Alignment infrastructure for ontology mediation and other applications

Jérôme Euzenat

INRIA Rhône-Alpes, Montbonnot, France,
Jerome.Euzenat@inrialpes.fr

Abstract. Web services are not the only application requiring ontology matching and mediation. Agent communication, peer-to-peer systems, etc. also need to find relationships between ontologies. However, they do not necessarily require the same kind of mediation as web services. In order to maximise the utility of the semantic web infrastructure, it seems reasonable to share the mediation services among these applications. To that extent we propose an infrastructure based on the reified notion of alignments and show how it can be used in these various cases.

1 Introduction

Like the web, the semantic web will have to be distributed and heterogeneous. Its main problem is the integration of the resources that compose it. For contributing solving this problem, data is expressed in the framework of ontologies. However, ontologies themselves can be heterogeneous and some work has to be done to achieve interoperability.

Web services and semantic web services suffer from the same problems. There is no reason to think that the resources required, provided and consumed by services are described in the same ontology by services. Requiring this would be a threat to innovation in the domain. In consequence, before comparing or composing services, it is necessary to find correspondences between the heterogeneous ontologies that are used for describing them.

Semantic interoperability can be grounded on ontology reconciliation: finding relationships between concepts belonging to different ontologies.

We call this process "ontology matching". The ontology matching problem may be described in one sentence: given two ontologies each describing a set of discrete entities (which can be classes, properties, rules, predicates, or even formulas), find the correspondences (e.g., equivalence or subsumption) holding between these entities. This set of correspondences is called an alignment. Since the terminology is not shared between everyone, here are the definitions of the various terms used in this paper:

matching is the task of comparing two ontologies and finding the relationships between them;

alignment is the result of the matching task: it is a set of correspondences;

correspondence the relation holding (or supposed to hold according to a particular matching algorithm or individual) between two entities of different ontologies. These entities can be as different as classes, individuals, properties or formulas. Some authors use the term "mapping" or "mapping rule" that will not be used here.

transformation a program that transforms an ontology from one ontology expression language to another;

translation a program that transforms formulas with regard to some ontology into formulas with regard to another ontology (translation can be implemented by a set of translation rules, an XSLT stylesheet or a more classical program).

bridge axioms are formulas in an ontology language that expresses the relations as assertions on the related entities. They are used when merging ontologies.

mediator in web services a mediator is a translation of an information stream, in query application it is a dual pair of translations that transforms the query from one ontology to another and that translate the answer back.

This has been identified in some frameworks as ontology mediation [4]. Ontology mediation consists of taking advantage of the relations between ontologies for composing services. The semantic web service concept depends on the availability of mediators anywhere and anytime.

So it seems that the solution is in sharing these alignments. We pretend that semantic web services should benefit from a wider infrastructure than ontology mediation tools built for semantic web services only. There are several reasons for this:

Each application can benefit from more algorithms Many different applications have needs similar to those of web services. It is thus appropriate to share the solutions to these problems. This is especially true as alignments are quite difficult to provide.

Each algorithm can be used in more applications Alignments can be used for different purposes and must be expressed in a more general way than web service mediators so that they could be used in other contexts.

Each individual alignment can be reused by different applications There is no magic algorithm for quickly providing a useful alignment. Once high quality alignments have been established – either automatically or manually –, it is very important to be able to store, share and reuse them.

We present in the next section various contexts in which aligning ontologies are necessary (§2) and what are the needs of these applications (§3). We then describe an alignment infrastructure for the semantic web (§4) that can be used, in particular, for ontology mediation in semantic web services.

2 Ontology heterogeneity problems on the semantic web

In this section we introduce a number of examples of semantic heterogeneity on the semantic web and the benefit they can have from some alignment services. This section can be skipped by those already convinced of the ubiquity of this problem and that alignments can be applied to solve it.

2.1 Editing

The first place where ontology heterogeneity can be found is while designing an application. If it makes heterogeneous resources interoperate, finding the correspondences

between ontologies in order to include some data transformation in the applications is an option. In this simple case, the application developer can find the correspondences by hand and design the corresponding transformations.

Some tools provide support for finding the correspondences, like Protégé through the Prompt suite of tools [15]. This is a first step towards the integration of mediators. However, if the alignments between widely accepted ontologies are required, there are chances that they will have to be found over and over again. An infrastructure capable of storing the alignments and/or transformations and of providing them on demand to other users would be useful.

Newer ontology development environments whose goal is to take into account, from the beginning, networked ontologies take this step. These systems provide together with the ontologies, the mediators for having their data interoperating, both ontologies and mediators being distributed on the web.

This first kind of application is simple because it is static: ontologies are encountered at design time and mediators can be built at that moment. However, in an evolving world, it is better to design adaptive applications that can dynamically take advantage of non expected resources and ontologies.

2.2 Semantic web services

Web services have clearly been designed for being independent and replaceable. So, web service processors are open to incorporate new services in their workflow. For that purpose, they must be able to compare the description of these services (in order to know if they can be used and for what) and to route the knowledge they process (in order to compose different services together).

However, in case of semantic web services, which can be described with regard to different ontologies, imposing a central common ontology does not seem realistic and would freeze the possibility of such services. So ontologies used in descriptions must be matched and mediators must be able to translate the output of one service in a suitable input for another service [4]. This task can be carried out by hand or by a programme, online or off-line.

It is thus necessary to generate and store these mediators. Although this can be done within some static web service workflows, dynamic composition of services requires a more open infrastructure in which the mediator is defined and involved at run time. We think that it is preferable to be able to take advantage of what can be provided by the environment. For that purpose, the environment must store this mediator or some independent representation from which a mediator can be generated on demand.

2.3 Meaning negotiation in multi agent systems

Agents communicate by exchanging messages. The content of these messages is expressed in some language, very often with regard to some ontology accessible to the agent. The FIPA Agent Communication Language [10, 9] makes provision for declaring within messages, the ontology in which their content is expressed.

In a society of independently developed agents, it may happen that agents using different ontologies have to communicate. Several proposals have been made to address

this situation [1, 17, 16] and we recently proposed ours [8]. Their goal is to assess the correspondences between the terms of the ontologies. However, these correspondences, once established (through negotiation or other means), are not stored and cannot be used in further dialogues or by other similar agents.

In order to share the benefits of the agreed alignments, it is useful to have an infrastructure able to find and deliver these alignments. To avoid having all agents embedding heavy matching methods, it is more convenient to have agents and services able to share stored alignments and matching capabilities. Moreover, having a specific representation of alignments enables agents to negotiate them through argumentation before using them [8].

2.4 Matching contexts in ambient computing

In ambient computing, applications take advantage of the environment for providing services to users. This environment changes, e.g., with regard to the user location, and applications must always keep track of the changing environment involving new devices and sensors. By doing so, they will provide context aware solutions. If one wants to design flexible and smart ambient computing applications, it is necessary to take advantage of ontologies of these various devices, sensors and their capabilities [5]. Like in web service descriptions, these ontologies will provide description of the devices (even abstract devices like a temperature service) and the way to interact with them.

Again, it is expected that device providers will develop different ontologies adapted to their products or will extend some standard ontologies. Moreover, since applications evolve in ever changing environments in which devices fail and new ones are added, there is no way to freeze once and for all the considered ontologies.

In order to properly operate ambient computing environments, applications have to be expressed in terms of generic features that are matched against the actual environment. This matching process can take advantage of ontology matching. Because the same devices will be met by similar applications, providing a service for reconciling the various ontologies and storing the result will help these application to share established alignments.

2.5 Peer to peer mediation

Peer-to-peer systems, when used for sharing discrete resources (files containing images, music, texts, etc.), are organised independently by each peer. Currently, their descriptions correspond to a hierarchy in which the files are stored. However, improving the search on these systems requires a finer description of items. Anyone wanting to share their pictures with their family wants to index them by the name of depicted people, the place where it is, the sights in the pictures, etc. This tagging scheme will benefit from using some ontological description (for retrieving the pictures of “one of your daughters on a horse” by opposition to “Jenny on Tornado”).

Individuals have no or little reasons to commit to the same ontology (like they do not do in current peer-to-peer systems) and they cannot be prevented from modifying and refining their current lightweight ontologies [3].

So the use of peer-to-peer systems consists of querying for information and resources to a network of peers, which are all described in autonomous ontologies; in this situation the queries (and sometimes the answers) must be translated from peer to peer.

This is up to the peer-to-peer systems (with the help of their users) to find the correspondences between ontologies that enable answering queries. This need was explicit in projects like Edutella [14].

For that purpose, it is useful to be able to propose similar ontologies to users if they feel like adopting some standard ontologies. It is also useful to provide them with alignments when they want to issue some query towards other peers and to keep track of these alignments in order not to bother users again and again. These alignments can be used in the reverse direction and shared among the peers.

2.6 Emergent semantics between users

In most of the mentioned applications, nodes have their own ontologies and share alignments between these ontologies. Because individuals and organizations can have different needs and different standpoints, it may be uncomfortable to commit to a single common ontology. However, the social action of constantly aligning ontologies can be the occasion to confront and revise these ontologies. So, in an even more dynamic way, users may want to establish more consensual ontologies from this confrontation [18].

There are several ways in which this can be helped by some alignment infrastructure:

- Producing alignments contributes to solve the gap between ontologies. The alignments are a basis from which some discussion can start (like agent protocol for arguing about correspondences, see § 2.3);
- Alignment algorithms are very often able to compute a distance between ontologies. This is useful when one wants to find the closest ontology.
- By building some network of ontologies and alignments in which the alignments measure the distance between ontologies, it is possible to find out the proximity between people and agents with the help of social network analysis techniques. They will provide indication that can help customising the query process, the matchmaking process between people and even the consensus building.

These results will help users and community consolidating their ontologies by first achieving consensus among the more similar representations. This can also be used for selecting the most central ontology (in social network analysis terms).

2.7 Safe composition infrastructure

Last, but not least, we proposed in [6] a transformation infrastructure for the semantic web. Its goal is to provide transformations between heterogeneous ontologies that guarantee the properties satisfied by the transformations. These properties can be semantic properties like model or consequence preservation, or the preservation of some types of formulas or that only some types of entities are aligned. An organisation can also certify the alignment.

In a web context, transformations, their properties and their proofs can be made available on the network. Knowing the semantics of languages and the proofs of the transformation enables the application of the "proof-carrying code" idea [13] to knowledge exchange. Consequently, the user can be sure that the result of the transformation is the transcription of the initial knowledge with the desired properties.

In applications like semantic web services, it is critical, before launching some complex workflow to know that it satisfies some properties. In particular, it is important to assess the level of correctness of mediators (i.e., that an entity is not translated into another entity that will lead the web service to perform inadequately): the correspondences between ontologies cannot be assumed correct. Proving them would be a must.

3 Towards a common solution

It is clear, from the above examples, that matching ontologies is a major issue on the semantic web. It is not circumscribed to semantic web services, but applies to any application that communicates through ontologies.

3.1 Global needs...

As heterogeneous ontologies are a global problem of the semantic web, this calls for an infrastructure able to help these different applications to deal with it. In such a way, the effort of interoperating ontologies does not need to be solved in each kind of use made of this semantic web.

Moreover, given the difficulty of the matching tasks, there are a few algorithms available and when good alignments are available, it would be very useful to share them among applications. Hence a global solution requires sharing among various instances of applications and not only within each kind of application (web services, agents, etc.).

Given that the task (finding correspondences between ontologies) is very basically the same, it seems possible to use the same tools for that purpose.

3.2 ...but heterogeneous needs

However, if all these applications require ontology matching, they require it for different purposes:

- edition requires the ability to transform some ontology in order to integrate it or to generate a set of bridge axioms that will help identify corresponding concepts (the transformations act at the ontological level);
- agent communication requires translators for messages sent from one agent to another (they act at the data level); similarly, semantic web services require one-way data translations for composing services;
- peer-to-peer systems and more generally query systems require bidirectional mediators able to translate queries (ontological level) and translate back answers (data level).

So, sharing between the applications mentioned here is difficult at the level of the particular reconciliation services they require. For these reasons, it is certainly more convenient to share the matching results themselves (i.e., the alignments) and to ask the infrastructure to be able to generate the appropriate mediators.

3.3 Requirements

This infrastructure should be able to store and retrieve alignments as well as providing them on the fly. We call it an infrastructure because it will be shared by the applications using ontologies on the semantic web. However, it may be seen as a directory or a service by web services, as an agent by agents, as a library in ambient computing applications, etc.

Services that are necessary in such an infrastructure are:

- The ability to store alignments, whether they are provided by automatic means or by hand;
- Their proper annotation in order for the clients to evaluate the opportunity to use one of them or to start from it (this starts with the information about the matching algorithms, the justifications for correspondences that can be used in agent argumentation, as well as properties of the alignment, see § 2.7);
- The ability to produce alignments on the fly through various algorithms that can be extended and parameterized;
- The ability to generate knowledge processors such as mediators, transformations, translators, rules as well as to process these processors if necessary;
- The possibility to find similar ontologies and to contact other such services in order to ask them for operations that the current service cannot provide by itself.

In addition, it is necessary that services be able to exchange between them the alignments they found and select them on various criteria.

4 Architecture proposal for an alignment infrastructure

We argue below that alignments are the necessary structure for supporting this infrastructure. We propose an infrastructure made of a network of services that can be invoked with some particular commands. These services will manipulate alignments through an embedded implementation of our Alignment API.

4.1 Alignments

[2] tried to provide some strict definition of the alignment structure so as to be able to use and reuse it in various situations. Given two ontologies O and O', alignments are made of a set of correspondences (called mappings when the relation is oriented) between pairs of (simple or complex) entities $\langle e, e' \rangle$ belonging to O and O' respectively.

A correspondence is described as a quadruple:

$$\langle e, e', R, n \rangle$$

where:

- e and e' are the entities (e.g., formulas, terms, classes, individuals) between which a relation is asserted by the correspondence;
- R is the relation holding between e and e', asserted by the correspondence. For instance, this relation can be a simple set-theoretic relation (applied to entities seen as sets or their interpretation seen as sets), a fuzzy relation, a probabilistic distribution over a complete set of relations, a similarity measure, etc.
- n is a degree of confidence in that correspondence (this degree does not refer to the relation R, it is rather a measure of the trust in the fact that the correspondence is appropriate – "I trust 70% the fact that the correspondence is correct/reliable/..." – and can be compared with the certainty measures provided by meteorological agencies). The trust degree can be computed in many ways, including users' feedback or log analysis.

So, the simplest kind of correspondence (level 0) is:

$$URI1 = URI2$$

while a more elaborate one could be:

$$URI1(x, y, z) \Leftarrow_{.85} URI2(x, w) \wedge URI3(z, concat(y, w))$$

The first one express the equivalence (=) of what is denoted by two URIs (with full confidence), while the second one is a Horn-clause expressing that if there exists a w such that $URI2(x, w)$ and $URI3(w, concat(y, z))$ is true in one ontology then $URI1(x, y, z)$ must be true in the other one (and the confidence is here quantified with a .85 degree).

As can be observed from these two examples, alignments in themselves are not tied to a particular language. But in order to use complex alignments like the second one, systems must be able to understand the language in which formulas and relations are expressed. This is supported through the definition of a particular subtype of alignment (the first example resorting to the level zero of alignment).

We claim that alignments are more intelligible than transformations: they only express correspondences between ontology entities, not the way they must be used. This can be the basis for studying their properties (moreover, these properties can also be inferred from the methods used for generating alignments).

In order to help developing applications based on alignments, we designed the Alignment API [7]. It has been developed in the aim of manipulating a standard alignment format for sharing among matching systems. But it provided the features required for sharing them more widely. The API is a JAVA description of tools for accessing alignments in the format presented above.

The Alignment API as been implemented on top of the OWL API (other implementations could be based on totally different languages). This implementation offers the following services:

- Storing, finding, and sharing alignments;
- Piping alignments algorithms (for improving an existing alignment);
- Manipulating (trimming and hardening) and combining (merging, composing) alignments;

- Generating "mediators" (transformations, axioms, rules in format such as XSLT, SWRL, OWL, C-OWL, WSML);
- Comparing alignments (like computing precision and recall or a symmetric distance with regard to a particular reference alignment).

The API also provides the ability to compose matching algorithms and manipulating alignments through programming. Part of the interface of the API is presented in Table 1. The API can be used for producing transformations, rules or bridge axioms independently from the algorithm that produced the alignment. Since its definition, several matching systems have been developed within this API (OLA, OMAP) and more of them are able to generate its format (FOAM, Prompt, Falcon, etc.).

4.2 Alignment services

Our architecture is based on *Alignment services*. These services are able to perform a number of alignment tasks and offer them to the other agents or services. These tasks are summarised in Table 1.

Service	Syntax
Finding a similar ontology	$O' \Leftarrow Match(O, T)$
Align two ontologies	$A' \Leftarrow Align(O, O', A, P)$
Thresholding	$A' \Leftarrow Threshold(A, V)$
Generating code	$P \Leftarrow Render(A, language)$
Translating a message	$m' \Leftarrow Translate(m, A)$
Storing alignment	$n \Leftarrow Store(A, O, O')$
Suppressing alignment	$Delete(n)$
Finding (stored) alignments	$\{n\} \Leftarrow Find(O, O')$
Retrieving alignment:	$\langle O, O', A\rangle \Leftarrow Retrieve(n)$

Table 1. Services provided by the alignment service and corresponding API primitives (O denotes an ontology, A an alignment, P parameters, n an index denoting an alignment, P a programme realising the alignment and T and m some expressions).

Most of these services correspond to what is provided by any implementation of the Alignment API. They are exposed to clients through various communication channels (FIPA ACL, SOAP messages) so that all clients can effectively share the infrastructure.

The alignments are indexed by ontology pairs and by surrogates allowing fast retrieving. To one surrogate corresponds only one alignment while for an ontology pair, there can be several such alignments. There is no constraint that the alignments are computed online or off-line (i.e., they are stored in the alignment store) or that they are processed by hand or automatically. This kind of information can however be stored together with the alignment in order for the client to be able to discriminate among them.

The main principle of the Alignment API is that it can always be extended. In particular, it is possible to add new matching algorithms and mediator generators that will be

accessible through the API. They will also be accessible through the alignment services. Services can thus be extended to new needs without breaking the infrastructure.

Moreover, the kind of annotations put on alignments is also extensible. So far, alignments contain information about:

- the kind of alignment it is (1:1 or n:m for instance);
- the algorithm that provided it (or if it has been provided by hand);
- the language level used in the alignment (level 0 for the first example, level 2Horn for the second one);
- the confidence in each correspondence.

Other valuable information that may be added to the alignment format are:

- the parameters passed to the generating algorithm;
- the properties satisfied by the correspondences (and their proof if necessary);
- the certificate from a issuing source;
- the limitations of the use of the alignment;
- the arguments in favour or against a correspondence [8].

4.3 End-user support

The main purpose of the alignment infrastructure is to be invoked by applications that use the alignments for themselves and generally hide the alignments away from users.

However, in some cases, it is necessary that users have access to these alignments and the alignment services. This is particularly the case in the editor application in which it may be useful to allow users to display and modify the alignments by hand. More generally, it is useful for any application in which alignments can be computed and reviewed off-line. Edition facilities must enable to provide high quality reviewed alignments to the infrastructure.

This is also useful in the emergent semantics application when it involves users, because inspecting alignments can help providing arguments for reaching consensus.

Again, using alignments by opposition to more operational transformation or mediators opens the opportunity to share the same tools all over the infrastructure (which does not exclude, in addition, to use some more specific editors). It will thus be useful to provide standard tools for ontology editing that can be shared among these various applications. The VisOn[1] tool developed by University of Montréal is such a tool that can be used for editing alignments in the Alignment API format. Other tools such as the WSML Mapping language editor [11] could be adapted.

4.4 Inter-service communication

Alignment services must be found on the semantic web. For that purpose they can be registered by service directories (e.g., UDDI for web services). Services or other agents should also be able to subscribe some particular results of interest by these services.

[1] http://www.iro.umontreal.ca/~owlola/visualization.html

These directories are useful for other web services, agents, peers to find the alignment services. They are even more useful for alignment services to basically outsource some of their tasks. In particular, it may happen that:

- they cannot render an alignment in a particular format;
- they cannot process a particular matching method;
- they cannot access a particular ontology;
- a particular alignment is already stored by another service.

In these events, the concerned alignment service will be able to call other alignment services. This is especially useful when the client is not happy with the alignments provided by the current service, it is then possible to either deliver alignments provided by other services or to redirect the client to these services.

Moreover, this opens the door to value-added alignment services which use the results of other services as a pre-processing for their own treatments or which aggregates the results of other services in order to deliver a better alignment.

Such an organisation takes full advantage of the goal assigned first to the Alignment API, namely, the ability to compose matching algorithms, here under the form of alignment services.

5 Example

We provide below an example of the use of this alignment infrastructure in the context of agent communication. All actors are agents which communicate through messages using a small set of FIPA Agent Communication Language message types. The communication rules obey a precise protocol that has been defined in [8].

The scenario presented here involves four agents: two agents a and b are communicating but agent a uses ontology O while agent b uses O'. b will call two alignment services c and d with c being a powerful aligner with a restricted access to b environment (it cannot access O') and d having a broader access. Lines beginning with "//" provide explanations for the dialogue moves.

// Agent a is looking for a book and asks agent b
a−query-ref(:ontology O
:language RDQL
:content "SELECT x WHERE x O:autobiography http://www.bertrandrussell.com"
:reply-with 1)→ b
// Agent b does not understand ontology O and asks service c to align it with O'
b−request(:content align(O,O',$\emptyset$,$\emptyset$) :reply-with 1)→ c
// Service c cannot reach ontology O'
b ←failure(:in-reply-to 1 :content unreachable(O'))−c
// Agent b asks d to find a similar ontology
b−request(:content find(O',m) :reply-with 2)→ d
O'' ⇐Match(O',T)
// Service d found O''
b ←inform(:in-reply-to 2, :content O'')−d

// Agent b asks service d to align O' with O''
$b-$request(:content is-align(O',O'') :reply-with 3)$\rightarrow d$
$s \Leftarrow$ Find(O',O'',$\emptyset$,$\emptyset$)
// Service d had already stored such an alignment and returns it
$b \leftarrow$inform(:in-reply-to 3, :content s')$-d$
// Agent b asks service c to align O with O''
$b-$request(:content align(O,O'',$\emptyset$,$\emptyset$) :reply-with 4)$\rightarrow c$
$A \Leftarrow$ Align(O,O'',$\emptyset$,$\emptyset$)
$s \Leftarrow$ Store(O,O'', A)
// Service c computes the alignment
$b \leftarrow$inform(:in-reply-to 4, :content s)$-c$
// Agent b asks service c to translate the message with the found alignment
$b-$request(:content translate(m, s) :reply-with 5)$\rightarrow c$
$\langle O, O'', A\rangle \Leftarrow$ Retrieve(s)
$m \Leftarrow$ Translate(m, A)
$b \leftarrow$inform(:in-reply-to 5
:content "SELECT x
WHERE x O':biography http://www.bertrandrussell.com.
x O':author http://www.bertrandrussell.com.")$-c$
// Agent b asks service d to translate the result with the O' to O'' alignment
$b-$request(:content translate(m', s') :reply-with 6)$\rightarrow d$
$\langle O', O'', A'\rangle \Leftarrow$ Retrieve(s')
$m'' \Leftarrow$ Translate(m', A')
$b \leftarrow$inform(:in-reply-to 6
:content "SELECT x
WHERE x rdf:type O':biografia.
x dc:subject http://www.bertrandrussell.com.
x dc:creator http://www.bertrandrussell.com.")$-d$
// The returned query is evaluated by agent b
QueryResult(m'') $\Rightarrow$ x=http://isbn.org/2-436-4428-1
// which returns the answer to agent a
$a \leftarrow$reply-ref(:content "x=http://isbn.org/2-436-4428-1" :in-reply-to 1)$-b$
// a is satisfied and wants to know the publisher of the book
$a-$request-ref(:content "http://isbn.org/2-436-4428-1 O:publisher x" :reply-with 2)$\rightarrow b$
// b had not recorded the alignment surrogate and asks it to c
$b-$request(:content align(O,O",$\emptyset$,$\emptyset$) :reply-with 7)$\rightarrow c$
// which only have to retrieve it in its store
$s \Leftarrow$ Find(O,O",$\emptyset$,$\emptyset$)
$b \leftarrow$inform(:content s :in-reply-to 7)$-c$
// b asks c for a program in order to translate the messages by itself
$b-$request(:content render(s, C-OWL) :reply-with 8)$\rightarrow c$
// but c cannot deliver this format
$b \leftarrow$failure(:content unsupported(C-OWL) :in-reply-to 8)$-c$
// so b ask for another one
$b-$request(:content render(s, XSLT) :reply-with 9)$\rightarrow c$

$\langle O, O'', A\rangle \Leftarrow$ Retrieve(s)
$P \Leftarrow$ Render(A, XSLT)
$b \leftarrow$inform(:content P :language XSLT :in-reply-to 9)$-c$
// which is delivered and used by b to translate the message
$m' \Leftarrow P(m)$
// The translation goes once again through d
$b-$request(:content translate(m', s') :reply-with 10)$\rightarrow d$
$\langle O', O'', A'\rangle \Leftarrow$ Retrieve(s')
$m'' \Leftarrow$ Translate(m', A')
$b \leftarrow$inform(:in-reply-to 10
:content "SELECT x
WHERE http://isbn.org/2-436-4428-1 dc:publisher x")$-d$
// and the query is processed by b
QueryResult(m'') $\Rightarrow$ "x=http://www.example.com/#Routledge"
// which returns the result
$a \leftarrow$reply-ref(:content "x=http://www.example.com/#Routledge" :in-reply-to 2)$-b$

We have not described the use of these services for semantic web services. However, any application for composing web services may use exactly the same services for generating mediators.

6 Related work

Most of the work on general organisation of alignments is tied to some kind of application (e.g., C-OWL for peer-to-peer applications, WSMX for web services, Edutella for emerging semantics). The work most similar to the one presented here is that on MAFRA [12]. MAFRA proposes an architecture for dealing with "semantic bridges" that offers many functions such as creation and storing of such bridges. The dimension that distinguishes both works is the insistence of MAFRA to have a transformation associated with bridges. Although the transformations can take very different forms, this prevents the use of the same alignment for different purposes: the very benefit of the proposed architecture.

7 Conclusion

In the semantic web, ontologies are used by different kinds of applications and they all suffer from ontology heterogeneity. Henceforth, we have considered the problem of generating mediators for semantic web services as a global problem for the semantic web rather than a specific web service problem. Doing so enables semantic web applications to share ontology alignments instead of developing concurrent pieces of infrastructure.

A common infrastructure have to rely on alignments instead of mediators because alignments can be used by any application. We proposed an alignment infrastructure

based on our Alignment API. The API already implements most of the functions required for the alignment service. It has been enhanced by an agent interaction protocol that enables communicating with it and we are developing a web service interface. The API has been used by various groups, in particular for plugging in their matching algorithms. This API is extensible and can deliver alignments for many different languages.

It is too early to consider the scalability of this approach. On the one hand, aligning on the fly and using many different ontologies seems a threat to scalability. On the other hand this may be regarded as a lightweight process with regard to the cost of some heavy standardisation process. We think that the factor that will help this alignment infrastructure to scale is its flexible nature.

Of course, the proposed architecture, beside the alignment format and the interaction primitives, is open. This means that any other implementation than the current alignment API can be provided as one service and interact with the other ones.

The same service can be shared by all in a natural way (for agent, the alignment service is another agent, for services, it is a web service). The fact that the same service is used throughout the semantic web multiplies the chances that required alignments are already available and prevents the waste of resources.

We hope to have been convincing that such an infrastructure is worthwhile and would contribute to the growth of the semantic web and semantic web services.

Acknowledgements: This work has been partly supported by the European network of excellence Knowledge Web (IST-2004-507482).

References

1. Sidney Bailin and Walt Truszkowski. Ontology negotiation: How agents can really get to know each other, 2002.
2. Paolo Bouquet, Jérôme Euzenat, Enrico Franconi, Luciano Serafini, Giorgos Stamou, and Sergio Tessaris. Specification of a common framework for characterizing alignment. deliverable D2.2.1, Knowledge web NoE, 2004.
3. Paolo Bouquet, Luciano Serafini, and Stefano Zanobini. Semantic coordination: A new approach and an application. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proceedings of the 2nd International Semantic Web Conference (ISWC)*, volume 2870 of *Lecture Notes in Computer Science*, pages 130–145, Sanibel Island (FL, USA), October 2003. Springer Verlag.
4. Chris Bussler, Dieter Fensel, and Alexander Mädche. A conceptual architecture for semantic web enabled web services. *SIGMOD Records*, 31(4):24–29, 2002.
5. Joelle Coutaz, James Crowley, Simon Dobson, and David Garlan. Context is key. *Communications of the ACM*, 48(3):49–53, 2005.
6. Jérôme Euzenat. An infrastructure for formally ensuring interoperability in a heterogeneous semantic web. In Isabel Cruz, Stefan Decker, Jérôme Euzenat, and Deborah McGuinness, editors, *The emerging semantic web*, pages 245–260. IOS press, Amsterdam (NL), 2002.
7. Jérôme Euzenat. An API for ontology alignment. In *Proc. 3rd international semantic web conference, Hiroshima (JP)*, pages 698–712, 2004.
8. Jérôme Euzenat, Loredana Laera, Valentina Tamma, and Alexandre Viollet. Negotiation/argumentation techniques among agents complying to different ontologies. deliverable D2.3.7, Knowledge web NoE, 2005.

9. FIPA ACL communicative act library specification. Technical report, FIPA, 2002. http://www.fipa.org/specs/fipa00037.
10. FIPA ACL message structure specification. Technical report, FIPA, 2002. http://www.fipa.org/specs/fipa00061.
11. Armin Haller, Emilia Cimpian, Adrian Mocan, Eyal Oren, and Chris Bussler. WSMX – a semantic service-oriented architecture. In *Proceedings International Conference on Web Services (ICWS 2005), Orlando (FL US)*, 2005.
12. Alexander Mädche, Boris Motik, Nuno Silva, and Raphael Volz. MAFRA – a mapping framework for distributed ontologies. In *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, pages 235–250, 2002.
13. George Necula and Peter Lee. Efficient representation and validation of proofs. In *Proceedings of the 13th symposium on "logic in computer science", Indianapolis (IN US)*, pages 93–104, 1998.
14. Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. EDUTELLA: A P2P networking infrastructure based on RDF. In *Proceedings WWW Conference, Hawaii (HA US)*, 2002.
15. Natasha Noy and Mark Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proc. 17th AAAI, Austin (TX US)*, pages 450–455, 2000. http://citeseer.nj.nec.com/528663.html.
16. Jun Wang and Les Gasser. Mutual online ontology alignment. In *AAMAS OAS workshop*, 2002.
17. F. Wiesman, N. Roos, and P. Vogt. Automatic ontology mapping for agent communication. Research memorandum, MERIT-Infonomics, Maastricht (NL), 2001.
18. Anna Zhdanova, Reto Krummenacher, Jan Henke, and Dieter Fensel. Community-driven ontology management: DERI case study. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Compiegne (FR)*, 2005.