assumes an essential role to overcome this interoperability problem. Using ontology translation web services, it is possible for a program (or user) P_1 using ontology O_1 to communicate with another program P_2 using a distinct ontology O_2 . One possible strategy is for P_1 to ask an ontology translation service to translate the expressions of ontology O_1 to expressions of ontology O_2 and then share the translated expressions with P_2 . P_2 would get the information in a known ontology, being therefore able to process it.

This paper presents our approach for OWL ontology translation in the scope of the O_3F ontology representation framework [9], which has been improved in [2]. The O_3F framework represents the ontologies using an object-oriented model, and introduces definitions for different levels of ontology translatability. O_3F defines translation conditions for object oriented ontologies composed of classes, properties, attributes, facets, methods and arbitrary axioms capturing relationships among the entities of the domain. Namely, the translation of methods involves, among other tasks, the verification that the translated axioms describing the method of the object ontology logically imply the axioms defining the corresponding method in the target ontology. This is a hard and generally not decidable problem which we will not address for the time being.

In terms of ontology representation, recent years have seen the rapid development of ontology representation methods using languages with XML syntax, and several efforts have been proposed [7, 3]. These efforts have at a given point led to the Web Ontology Language (OWL), which became a W3C proposal and, more recently, a recommendation[11]. OWL is a description logic based language and has had wide dissemination

¹ <http://jakarta.apache.org/tomcat/>

because of the W3C support, backed up by a large enterprise and scientific community supporting its development.

In its nature, O_3F has been conceived as a very expressive framework, powerful enough to allow the lossless conversion of ontologies represented in other frame or object based paradigms (e.g. OWL and Ontolingua) to O_3F ontologies. This high expressivity has, though, the disadvantage of making it undecidable. The implementation of an ontology translation service for full O_3F ontologies is therefore a difficult task that can only be accomplished in the long term.

As a step towards this long term goal, we tackled the question of translating OWL ontologies, which are less expressive than O_3F . These translation tasks are performed by a Tomcat-based web service. To determine translatability, this service uses mapping information between basic concepts, available from results of other research projects.

In section 2 we briefly describe the O_3F framework and, in section 3, its application to OWL. Section 4 presents the ontology translation process, its implementation details and the ontology service description. A detailed usage scenario showing the interoperability enhancement made possible by the implemented service is included in section 5. Finally, in sections 6 and 7 we present related work and draw some conclusions.

2. The O_3F Framework

The *Object-Oriented Ontology Framework* (O_3F) [9] has two major components: an ontology definition model and a set of definitions pertaining the relationships between ontologies.

2.1. Object Oriented Model

The concepts in an ontology can be modelled through the usual primitives of object oriented models, namely datatypes, classes, attributes, facets and methods. To allow a more detailed modelling of ontologies, O_3F also permits the writing of arbitrary axioms. These axioms, written in first order logic, can be used with three different purposes: to define the result of functional methods, to determine the behaviour, preconditions and consequences of action methods, and to capture aspects of the ontology that cannot be represented through the object-oriented model.

The model also considers the concept of Property. Properties are not common in object oriented models, but were included in O_3F to simplify the conversion of ontologies defined in languages that use properties, like OWL and RDF, to O_3F .

2.2. O_3F Ontology Relationship Framework

The FIPA Ontology Service Specification [4] introduced a framework to classify the relation between two ontologies. This relation can have different levels, e.g., *weakly-translatable* and *equivalent*. This classification presents significant shortages: it is not formally defined and applies to the whole ontology, overlooking the relation between the individual concepts in the ontologies.

In O_3F , the determination of relationships between ontologies is made through a recursive approach: an ontology O_1 is translatable to O_2 if the classes in O_1 are translatable to O_2 . Further, a class C_1 is only translatable to a class C_2 in another ontology if the attributes, facets and methods of C_1 are translatable to attributes, facets and methods of C_2 . The determination of the relation level between ontologies is therefore grounded in the relations between concepts defined in the ontologies.

The determination of relations between ontologies is built on top of a name mapping from the basic vocabulary of the object ontology to the vocabulary of the target ontology. This mapping in itself does not ensure any kind of translatability. It is therefore a necessary but not sufficient condition of translatability.

Datatypes, properties, attributes, facets, methods and arguments in different ontologies are considered *translatable* if there is a name mapping between them and the conditions in the corresponding translatability definition are met. Classes in two ontologies can be *approx-translatable*, *weakly-translatable*, *strongly-translatable*, *equivalent* or *identical*. The translation level of ontologies can have the values defined in [4].

3. Ontology translation definitions for OWL

The general translatability rules defined in the O_3F framework can be applied to ontologies defined in OWL. OWL has, however, a lower expressivity, since it does not allow the definition of attributes, methods or arbitrary axioms. O_3F 's translatability definitions can therefore be substantially simplified when applied to the OWL model. The remainder of this section shows these adapted definitions. It should be noted that the definitions are made on basis of sufficient requirements: if they are met, all instances of the complying properties and classes are guaranteed to be translatable.

3.1. Vocabulary mapping

The application of the translation definitions is, as said before, dependent on the existence of basic name mappings between concepts defined in different ontologies. In the case of OWL ontologies, some research efforts have been devel-

oped with the goal of mapping discovery and extraction. The application of the translation definitions to the results of these efforts is shown and explained in section 4.

3.2. Relations between classes and properties

Definition 1: Datatype Property DP_1 in ontology O_1 is translatable to datatype property DP_2 in ontology O_2 , with respect to the *basic vocabulary mapping* I_{O_1,O_2} iff the name of DP_1 is mapped to the name of DP_2 by I_{O_1,O_2} , the class in the domain of DP_1 is translatable to the class in the domain of DP_2 with respect to I_{O_1,O_2} and the datatype in the range of DP_1 is compatible with the datatype of DP_2 . A datatype D_1 is said to be compatible with datatype D_2 iff all the possible values of D_1 are also valid values of D_2 .

Definition 2: Object Property OP_1 in ontology O_1 is translatable to object property OP_2 in ontology O_2 , with respect to the *basic vocabulary mapping* I_{O_1,O_2} iff the name of OP_1 is mapped to the name of OP_2 by I_{O_1,O_2} , the class in the domain of OP_1 is translatable to the class in the domain of OP_2 with respect to I_{O_1,O_2} and the class in the range of OP_1 is translatable to the class in the range of OP_2 with respect to I_{O_1,O_2} .

Definition 3: A Restriction R_1 in ontology O_1 is translatable to restriction R_2 in ontology O_2 , with respect to the *basic vocabulary mapping* I_{O_1,O_2} iff the property P_1 on which R_1 is applied is translatable to the property P_2 on which R_2 is applied. Further requirements apply to specific kinds of restrictions. If R_1 is a minimum cardinality restriction, then the value of R_2 is required to be less or equal to the value of R_1 . A maximum cardinality restriction is similarly defined, whereas in cardinality restrictions both restrictions must have the same value. In *AllValuesFrom* and *SomeValuesFrom* restrictions, the restricted range must be translatable. Similarly, in *HasValue* restrictions the value must be translatable.

In the next definition, a property P is said to be a *declared property* of class C if C is the domain of P .

Definition 4: A class C_1 from ontology O_1 is *strongly-translatable* to class C_2 from ontology O_2 with respect to the mapping I_{O_1,O_2} iff the name of C_1 is mapped to the name of C_2 by I_{O_1,O_2} , for each declared property $P_{1,i}$ of C_1 there is a declared property $P_{2,j}$ of C_2 such that $P_{1,i}$ is translatable to $P_{2,j}$ with respect to I_{O_1,O_2} , for each restriction $R_{2,m}$ of C_2 there is a restriction $R_{1,n}$ of C_1 such that $R_{1,n}$ is translatable to $R_{2,m}$ with respect to I_{O_1,O_2} and for each restriction $R_{1,o}$ of C_1 there is a restriction $R_{2,p}$ of C_2 such that $R_{1,o}$ is translatable to $R_{2,p}$ with respect to I_{O_1,O_2} .

Definition 5: A class C_1 from ontology O_1 is *equivalent* to the class C_2 from ontology O_2 with respect to a *basic vocabulary mapping* I_{O_1,O_2} , iff C_1 is *strongly-translatable* to C_2 with respect to I_{O_1,O_2} , and C_2 is *strongly-translatable* to C_1 with respect to I_{O_1,O_2} .

Definition 5: A class C_1 from ontology O_1 is *identical* to the class C_2 from ontology O_2 with respect to *basic vocabulary mapping* I_{O_1,O_2} iff C_1 is *equivalent* to C_2 with respect to I_{O_1,O_2} and they share the same *extended vocabulary*. Two classes share the same extended vocabulary iff all property and class names involved in the two definitions are identical.

Definition 6: A class C_1 from ontology O_1 is *weakly-translatable* with respect to the *basic vocabulary mapping* I_{O_1,O_2} iff the class name of C_1 is mapped to the class name of C_2 by I_{O_1,O_2} and if for each restriction $R_{2,i}$ of C_2 there is a restriction $R_{1,j}$ of C_1 such that $R_{1,j}$ is translatable to $R_{2,i}$ with respect to I_{O_1,O_2} .

Definition 8: A class C_1 from ontology O_1 is *approximately-translatable* to class C_2 from ontology O_2 with respect to the *basic vocabulary mapping* I_{O_1,O_2} , iff the class name of C_1 is mapped to the class name of C_2 by I_{O_1,O_2} . Classes that are approximately translatable have thus a similar semantic intention, but have an incompatible ontological design.

3.3. Relations between Ontologies

Ontologies can have six different levels of relationship with respect to translation: *strongly-translatable*, *extension, equivalent*, *identical*, *weakly-translatable* and *approximately-translatable*. The definitions presented in [9] depend solely on the translation level of classes, and thus can be directly applied to OWL ontologies, considered the definitions presented in the previous section.

4. Ontology Translation Process

The OWL ontology translation process is made of two major tasks: the verification of the translatability of concepts defined in the ontology and the translation of expressions complying to the ontology.

The determination of translatability levels depends on the verification of the before-mentioned definitions, which rely on the existence of basic name mappings, among other conditions. There is, therefore, the need to obtain a source of such mappings. Webscripiter[5] is a system for the creation of Semantic Web-based reports, integrating data structures using different DAML+OIL and OWL ontologies. This system helps users making reports of heterogeneous sources in the Semantic Web. Simultaneously, as Webscripiter users identify concepts as being similar, the system stores the mapping information between classes and properties. Centrally gathering the information from multiple users, the system is able to collect trustworthy mappings. The goal of this system is thus to find mappings, something O_3F needs for its operation. The presently implemented translation ser-

vice uses Webscripser mappings from concepts in the object ontology to concepts in the target ontology to determine the translation level. This usage allows the creation of an autonomous translation web service which does not need human intervention to create mappings between ontologies and simply reuses the results of other projects. The implemented service is, however, not limited to the exploitation of results of the Webscripser project: it could easily use mappings from a different source, if the format in which they are stored is known.

The translation level determination task was accomplished by closely following the relevant translatability definitions presented in the previous section. The Jena2² toolkit was employed to parse RDF and OWL files. An added set of functions checks the existence of applicable name mappings and the fulfilment of other translation conditions.

The translation web service can receive and translate RDF files, provided that the expressions in the file are translatable into the target ontology. The parser extracts all the statements from the file and translates them individually. If one of these statements is not translatable, the translation of the file is considered unfeasible and accordingly refused.

4.1. Ontology Translation Web Service

An ontology translation web service was implemented using the described algorithm. This service resides on a Tomcat v5.5 server, and is accessible through the HTTP POST method. The service can not be accessed through the GET method because its arguments are URI's, which raise problems when sent as GET arguments. This service can easily be described using WSDL

There are two different requests that can be made to this service: determination of translation level between two concepts and translation of RDF expressions. These requests should have the following arguments and results:

- **TranslationLevel**(URI ?from, URI ?to, string ?mappingName): computes the level of the translation from the concept (ontology or class) identified by ?from to the concept identified by ?to using the mapping whose name is ?mappingName. Returns the determined translation level as a string.
- **TranslateRDF**(String ?expression, URI ?fromOnt, URI ?toOnt, String ?mappingName): translates the RDF expressions (ontologies, classes or properties) defined in the full RDF file included in ?expression from the ontology in ?fromOnt to the ontology in ?toOnt, using the mapping whose name is ?mappingName. Returns the translated RDF expression.

This service can easily be described using WSDL. As an example, for the first request, the definition would be as follows:

```
<definitions name="Translation"
  <message name="TranslationLevelInput">
    <part name="fromURI" type="...URI"/>
    <part name="toURI" type="...URI"/>
    <part name="mappingName"
      type="xsd:string"/>
  </message>

  <message name="TranslationLevelOutput">
    <part name="translationLevel"
      type="xsd:string"/>
  </message>

  <portType name="TranslationLevelPT">
    <operation name="TranslationLevel">
      <input message="...LevelInput"/>
      <output message="...LevelOutput"/>
    </operation>
  </portType>

  <binding name="TranslationLevelB"
    type="TranslationLevelPT">
    <http:binding verb="POST"/>
    <operation name="TranslationLevel">
      <http:operation
        location="TranslationLevel"/>
      <input>
        <mime:content
          type="
            ...x-www-form"/>
      </input>
      <output>
        <mime:content
          type="text/plain"/>
      </output>
    </operation>
  </binding>

  <service name="TranslationService">
    <port name="TranslationLevelPort"
      binding="tns:TranslationLevelB">
      <http:address
        location="http://sth.com/" />
    </port>
  </service>
</definitions>
```

2 <http://jena.sourceforge.net>

5. Translation Web Service Usage Scenario

This section presents a scenario with examples showing the level of interoperability achieved as the result of the interaction with the implemented OWL ontology translation web service.

In this scenario, an event reminder service, called *reminder*, wants to send SMS messages to the mobile phones of all the participants in a conference, alerting them to the eminence of the starting date of this event. In terms of scientific conferences, *reminder* knows the 'eBiquity' ontology³, of which the relevant excerpt is shown.

```
<owl:Class rdf:ID="Conference">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
        rdf:resource="#title"/>
      <owl:cardinality>1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
        rdf:resource="#startDate"/>
      <owl:maxCardinality>1
      </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
        rdf:resource="#endDate"/>
      <owl:maxCardinality>1
      </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="title">
  <rdfs:domain
    rdf:resource="#Conference"/>
  <rdfs:range
    rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="startDate">
  <rdfs:domain
    rdf:resource="#Conference"/>
  <rdfs:range
    rdf:resource="&xsd:dateTime"/>
</owl:DatatypeProperty>
```

3 adapted from <http://ebiquity.umbc.edu/v2.1/ontology/conference.owl>

```
<owl:DatatypeProperty rdf:ID="endDate">
  <rdfs:domain
    rdf:resource="#Conference"/>
  <rdfs:range
    rdf:resource="&xsd:dateTime"/>
</owl:DatatypeProperty>
```

This ontology is, however, not known by the software installed in the phones of some of the intended receivers. One of the intended receivers, called *Frank*, has a personal event manager software that only knows the concept of conference as defined in a different ontology, called 'mindswap'⁴. The conference concept is defined as follows.

```
<owl:Class rdf:ID="Event"/>

<owl:DatatypeProperty
  rdf:ID="hasStartDate">
  <rdfs:domain rdf:resource="#Event"/>
  <rdfs:range
    rdf:resource="&xsd:dateTime"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Conference">
  <rdfs:subClassOf
    rdf:resource="#Event"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
        rdf:resource="#hasTitle"/>
      <owl:maxCardinality>1
      </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
        rdf:resource="#hasStartDate"/>
      <owl:maxCardinality>1
      </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty
  rdf:ID="hasTitle">
  <rdfs:domain rdf:resource="#Conference"/>
  <rdfs:range
    rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
```

4 adapted from <http://www.mindswap.org/2004/www04photo.owl>

Since the two programs do not have a shared ontology, the communication must be mediated by the ontology translation service. As a first step, *reminder* must know what ontologies are used by *Frank*'s software.

At this point, *reminder* needs to find if the 'eBiquity' ontology is translatable to one of *Frank*'s ontologies. Consequently, *reminder* sends messages to the translation web service querying the translation level between the 'eBiquity' ontology and each of *Frank*'s ontologies. In order to query the translation level between the 'eBiquity' and 'mindswap' ontologies, identified by their respective URI, with respect to the 'webscripeter' mapping, a POST request should be sent, according to the WSDL previous definition, to the address `http://sth.com/TranslationLevel`, with the three necessary arguments:

- `http://ebiquity.(...)#`
- `http://mindswap.(...)#`
- `webscripeter`

The ontology translation service will send a reply to this request, indicating that the translation level to the 'mindswap' ontology is *approx-translatable*, since some classes in the object ontology (some not shown in the previous ontology excerpt) are translatable to the target ontology. Since *reminder* only needs to use the conference class, it sends another message asking the translation level between these classes in the two ontologies, using 'webscripeter' as the source of the name mapping. This would require a new POST, very similar to the previous and sent to the same address, requesting the determination of the translation level of the *Conference* class in the 'eBiquity' ontology to the *Conference* class in the 'mindswap' ontology. This request would have the following arguments:

- `http://ebiquity.(...)#Conference`
- `http://mindswap.(...)#Conference`
- `webscripeter`

After receiving this request, the ontology translation service determines the translation level by following these steps:

1. check that Webscripeter provides a mapping between the two classes, which is true since some Webscripeter users have used both classes in compatible roles and sent this mapping information to the central repository;
2. check that it is possible to translate restrictions on the object class to all the restrictions in the target class: the restriction on the *hasTitle* property is translatable from

the *title* property and the restriction on the *hasStartDate* property is translatable from *startDate*. This allows the translation service to classify the translation as *weakly-translatable*;

3. check if it is possible to translate all the restrictions in the object ontology to restrictions in the target ontology. The restriction on the *endDate* property is not translatable, since there is no compatible restriction in the target class. The object class is therefore not *strongly-translatable* and the translation level determination can stop at this point.

The translation service replies thus to *reminder* with the determined translation level. Since the translation of an instance of the object class is, according to the definition of *weakly-translatable* classes, a valid instance of the target ontology, *reminder* decides to request the translation of the expression it intends to send. This is made through a new POST request sent to `http://sth.com/TranslationLevel` with the following arguments:

- ```
<rdf:RDF (...)>
 <ebq:Conference>
 <ebq:title>WWW2005
 </ebq:title>
 <ebq:startDate>05.10.05
</ebq:startDate>
 <ebq:endDate>05.14.05
</ebq:endDate>
</ebq:Conference>
</rdf:RDF>
```
- `http://ebiquity.(...)#`
- `http://mindswap.(...)#`
- `webscripeter`

The translation service translates this RDF expression and sends the translation back to *reminder*.

```
<rdf:RDF (...)>
 <msw:Conference>
 <msw:hasTitle>AAMAS05</msw:hasTitle>
 <msw:hasStartDate>07.25.05
 </msw:hasStartDate>
</msw:Conference>
</rdf:RDF>
```

The translated expression does not include the end date of the conference, since there is no property modelling this information in the target ontology. The translation expression is, however, still valid according to the 'mindswap' ontology. Upon receiving this translation, *reminder* can use the expression to send a message to *Frank*, which will be understood by the latter's software, because it is written according to a known ontology.

## 6. Related Work

Ontology translation and merging is a popular research topic in the (Distributed) Information Systems community. This community needs this kind of translation in order to integrate data resident in different repositories and organised with different schema. One typical effort in using agent systems and ontologies to overcome this problem can be found in [8], where broker agents use ontology translation information to interact with agents that use different ontologies. In this case, the mapping of high level concepts is pre-existent, entirely man-made and allows only one relationship level: equivalence. It would be thus more suitably called a framework for usage of user-driven ontology integration.

An initiative closely motivated by open agent systems can be found in [1]. The authors propose a technique of "Approximate Ontology Translation", that relies, like the previous one, on a human-made concept mapping between ontologies. The mappings can have different levels, e.g. equivalent or generalisation. During query reformulation, concepts can be substituted by their equivalent or generalised counterparts. The authors claim that these substitutions can be quantitatively evaluated by closeness metrics, which can unfortunately only be estimated. This kind of approach can thus be helpful in domains where rigour is not essential, but can have no general application.

These two systems differ greatly from  $O_3F$ : they are built on top of predefined and complete mappings between ontologies, while, in  $O_3F$ , inter-ontology relationships are derived from simple name mapping and the comparison of each concept's set of attributes, methods and relations with other concepts in its ontology.

## 7. Conclusions and Future Work

We have defined an algorithm for OWL ontology translation based on the  $O_3F$  framework, which defines rigorous criteria for determining translation relations between ontologies: approximately-translatable, weakly-translatable, strongly-translatable, equivalent and identical as defined in the FIPA ontology specifications. However, contrarily to what happens with FIPA specs, the  $O_3F$  definitions allow deducing the translation level of ontologies from the translation level of their basic entities. Based on the  $O_3F$  framework, the algorithm for ontology translation was implemented through a Web Service that answers to two kinds of requests: determination of translation levels and actual translation across ontologies of RDF expressions. The ontology translation service is fully implemented in the JAVA programming language, using the Jena2 toolkit and the Tomcat Web Service server.

To the best of our knowledge, ours is the only implemented web service performing autonomous ontology translation, actually improving interoperability in the Semantic Web. This web service uses basic mapping information which is available from the results of other research projects. It can therefore operate without human definition of the mappings, and it can integrate and use new mapping information as it becomes available.

Currently, the implemented ontology translation service is limited to OWL ontologies and does not handle the full  $O_3F$  ontology model. Steps will be made to come closer to this goal in the near future.

One possible future concern is the validity of the basic mappings used by the ontology translation service, which presently uses Webscripiter as the only mapping source. These mappings are generally considered good, but they have undergone no formal process of validation. This means that some of the mappings may originate from reports sent by different and numerous users, while other mappings can have been reported by a single user and consequently have no general validity. All mappings could be subject to a statistical trust analysis, where the mappings with higher number of positive reports would have a higher trust level. The translation service could subsequently consider the trust level and set a trust threshold under which a mapping would be discarded, or considered a *weak* mapping, as defined in [9].

## References

- [1] J. Akahani, K. Hiramatsu, and K. Kogore. Approximate ontology translation and its application to regional information services. In *1st International Semantic Web Conference (ISWC2002)*, 2002.
- [2] L. M. Botelho and P. Ramos. *CO<sub>3</sub>L: Compact O<sub>3</sub>F language*. In *Workshop on Ontologies in Agent Systems, AAMAS 03 - Autonomous Agents and Multi Agent Systems*, July 2003.
- [3] DARPA. *DAML+OIL (March 2001) ontology markup language*, <http://www.daml.org/2001/03/daml+oil.daml>. Defense Advanced Research Projects Agency, 2001.
- [4] Foundation for Intelligent Physical Agents. *FIPA Ontology Service Specification*, <http://www.fipa.org/specs/fipa00086/>, 2001.
- [5] M. Frank, P. Szekely, R. Neches, B. Yan, and J. Lopez. Webscripiter: World-wide grass-roots ontology translation via implicit end-user alignment. In *WWW-2002 Semantic Web Workshop*, 2002.
- [6] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
- [7] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. OIL: The Ontology Inference Layer. *IEEE Intelligent Systems*, 15:69–72, December 2000.

- [8] H. Kitajima, R. Masuoka, and F. Maruyama. Integrating information and knowledge with software agents. *Fujitsu Sci. Tech. Journal*, 36(2):162–174, December 2000.
- [9] L. Mota, L. M. Botelho, H. Mendes, and A. Lopes. *O<sub>3</sub>F*: an object oriented ontology framework. In *AAMAS 03 - Autonomous Agents and Multi Agent Systems*, July 2003.
- [10] J. Pan and S. Cranefield. A lightweight ontology repository. In *AAMAS 03 - Autonomous Agents and Multi Agent Systems*, July 2003.
- [11] M. K. Smith, C. Welty, and D. L. McGuinness. *OWL Web Ontology Language Guide*, 2004.
- [12] H. Suguri, E. Kodama, M. Miyazaki, H. Nunokawa, and S. Noguchi. Implementation of FIPA ontology service. In *Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents*, May 2001.
- [13] Sun Microsystems. *Java Servlet Technology*, <http://java.sun.com/products/servlet/>.