Ontology Agents in FIPA-compliant Platforms: a Survey and a New Proposal

Daniela Briola
DISI, Univ. di Genova,
Via Dodecaneso 35, 16146, Ge, IT
E-mail: daniela.briola@unige.it

Angela Locoro
DIBE, Univ. di Genova,
Via Opera Pia 11/A, 16146, Ge, IT
E-mail: angela.locoro@unige.it

Viviana Mascardi
DISI, Univ. di Genova,
Via Dodecaneso 35, 16146, Ge, IT
E-mail: viviana.mascardi@unige.it

Abstract—**In 2001, FIPA delivered a specification suggesting that each MAS should integrate an "Ontology Agent" (OA) offering services for ontology management. These services should include ontology discovery, maintenance, matching, as well as translation of expressions between different ontologies or content languages. Currently, no FIPA-compliant OA exists that implements all of them. One of the reasons is that providing a service for ontology matching is not an easy task, and coping with translation between ontologies and/or content languages may be even harder. In this paper we survey the state of the art in the area, and we describe our prototypical implementation of an OA for Jade able to match ontologies. Besides "standard" ontology matching algorithms, our OA offers a "matching via upper ontologies" method that, as we showed in a recent technical report, improves the precision of the matching w.r.t. the use of traditional techniques.**

I. Introduction

In 2001, FIPA delivered a specification for organizing and managing ontologies in a MAS [7]. This specification suggests that each MAS should integrate an "Ontology Agent" (OA) providing services to deal with ontologies. The OA should be at the same conceptual level as the Directory Facilitator Agent, and should be able to deal with ontologies explicitly represented in some ontology language, and stored somewhere (perhaps in some server), where agents can access, query, and in case update them.

In particular, an OA should offer the following services to the agents in the MAS:

1) discover public ontologies in order to access them,
2) maintain (for example, register with the Directory Facilitator, upload, download, and modify) a set of public ontologies,
3) translate expressions between different ontologies and/or different content languages,
4) answer queries about relationships between terms or between ontologies,
5) facilitate the identification of a shared ontology for communication between two agents.

It's not mandatory for an OA to be able to realize all these services, provided that it is able to answer that it cannot process the required service.

The FIPA specification assumes that each ontology accessible through the OA's services adheres to the OKBC model [19]. OKBC supports an object-oriented representation of knowledge and provides a set of representational constructs commonly found in object-oriented knowledge representation systems. This standard is widely used there, but differs from the standards commonly accepted in the semantic web area, where OWL [27] is the most widespread model. The RDF, RDFS, and OWL languages are represented as a graph of triples <*object, property, subject*>. The semantics of OWL is based on Description Logic [2]. Even when the modelling primitives look similar to OKBC, the semantics is different, thus converting the operations associated to an OWL ontology into OKBC operations is not feasible. The lack of support to OWL ontologies is one of the main limitations of the FIPA OA specification, and motivates the lack of fully implemented FIPA-compliant OAs.

Another reason why no OAs exist that implement *all* the services suggested by the FIPA specification is that "*answering queries about relationships between terms or between ontologies*", as the specification suggests, is definitely an hard task if we consider *semantic* relationships, and not just *structural* ones. Answering a query on the structure of ontologies, such as "Is concept $c_1 \in o$ a subconcept of $c_2 \in o$ (or even a subconcept of $c' \in o'$)?" is almost trivial; answering a query on the semantics of terms, such as "What is the confidence in $c \in o$ and $c' \in o'$ having the same meaning?" is much more difficult. This activity requires that the OA is able to "match" ontologies o and o', namely, it is able to compute an "ontology alignment" between them.

Many algorithms for ontology matching exist, and some of them have been implemented and are available to the research community[1]. Surprisingly, none of them has been integrated into an OA. Thus, to the best of our knowledge, no existing OAs offer services for ontology matching.

In this paper we describe our implementation of an OA for Jade, able to provide both "standard" and new ontology matching services, as well as services for comparing one or more alignments to a reference one. In particular, our OA offers services for ontology matching via upper ontologies, a new approach that has proven to give good results in precision

[1]For example, the Alignment API developed by J. Euzenat and his team at INRIA-Rhône Alpes is available at http://alignapi.gforge.inria.fr/ under GNU Lesser General Public License. It provides string-based and simple language-based ontology matching methods, as well as methods for computing precision and recall of a given alignment w.r.t. a reference one.

and recall [16].

Our OA is far from implementing all the services suggested by the FIPA specification as, at the time of writing, it only offers the matching one. Since most of the existing FIPA-compliant OAs provide services for ontology discovery and interrogation, whereas none of them provides a service for ontology matching, we started our research by implementing the latter, in order to complement existing proposals. As other researchers, we deviate from the FIPA specification since we assume that ontologies are represented in OWL instead than OKBC.

The paper is organized as follows: Section II discusses the state of the art of FIPA-compliant OAs, after providing a short background on ontology matching and upper ontologies. Section III describes the functionalities of our OA, briefly presents the algorithms for matching OWL ontologies, discusses the implementation of a simple MAS consisting of a Request Agent and one OA, and shows experimental results. Finally, Section IV concludes and highlights future improvements of our work.

II. Background

A. Ontology Matching

A formalization of the ontology matching process can be found in [6]. Quoting the authors, we define a matching process as *"a function f which takes two ontologies o and o', an input alignment a, a set of parameters p and a set of oracles and resources r, and returns an alignment a' between o and o'"*.

A correspondence (also named "mapping") between an entity e belonging to ontology o and an entity e' belonging to ontology o' is a 5-tuple $< id, e, e', R, conf >$ where:

- id is a unique identifier of the correspondence;
- e and e' are the entities (e.g. properties, classes, individuals) of o and o' respectively;
- R is a relation such as "equivalence", "subsumption", "disjointness", "overlapping", holding between the entities e and e';
- $conf$ is a confidence measure (typically in the [0;1] range) holding for the correspondence between the entities e and e'.

An alignment of ontologies o and o' is a set of correspondences between entities of o and o'.

Among the matching techniques, we just discuss those that fall under the "Granularity / Input Interpretation" classification described in [6], based on the granularity of the matcher and on the interpretation of the input information.

- *String-based methods.* These methods measure the similarity of two entities just looking at the strings (seen as mere sequences of characters) that label them. Among them we may cite substring distance, where two strings are compared to find the longest common substring, n-gram distance [4], where two strings are the more similar the more n-grams (sequence of n characters) they have in common, and SMOA measure [24], which is a function of the commonalities (in terms of substrings) as well as of differences between two strings.
- *Language-based methods.* These methods exploit natural language processing techniques to find the similarity between two strings seen as meaningful pieces of text rather than sequences of characters. Some of them exploit external resources like WordNet, and exploit the semantic relations that it offers to compute the correspondences.

B. Upper Ontologies and their Application to Ontology Matching

An upper ontology (also named top-level ontology, or foundation ontology) is "*an attempt to create an ontology which describes very general concepts that are the same across all domains*" [28]. Few upper ontologies exist: BFO [10], Cyc [13], DOLCE [9], GFO [11], PROTON [5], Sowa's ontology [23], and SUMO [17]. They vary in dimension, ranging from the 30 classes of Sowa's ontology to the 300,000 of Cyc, in representation language (OWL, KIF [1], First Order Logic), in structure (monolithic vs. decomposed into modules), and in developed applications. Nevertheless, all of them describe general concepts (also named "classes") and share the aim to have a large number of ontologies accessible under them.

In our previous work, we implemented different algorithms that used upper ontologies for boosting the ontology matching process [16]. We run experiments with SUMO-OWL (a restricted version of SUMO translated into OWL), OpenCyc (the open version of Cyc, which is a commercial ontology), and DOLCE. The experiments demonstrate that when the "structural matching method via upper ontology" uses an upper ontology large enough (OpenCyc, SUMO-OWL), the recall is significantly improved and the precision is kept w.r.t. not using upper ontologies. Instead, the "non structural matching method" via OpenCyc and SUMO-OWL improves the precision and keeps the recall. The "mixed method", that combines the results of structural alignment without using upper ontologies and structural alignment via upper ontologies, improves the recall and keeps (improves, with OpenCyc) the F-measure, whatever the upper ontology used.

C. FIPA-compliant OAs: the State-of-the-Art

The FIPA reference model for the services provided by the OA is shown in Figure 1.

In the literature there have been few attempts to realize the FIPA OA, and each one adopts a particular point of view of the problem.

Some solutions implement only a subset of the OA's services, others change the FIPA specification in order to use the OWL specification language, others realize a Web Service playing the OA role. Besides being different from the design point of view, these solutions also differ in the choice of the middleware where the OA is integrated, and hence of the language used to implement it.

In the following sections we discuss the most relevant solutions for the integration of the OA in a FIPA compliant framework.

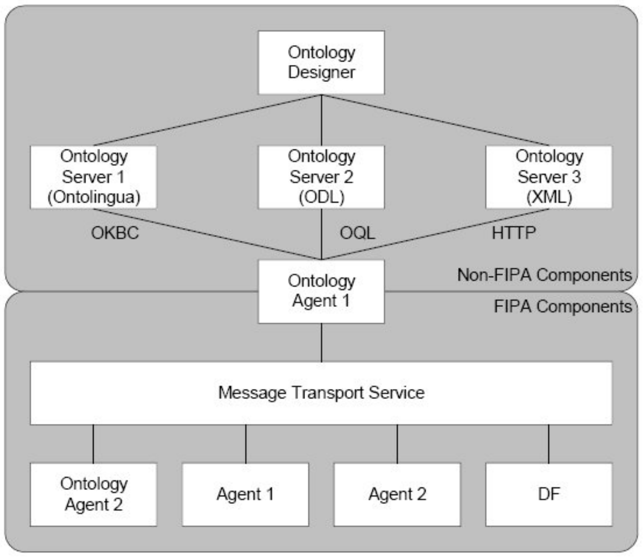

Figure 1. FIPA Ontology Service Reference Model

Implementation over the COMTEC platform

The first attempt to realize an OA was made in 2001 by Suguri, Kodama, Miyazaki [25]. They realized an OA for the COMTEC platform.

The OA is divided into two parts. The first part is an interface to the OKBC front-end. From the OKBC point of view, the OA is one of the front-end user applications. The second part is the FIPA interface where the agent wrapper is implemented. The FIPA interface is an agent wrapper that takes care of generating and interpreting SL [8] actions, predicates and ACL communicative acts based on appropriate interaction protocols. It also processes the registration with the DF, the management of ontology names and the relationship between the ontologies.

The COMTEC OA implements a subset of the services of a generic FIPA-compliant OA, in particular

1) register an ontology in the framework;
2) operate over an ontology (create, delete frames, slot, modify the hierarchies);
3) answer queries about ontologies's structure, and their level of similarity.

No ontology matching service is provided by this OA.

From a design point of view, this solution is one of the best because it tries to faithfully adhere to the FIPA specification. Its main problem is that the COMTEC platform on which it was implemented is no longer available.

Implementation over the AgentService platform

Vecchiola, Grosso, and Boccalatte implemented a FIPA compliant framework called AgentService [26], based on the .NET platform. Together with Passadore, they integrated an OA into AgentService [20].

Ontologies in AgentService are represented in OKBC: the implementation of the OA is thus fully compliant with the FIPA specification.

The services that the OA offers are a subset of the possible ones: the OA implements the discovery and publication of the ontology and its maintenance, allows two agents to check whether they use the same ontologies and if not, it helps them to download the “missing” ones. However, neither ontology matching nor translation are supported.

AgentService uses Protégé [22] to support the designer from the creation of the ontology to the development of an agent that can communicate using that ontology, and MS Visio (http://office.microsoft.com/en-us/visio/default.aspx) to design agent interaction protocols. Visio allows the designer to import an ontology and supports him/her in the creation of message contents compliant with concepts expressed in the ontology itself.

Thus, the main usefulness of having ontologies integrated into AgentService is to support developers in designing message exchange in a clear and well-founded way. This makes the design easily sharable among developers. AgentService OA has been conceived for managing ontologies within closed MASs, rather than for implementing open systems where heterogeneous/unknown agents interact and choose an ontology on the fly.

Implementation over the Jade platform

One of the most used FIPA-compliant platform is Jade [3]. This framework is fully compliant to the FIPA standards, and is based on Java. Jade offers some utilities to model ontologies, but in the spirit of integrating the ontology into the agent's code. Thus, Jade supports the hard coding of the ontology both at design and compile time: all the entities of the ontology have to be transformed in classes and objects, in order to allow a Jade agent to use them. This approach gives little help to a MAS developer who wants to create agents that can communicate with others automatically, after having agreed on an ontology known and available to all of them. Then, Jade lacks a real support to the use of ontologies in open MASs.

We have implemented an OA in Jade offering services for matching OWL ontologies using different algorithms, and for evaluating the result of the matching. Its functionalities are described in Section III.

The only other attempt of integrating a FIPA-compliant OA into Jade we are aware of, is that by Obitko and Snášel [18]. Their implementation follows the FIPA specification but adapts it to ontologies represented in OWL, as we do.

Since Obitko and Snášel intended to store OWL ontologies only, they had to adapt the language for describing actions performed by the ontology agent. Their OA agent exploits Jena [12] and implements the basic functionalities of the ontology services as specified in the FIPA proposal, i.e. the possibility of modifying ontologies (assert and retract) and of querying ontologies using RDQL.

Obitko and Snášel's OA is well organized and closely follows the FIPA specification except for the usage of OWL instead to OKBC.

Non FIPA-compliant solutions

Ontology Services as the result of Distributed Cooperation: A good effort to design and implement a MAS with a support to ontology matching comes from Li, Wu and Yang [14], [15]. Their work concentrates on the process of mapping and integrating ontologies: these functionalities are integrated in the MAS thanks to a set of agents which collaborate to offer them to the other agents. The agents which are involved in the delivery of ontology services are:

1) User Agent (UA): assists the user in formulating his/her requests, posts queries (e.g. tasks) to the proposed system via the IA and visualises the required results according to the user's requirement. The UA only knows the IA.
2) Interface Agent (IA): acts as an interface among agents in the MAS and the UA. Every agent knows the IA.
3) Ontology Agent (OA): acts on behalf of the corresponding ontology, is in charge of ontology related tasks. It provides as much information of the ontology it acts on as possible. The OA operates over the ontology structure and the mapping result file. When a new ontology is loaded in the system, a corresponding OA is created to manage it.
4) Mapping Agent (MA): maps, if possible, concepts from an ontology to the concepts of a second ontology, using also the SA.
5) Similarity Agent (SA): maintains a thesaurus for the purpose of similarity. It holds a list of common words and synonyms of words.
6) Query Agent (QA): operates over the mapping results to investigate ontology-understandable of heterogeneous ontologies after executing ontology mapping.
7) Integration Agent (InA): merges two ontology in a new one (in RDF format). Is based on the result of ontology mapping.
8) Checking Agent (CA): checks the consistency of the integrated ontology (assuming all given ontology are consistent at the beginning).

The purposes of this system, namely providing a large set of ontology services that include ontology matching ones, make it very close to our proposal. However, the way these services are implemented make the system really far from the FIPA OA specification, since services are distributed among different agents, and are not integrated within a FIPA-compliant framework.

An OA implemented as a Web Service: In [21] Peña, Sossa and Gutierrez implement the OA as a web service, in order to offer its services also over the Internet. The OA carries out the management of the ontologies through an interface between the application agents and the ontologies. It is responsible for catching the requests arriving from the agents, interpreting them, forwarding them to the Ontology Manager in charge of the Ontology referenced in the request, and forward back the response from the Ontology manager. Ontologies are in OWL format, and each Ontology Manager answers only to requests about the structure of the ontology or for changing its structure, but no support is offered to the mapping, translating or more complex queries about two ontologies.

III. Our OA in Jade

The OA we have implemented in Jade provides the following services:

1) matching two OWL ontologies through a direct matching;
2) matching two OWL ontologies via an upper ontology (represented in OWL too);
3) evaluating an alignment against a reference alignment.

The implementation of a fourth service, namely the repair of an alignment based on word sense disambiguation techniques, is under way. We do not discuss it here for space constraints.

In this section, we describe the algorithms that realize all the implemented services of the OA. It is worth specifying that in our matching algorithms we only consider concepts as entities to match, and equivalence as relation.

In order to be compliant with the Align API mentioned in Section I, and to be able to exploit some of the methods it provides, our alignments are represented in RDF and their format is like the one shown below.

```
<rdf:RDF .. namespaces here ...>
<Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type>**</type>
  <time>15</time>
  <method>StringDistAlignment</method>
  <onto1>file:///ka.owl</onto1>
  <onto2>file:///edumit.owl</onto2>
  <uri1>file:///ka.owl</uri1>
  <uri2>file:///edumit.owl</uri2>
  ...
  <map> <Cell>
    <entity1 rdf:res=ka.owl#Book/>
    <entity2 rdf:res=edumit.owl#Book/>
    <relation>=</relation>
    measure>1.0</measure>
  </Cell> </map>
  <map> <Cell>
    <entity1 rdf:res=
       ka.owl#TechnicalReport/>
    <entity2 rdf:res=
         edumit.owl#Techreport/>
    <relation>=</relation>
    <measure>1.0</measure>
  </Cell> </map>
  ...
</Alignment>
</rdf:RDF>
```

With respect to the definition provided in Section II-A, our correspondences are simpler due to the lack of the correspondence identifier, which is strictly necessary only in case we need to uniquely identify them (e.g. when storing them in a repository).

Direct Matching

The direct matching service is implemented by a function that we named $parallel\_match(o, o', \{WordNet\}, th)$, because it runs in parallel different "standard" matching methods, and combines their outputs. The matching methods that we used are the *substring*, *n-gram*, *SMOA* and *WordNet* ones that we introduced in Section II-A. We used the implementation of these methods offered by Euzenat's Align API[2]. The th parameter is used as a threshold for cutting all correspondences below it. In our experiments, we set it at 0.5. To obtain a final alignment from the ones obtained by the individual matching methods, we use an $aggregate$ function that composes the four alignments by making the union of all their correspondences. In case different alignments produced correspondences between the same concepts c and c', $aggregate$ keeps the correspondence with the highest confidence measure.

Matching via Upper Ontology

For matching two ontologies o and o' via an upper ontology uo we compute the $parallel\_match(o, uo, \{WordNet\}, th)$ and $parallel\_match(o', uo, \{WordNet\}, th)$, obtaining two alignments between o and uo, and o' and uo respectively. These two alignments are given in input to a $merge(a, a')$ function. $Merge$ produces the final alignment between o and o' by combining the correspondences of o-uo and o'-uo in such a way that: if $\exists$ a correspondence $< c, c_u, r, conf1 >$ in o-uo and $\exists$ a correspondence $< c', c_u, r, conf2 >$ in o'-uo, then the $merge$ function creates a new correspondence $< c, c', r, conf1 * conf2 >$ and adds it to the final alignment.

Alignment Evaluation

The Alignment API provided by Euzenat and colleagues offers a *PrecEval* method for measuring the goodness of an alignment based on precision, recall and F-measure. Precision is the number of correctly found correspondences with respect to the reference alignment (true positives), divided by the total number of found correspondences (true positives and false positives), and recall is the number of correctly found correspondences (true positives) divided by the total number of expected correspondences (true positives and false negatives). F-measure is the harmonic mean of precision and recall.

We have integrated the *PrecEval* method into our OA; an example of evaluation produced by the OA is given by the following RDF fragment:

```
<rdf:RDF ..namespaces here ..>
  <map:output rdf:about=''>
    <map:in1 rdf:resource="ka.owl"/>
    <map:in2 rdf:resource="edumit.owl"/>
    <map:precision>0.097</map:precision>
    <map:recall>0.084</map:recall>
    <map:fMeasure>0.09</map:fMeasure>
    <map:nbcorrect>7</map:nbcorrect>
   <map:nbfound>72</map:nbfound>
  </map:output>
</rdf:RDF>
```

[2]In particular, we used the Alignment API version 3.1. The methods we used are *StringDistAlignment* that provides `subStringDistance`, `ngramDistance` and `smoaDistance` string metrics, and *JWNLAlignment* that computes a substring distance between two concepts exploiting their WordNet 3.0 synsets.

Implementation and Experiments

In order to test the behaviour of our OA we implemented a Request Agent (RA) that acts as an interface between the user and the OA, and allows the user to request services to the OA. The RA receives a set of parameters from the user, and saves them for later use. These parameters are:

- the URI of the OWL file containing the first ontology to match, o;
- the URI of the OWL file containing the second ontology to match, o';
- the URI of the file where the computed alignment will be stored;
- the URI of the file containing the reference alignment for performing an evaluation of the computed alignment;
- the matching method ("direct matching" or "matching via upper ontology");
- a further optional parameter providing the URI of the file containing the upper ontology in OWL. This parameter is needed only in case of matching via upper ontology.

The system architecture is depicted in Figure 2.

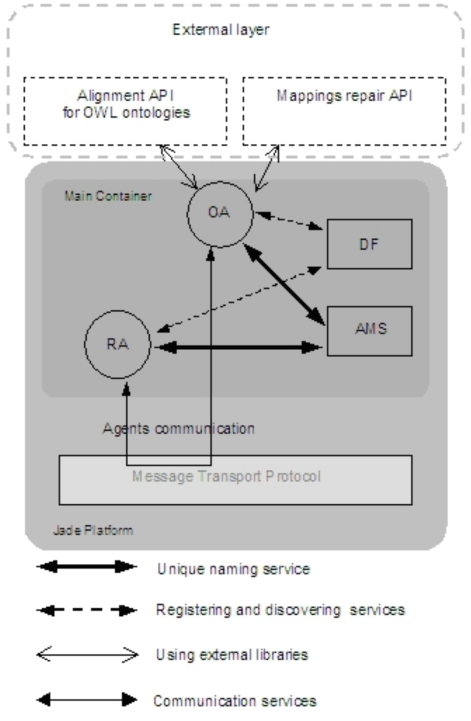

Figure 2. Architecture of the Ontology Agent integrated into Jade.

During its start up phase, RA searches and identifies an OA inside the Jade Platform.

After having received the parameters from the human user, and having found OA, RA:

- sends as many INFORM messages to the OA, as the parameters it wants to send to OA; the parameters are sent as the message content (one parameter for each message); the conversation id of the message is used to keep track of the conversation just started;
- sends a REQUEST message asking for the matching and evaluation services (which, in this first prototype, are always coupled); the ontologies to match, and the method to use, are those passed in the previous INFORM messages.

OA starts its life with the registration of its ontology services to the DF. When RA begins the interaction phase, OA reacts as follows:

- it receives all the parameters from the RA and sends an ACL Message in reply to each message, setting the same conversation id for correctly keeping track of the conversation, after having successfully received and saved each parameter;
- on reception of a REQUEST message, it runs the required matching method on the ontologies specified in the previous interaction steps with the RA;
- sends an INFORM message to notify the RA of the accomplishment of the matching service and to communicate the alignment file URI and the evaluation file URI; in case something goes wrong, it sends a FAILURE message to the RA.

Both agents terminate after the completion of these activities.

OA and RA have been structured according to Jade recommendations for the development of agents. RA's `setup()` method includes the steps to save parameters from the command line, the instantiation of a `wakerBehaviour` object in order to find the OA via DF query and the call to the `RequestPerformer` behaviour. This one includes the `action()` method where the agent sends the parameters, sends the alignment and evaluation requests, and waits for the results. It has been implemented as a generic `Behaviour` class.

OA's `setup()` method starts with the registration of OA's services to the DF and the instantiation of its two behaviours which are of type `ReceiveParameters` and `Align` respectively. The `action()` method of `ReceiveParameters` is dedicated to receive the parameters and has been implemented as a `OneShotBehaviour` class. The `action()` method of `Align` first waits for an ontology matching request. After the request message has been received, the matching and evaluation activities are performed, and the result is notified to the sender. This class has been modelled extending the generic `Behaviour` class.

To better synchronize the exchange of messages each of them is received by the two agents in blocking mode and the `MessageTemplate` class has also been used to deal with conversation id patterns.

For the development of our MAS consisting of the OA and the RA we used Jade 3.6.

In order to verify the feasibility of our approach, we have run a large set of experiments. We discuss only two of them in details; the other ones have been carried out in a similar way.

In the first experiment, the RA asks the OA to execute a direct alignment between *Ka* (protege.cim3.net/file/pub/ontologies/ka/ka.owl) and *Bibtex* (oaei.ontologymatching.org/2004/Contest/304/onto.rdf). *Ka* has 96 concepts dealing with the academic domain, including Event (Activity, Meeting, Conference, Workshop), Publication (concepts similar to those of the *Bibtex* ontology), Organisation (Department, Enterprise, Institute, University), ResearchTopic, whereas *Bibtex* has only 15 concepts including Article, Book, Conference, Manual, Person, Publisher, etc.

Before using *Ka* and *Bibtex* in our experiments, we preprocessed them by hand in order to remove properties and individuals that we do not use in our matching algorithms. We also built a reference alignment between *Ka* and *Bibtex* by hand because we wanted to evaluate how good the alignments resulting from the automatic matching methods were, w.r.t. the reference one.

We passed the URIs of the two ontologies to match as input parameters to RA together with the URI of the file that had to contain the alignment, the URI of the file that contains the reference alignment, and the chosen method (direct alignment).

The second experiment also aimed at aligning *Ka* and *Bibtex*, but using the "matching via upper ontology" method. The upper ontology we used is SUMO-OWL, a lossy translation into OWL of the SUMO ontology, whose full version is encoded in KIF.

The results of these two experiments are shown below; the table summarizes the total correspondences found w.r.t. the correct ones and the precision, recall and f-measure computed by the *PrecEval* method integrated into our OA.

Method	**Found**	**Correct**	**Prec.**	**Recall**	**F-meas.**
Direct	27	7	0.26	**0.08**	**0.13**
SUMO-OWL	9	6	**0.67**	0.07	**0.13**

While running this experiment, we activated the Jade sniffer agent to follow the interactions between RA and OA. A screenshot of the "sniffed" messages is given in Figure 3. The messages exchanged in the first and second experiment are very similar. In both experiments OA and RA exchange messages to pass and save parameters. When the parameter negotiation phase ends, the RA sends a request to OA for performing the matching and the evaluation. The task has been successfully executed in both experiments. As a consequence OA sends an INFORM message to RA to inform it of the successful outcome of the matching, and both agents terminate.

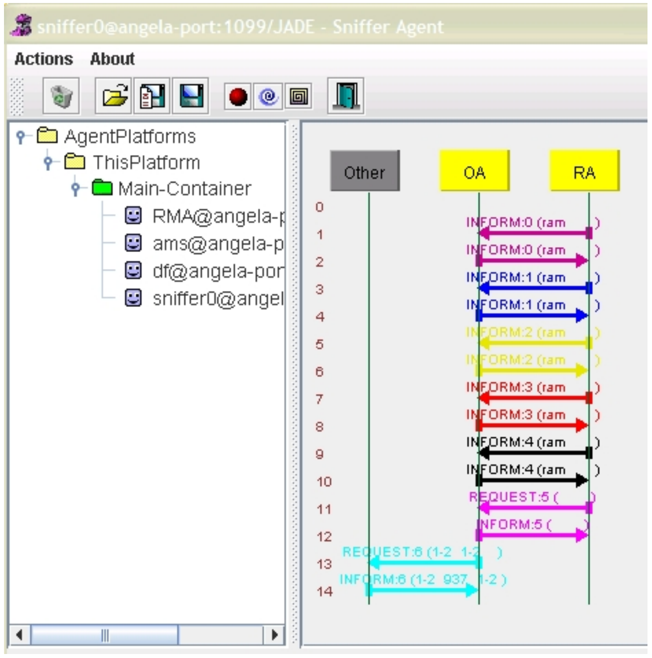

Figure 3. A sniffer screenshot with OA and RA agents.

The first experiment uses a direct matching method that took less than a minute to complete[3], while the second one required about 10 minutes due to the usage of a more sophisticated matching algorithm involving a large upper ontology (SUMO-OWL has 4,393 concepts). From a comparison of the alignments computed by the OA in the two experiments, which are reported in Table III, it turns out that matching *Ka* and *Bibtex* via SUMO-OWL gives the best precision (67% against 26% of the direct alignment), while recall and F-measure are the same (recall: 7% against 8%; F-measure: 13% for both).

We have run 9 more tests. Besides *Ka* and *Bibtex*, the ontologies that we have used in our experiments and the URLs from where they may be downloaded are:

- *Agent*, 212.119.9.180/Ontologies/0.2/agent.owl
- *Biosphere*, sweet.jpl.nasa.gov/ontology/biosphere.owl
- *Ecology*, wow.sfsu.edu/ontology/rich/EcologicalConcepts.owl
- *Food*, silla.dongguk.ac.kr/jena-owl1/food
- *Geofile*, www.daml.org/2001/02/geofile/geofile-ont.daml
- *HL7_RBAC*, lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/HL7_RBAC.owl
- *MPEG7*, dmag.upf.es/ontologies/2003/03/MPEG7Genres.rdfs
- *Restaurant*, guru-games.org/ontologies/restaurant.owl
- *Resume*, statistic.gunadarma.ac.id/research/WorkGroupInformationSystem/DownLoad/onto_colection/resume.owl
- *Space*, 212.119.9.180/Ontologies/0.3/space.owl
- *Subject*, www.library.yale.edu/ontologies/subject.owl
- *Top-bio*, www.co-ode.org/ontologies/basic-bio/top-bio.owl

[3]We run our experiments on a HP Pavillon Notebook with Intel Core Duo T2250 processor, 1.73 GHz of clock, 2 GB of RAM, and Windows XP.

- *Travel*, lsdis.cs.uga.edu/projects/meteor-s/downloads/Lumina/ontology/travelontology.owl
- *Vacation*, www.guru-games.org/ontologies/vacation.owl
- *Vertebrate*, www.co-ode.org/ontologies/basic-bio/basic-vertebrate-gross-anatomy.owl (Vertebrate has been reduced by hand when running our experiments)

The experiments run consisted of matching the following couples of ontologies:

- **1:** Ka - Bibtex;
- **2:** Biosphere - Top-bio;
- **3:** Space - Geofile;
- **4:** Restaurant - Food;
- **5:** MPEG7 - Subject;
- **6:** Travel - Vacation;
- **7:** Resume - Agent;
- **8:** Resume - HL7_RBAC;
- **9:** Ecology - Top-bio;
- **10:** Vertebrate - Top-bio

The obtained results are summarized in Table I. In 7 experiments out of 10, matching via SUMO-OWL allowed us to obtain the best precision. In many cases, the recall of the two methods can be compared even if, in some experiments, the direct matching performs definitely better. The suitability of the "matching via upper ontologies" method strictly depends on the ontologies to match. From our experiments we have learnt that for example, the type of English words used by both upper and matched ontologies counts, as well as the terminology used by both upper and matched ontologies.

IV. Conclusions

This paper describes a FIPA-compliant Ontology Agent integrated in Jade, able to deal with OWL ontologies.

Our OA is able to produce alignments between two OWL ontologies via direct matching or via an OWL version of an upper ontology. It evaluates the obtained alignment with respect to a given reference one and is going to be extended with more "standard" services such as registration, discovery, and maintenance of ontologies. The exploitation of an upper ontology for performing an alignment between two ontologies is a new approach that we only described in a recent technical report. This approach represents an original contribution to the ontology matching process.

For what concerns our decision to match concepts only, and to limit ourselves to considering the equivalence relation, our intention is to extend the matching methods towards the exploitation of properties and individuals, and to take into consideration at least subsumption and disjunction relations.

Our OA can be improved through a deeper analysis of the FIPA-Agent-Management ontology in order to see if it is feasible to join the description of the services provided by OA with concepts of the Jade `BasicOntology`, thus allowing the matching process, the evaluation and the correspondences repair services to be a part of the Jade Agents semantic. Future extensions we would like to carry on are: the modelling of more complex behaviours which would better reflect the

Test	Method	Found	Correct	Prec.	Rec.	F-meas.
1	Manual	83	83	1.00	1.00	1.00
1	Direct	27	7	0.26	**0.08**	**0.13**
1	SUMO-OWL	9	6	**0.67**	0.07	**0.13**
2	Manual	604	604	1.00	1.00	1.00
2	Direct	26	6	**0.23**	**0.01**	**0.02**
2	SUMO-OWL	2	0	0.00	0.00	0.00
3	Manual	513	513	1.00	1.00	1.00
3	Direct	164	38	0.23	**0.07**	**0.11**
3	SUMO-OWL	49	22	**0.45**	0.04	0.08
4	Manual	1041	1041	1.00	1.00	1.00
4	Direct	107	12	0.11	0.01	0.02
4	SUMO-OWL	82	28	**0.34**	**0.03**	**0.05**
5	Manual	637	637	1.00	1.00	1.00
5	Direct	323	94	0.29	**0.15**	0.20
5	SUMO-OWL	224	93	**0.42**	**0.15**	**0.22**
6	Manual	262	262	1.00	1.00	1.00
6	Direct	50	19	**0.38**	**0.07**	**0.12**
6	SUMO-OWL	26	8	0.31	0.03	0.06
7	Manual	1122	1122	1.00	1.00	1.00
7	Direct	157	58	0.37	**0.05**	**0.09**
7	SUMO-OWL	71	31	**0.44**	0.03	0.05
8	Manual	295	295	1.00	1.00	1.00
8	Direct	127	36	0.28	**0.12**	0.17
8	SUMO-OWL	60	34	**0.57**	**0.12**	**0.19**
9	Manual	308	308	1.00	1.00	1.00
9	Direct	88	28	**0.32**	**0.09**	**0.14**
9	SUMO-OWL	140	17	0.12	0.06	0.08
10	Manual	19	19	1.00	1.00	1.00
10	Direct	12	2	0.17	**0.11**	0.13
10	SUMO-OWL	7	2	**0.29**	**0.11**	**0.15**

Table I
COMPLETE RESULTS OF OUR EXPERIMENTS

alignment, evaluation and correspondences repair tasks (i.e. parallel behaviours for string based and language based partial alignments which could be nested inside a sequential behaviour able to produce the final alignment and the merged alignment via an upper ontology), the extension of OA life to prolong the availability of these services for the whole time the platform is running and the exploitation of FIPA standard interaction protocols to manage the conversation between our agents. Separating the alignment process from the evaluation one by providing different requests for alignment, evaluation and correspondences repair, hence exploiting a modular architecture, is a further goal we will pursue.

ACKNOWLEDGMENTS

The authors acknowledge the "Iniziativa Software" CINI-FINMECCANICA project that partially funded this work.

REFERENCES

[1] American National Standard. KIF Knowledge Interchange Format – dpANS NCITS.T2/98-004, 1998.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[3] F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. WILEY, 2007.

[4] E. Brill, S. Dumais, and M. Banko. An analysis of the askmsr question-answering system. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Proceedings*, 2002.

[5] N. Casellas, M. Blázquez, A. Kiryakov, P. Casanovas, M. Poblet, and R. Benjamins. OPJK into PROTON: Legal domain ontology integration into an upper-level ontology. In R. Meersman and et al., editors, *WORM 2005, 3rd International Workshop on Regulatory Ontologies, Proceedings*, volume 3762 of *LNCS*, pages 846–855. Springer, 2005.

[6] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, 2007.

[7] *FIPA Ontology Service Specification*, 2001. http://www.fipa.org/specs/fipa00086/XC00086D.pdf.

[8] *FIPA SL Content Language Specification*, 2002. http://www.fipa.org/specs/fipa00008/SC00008I.html.

[9] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In A. Gómez-Pérez and V. R. Benjamins, editors, *EKAW, 13th International Conference, Proceedings*, volume 2473 of *LNCS*, pages 166–181. Springer, 2002.

[10] P. Grenon, B. Smith, and L. Goldberg. Biodynamic ontology: Applying BFO in the biomedical domain. In D. M. Pisanelli, editor, *Ontologies in Medicine*, volume 102 of *Studies in Health Technology and Informatics*, pages 20–38. IOS Press, 2004.

[11] H. Herre, B. Heller, P. Burek, R. Hoehndorf, F. Loebe, and H. Michalek. General formal ontology (GFO): A foundational ontology integrating objects and processes. part i: Basic principles. Technical report, Research Group Ontologies in Medicine (Onto-Med), University of Leipzig, 2006. Version 1.0, Onto-Med Report Nr. 8, 01.07.2006.

[12] *JENA*. http://jena.sourceforge.net/.

[13] D. Lenat and R. Guha. *Building large knowledge-based systems*. Addison Wesley, 1990.

[14] L. Li. Agent-based ontology management towards interoperability. Master's thesis, Swinburne University of Technology, 2005.

[15] L. Li, B. Wu, and Y. Yang. Agent-based ontology integration for ontology-based application. In *AOW 2005, associated with the 18th CRPIT Conference series by Australian Computer Society, Vol 58*, pages 53–59, 2005.

[16] V. Mascardi, A. Locoro, and P. Rosso. Exploiting DOLCE, SUMO-OWL, and OpenCyc to boost the ontology matching process. Technical Report DISI-TR-08-08, University of Genoa, 2008. http://www.disi.unige.it/person/MascardiV/Software/OntologyMatchingViaUpperOntology.html.

[17] I. Niles and A. Pease. Towards a standard upper ontology. In C. Welty and B. Smith, editors, *FOIS 2001, 2nd International Conference on Formal Ontology in Information Systems, Proceedings*, pages 2–9. ACM Press, 2001.

[18] M. Obitko and V. Snášel. Ontology repository in multi-agent system. In M. H. Hamza, editor, *Artificial Intelligence and Applications, AIA 2004, Proceedings*, 2004.

[19] *OKBC*. http://www.ai.sri.com/~okbc/.

[20] A. Passadore, C. Vecchiola, A. Grosso, and A. Boccalatte. Designing agent interactions with Pericles. In *ONTOSE 2007, 2nd International Workshop*, 2007.

[21] A. Peña, H. Sossa, and F. Gutierrez. Web-services based ontology agent. In *Distributed Frameworks for Multimedia Applications, 2006. The 2nd International Conference on*, pages 1–8, 2006.

[22] *Protégé*. http://protege.stanford.edu/.

[23] J. F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing, 1999.

[24] G. Stoilos, G. B. Stamou, and S. D. Kollias. A string metric for ontology alignment. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *4th International Semantic Web Conference, ISWC 2005, Proceedings*, volume 3729 of *Lecture Notes in Computer Science*, pages 624–637. Springer, 2005.

[25] H. Suguri, E. Kodama, M. Miyazaki, H. Nunokawa, and S. Noguchi. Implementation of FIPA Ontology Service. In *Workshop on Ontologies in Agent Systems, Proceedings*, 2001.

[26] C. Vecchiola, A. Grosso, and A. Boccalatte. AgentService: a framework to develop distributed multi-agent systems. *Int. J. Agent-Oriented Software Engineering*, 2(3):290 – 323, 2008.

[27] W3C. OWL Web Ontology Language Overview – W3C Recommendation 10 February 2004, 2004.

[28] Wikipedia. Upper ontology – Wikipedia, the Free Encyclopedia, 2008. [Online; accessed 30-March-2008].