Undefined 1 (2010) 1–8
IOS Press

The Alignment server: storing and sharing alignments on the semantic web [1]

Jérôme Euzenat [a,*], and Chan Le Duc [b]
[a] *INRIA & LIG, 655 avenue de l'Europe, 38330 Montbonnot Saint-Martin, France*
[b] *LIASD Université Paris 8 - IUT de Montreuil, 140 rue de la Nouvelle France, 93100 Montreuil, France*

Abstract. For interoperability purposes, applications of the semantic web require alignments between ontologies. The Alignment server is a middleware component for storing and sharing alignments. It enables applications to share alignments featuring metadata annotations. The Alignment server provides access to libraries of matchers and alignments and offers retrieval, manipulation (through applying thresholds or transformations), and export of alignments under different formats. It is accessible from web browsers, integrated development environments, web services and agents. Thanks to its plug-in architecture it can be extended by new matchers, renderers, communication drivers and directories.

Keywords: Alignment server, Ontology matching, Alignment sharing, Ontology alignment, Alignment management

The semantic web relies on ontologies, i.e., theories describing the vocabulary used for expressing data. However, ontologies can be heterogeneous and thus interoperability requires matching them, i.e., finding correspondences between their elements [10].

Alignments are sets of correspondences between ontology entities. Each correspondence simply states that a relation between entities of two ontologies holds. The entities can be classes, properties, instances or more complex entity aggregation depending on the language used. The relations may also be diverse but are usually equivalence or subsumption statements. Alignments are declarative representations of the relations between ontologies, i.e., they are independent from any use of the alignment, such as a query transformation or merged ontologies. A declarative representation can be further manipulated, shared and improved and later used for merging, transforming ontologies or translating data. In addition, alignments are also independent from the way they have been produced, being it by hand or by matching algorithms.

We briefly present the Alignment server, a tool which builds on the Alignment API [6], for storing and sharing alignments on the semantic web. In this presentation, "Alignment server" identifies our own tool while "alignment server" denotes the generic class of systems to which it belongs.

We first motivate the purpose of alignment servers (§1) before describing the architecture (§2), the functions (§3) and plugable parts (§4) of the Alignment server.

1. Why supporting alignments?

We already advocated the need of alignment management for dealing with ontology alignments, and in particular sharing them [7]. There are several reasons to support alignments:

Sharing matching algorithms: Many applications have matching needs. It is thus appropriate to share, across applications, the solutions to these problems: the matching algorithms and systems.

[1] Partial support provided by European Integrated Project NeOn (IST-2005-027595) and ANR project WebContent.
[*] Corresponding author. E-mail: Jerome.Euzenat@inria.fr

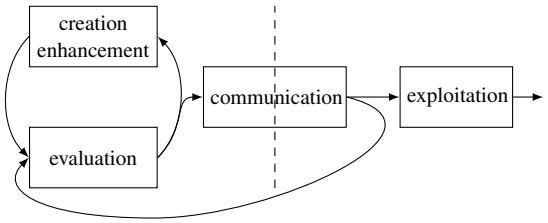

Fig. 1. The ontology alignment life cycle: once created, alignments may be evaluated before being enhanced. When an alignment is satisfactory it can be published so that others can use it. Finally, alignments can be used in applications.

Sharing alignments: Alignments are quite difficult to obtain. There is no magic algorithm for quickly providing a useful alignment. Once high quality alignments have been established – either automatically or manually – and accepted, it is very important to be able to store, share and reuse them.

Sharing exploitation means: Matching results, once expressed as alignments, may be used for different purposes. Hence, the same matcher implementation can be reused in different mediation scenarios by generating the appropriate processable form of alignments, e.g., axioms, query mediators, transformations.

Combining matchers: If one wants to combine several matching systems in a particular application, this is easier if all systems can exchange their results in a pivot language [3,8]: one matcher can work at improving the alignment provided by another matcher.

Like ontologies, alignments have their own life cycle (see Figure 1; [8]). They are first created through a matching process (which may be manual). Then they can go through an iterative loop of evaluation and enhancement. Evaluation consists of assessing properties of the obtained alignment. It can be performed either manually or automatically. Enhancement can be obtained either through manual change of the alignment or application of refinement procedures, e.g., selecting some correspondences by applying thresholds. When an alignment is deemed worth publishing, then it can be stored and communicated to other parties interested in such an alignment. Finally, the alignment is transformed into another form or interpreted for performing actions like mediation or merging.

Ontology alignments, like ontologies, must be supported during their life cycle phases by adequate tools. Indeed, finding these alignments is only the first step of the process. Very often these alignments have to be evaluated, improved and finally transformed into some executable procedure before being used by applications: transforming an ontology in order to integrate it with another one, generating a set of bridge axioms that will help identify corresponding concepts, translating messages sent from one agent to another, translating data circulating among heterogeneous web services, mediating queries and answers in peer-to-peer systems and federated databases.

Hence, an infrastructure capable of storing the alignments and of providing them on demand is needed.

2. Alignment server design

Alignment servers are independent software components which offer a library of matching methods and an alignment store that can be used by their clients. In a minimal configuration, alignment servers contribute to storing and communicating alignments. Ideally, they can offer all the following services:

Matching two ontologies possibly by specifying the algorithm to use and its parameters (including an initial alignment).

Storing an alignment in persistent storage.

Retrieving an alignment from its identifier.

Retrieving alignment metadata from its identifier can be used for choosing between specific alignments.

Finding (stored) alignments between two specific ontologies.

Comparing alignments for determining what their differences are.

Editing an alignment by adding or discarding correspondences (this is typically the result of a graphic editing session).

Trimming alignments, i.e., selecting correspondences within thresholds.

Generating code implementing ontology transformations, data translations or bridge axioms from a particular alignment.

Translating a message with regard to an alignment.

Finding a similar ontology is useful when one wants to align two ontologies through an intermediate one.

For instance, someone wanting to translate a message expressed in ontology o to ontology o'' can ask for matching the two ontologies and for a translation of the message with regard to the obtained alignment.

A more extreme scenario involves (1) asking for alignments between o and o'', maybe resulting in no alignment, (2) asking for an ontology close to o'' which may result in ontology o', (3) asking for the alignments between o and o', which may return several alignments a, a' and a'', (4) asking for the metadata of these alignments and (5) choosing a' because it is certified by a trusted authority, (6) matching o' and o'' with a particular algorithm, (7) suppressing the resulting correspondences whose confidence is under a reasonable threshold for this algorithm, (8) editing the results so that it is correct, (9) storing it in the server for sharing it with other parties, (10) retrieving alignment a' and this latter one as data translators, (11) finally applying these two translations in a row to the initial message.

Alignment servers are middleware components of the semantic web infrastructure. They can be used by semantic web applications as well as by the infrastructure itself. They can be used at two different moment in applications:

at design time through invocation by design and engineering environments: they can be integrated within development environments, where they will be loosely coupled components which may be asked for alignments and for exploiting these alignments (like the NeOn toolkit through the NeOn Alignment plug-in).

at run time alignment servers can be invoked directly by the application.

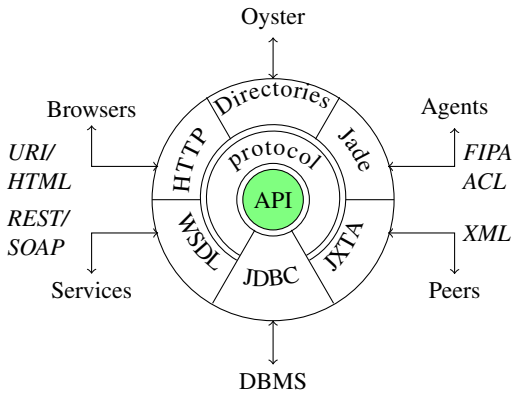

Fig. 2. The Alignment server is built on the Alignment API that is seated on top of a relational database for sharing alignments and is wrapped around a simple protocol. Each access method is a plug-in that interacts with the server through the protocol. Currently, HTML, agent and web service plug-ins are available.

The Alignment server architecture is made of four layers (shown in Figure 2):

The Alignment API which provides operations for manipulating alignments including extension points for adding new matchers and renderers.

A storage system which offers persistent storage and retrieval of alignments. It implements only basic storage and runtime memory caching functions. The storage is made through a DBMS interface and can be replaced by any database management system as soon as it is supported by JDBC.

A protocol manager which handles the server protocol. It accepts queries from plug-in interfaces and uses the server resources for answering them. It uses the storage system for caching results.

Protocol drivers which accept incoming queries in a particular communication system and invoke the protocol manager in order to answer them. These drivers are ideally stateless and only translators for the external queries.

3. Functions

Alignment servers provide the following features:

- A way to manipulate alignments, i.e., load and render them, providing support for traversing them;
- The ability to store alignments, whether they are provided by automatic means or by hand;
- The communication over the network for delivering required alignments;
- Their proper annotation in order for the clients to evaluate the opportunity to use one of them or to start from it.

We consider hereafter how the Alignment server provides these features.

3.1. Manipulating alignments

We take advantage of the Alignment API [6] which has been designed to help developing applications based on alignments. It has been developed in the aim of manipulating a standard alignment format for sharing alignments across matching systems, but it provides the features required for manipulating them more thoroughly. The API is a Java description of tools for accessing alignments. It offers the following services:

- Matching ontologies and representing alignments;
- Piping matching algorithms (for improving an existing alignment);

- Manipulating (trimming under a threshold and hardening) and combining (merging, composing) alignments;
- Loading alignment from files or other streams;
- Writing alignments in output formats (RDF/XML, HTML, etc.);
- Generating "mediators" (transformations, axioms, rules in formats such as XSLT, SWRL, OWL, C-OWL, WSML).
- Comparing alignments, e.g., 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 to manipulate alignments through programming.

The Alignment API can be used in conjunction with an ontology language API (OWL-API [2], Jena OWL API, or the SKOS API [11]). However, for most of the operations of the Alignment server, the ontologies are not loaded and thus no ontology API is required. In principle, only matching and some rendering require ontologies.

3.2. Alignment server for storing

Alignment storage is simply achieved by storing alignments in a relational database management system. Users must explicitly ask for storing an alignment in the database, otherwise, it exists only in main memory. There is no constraint that the alignments are computed online or off-line. However, this origin information can be stored together with the alignment in order for clients to be able to discriminate among them.

Each stored alignment is given a new unique URI: this URI is dependent from the server and cannot be generated by another server. Moreover, stored alignments cannot be modified. Indeed, each time a new URI is issued, so that those who may use this URI should be enabled to retrieve the alignment that they chose. Instead of modifying alignments, a new alignment, identified by a different URI, is created each time an operation is applied. These simple design choices imply that there cannot be concurrent edition of an alignment, hence, concurrency control is not necessary. The storage is permanent: the database can only grow.

We assume the same from ontologies: ontology overcoming changes should be identified differently so that a new alignment can be generated and an old alignment cannot be used with the new ontology. The Alignment server does not automatically update alignments if ontology change.

The Alignment server uses a cache mechanism for loading only metadata in main memory. Alignments themselves are only loaded when they are needed. We plan to implement cache management for selectively unload alignments that have not been used for a long time.

3.3. Communicating

Alignment servers are exposed to clients, either ontology management systems or applications, through various communication channels (agent communication messages, web services) so that all clients can effectively share the infrastructure. A server may be seen as a directory or a service by web services, as a web site by users, as an agent by agents, as a library in ambient computing applications, etc.

Alignment servers 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 be able to subscribe some particular results of interest by these services. These directories are useful for other web services, agents, peers to find alignment servers.

Most of the communication is achieved by synchronous message passing. Messages are transmitted to the protocol manager, which interprets them and returns answers. Except for uploading and rendering alignments, only URI identifying them are transmitted. Hence the protocol uses only the necessary resources.

The Alignment server can be used locally within one site for storing alignments, but its true purpose is that a constellation of Alignment servers be dispatched within the semantic web for any application to use them. Servers may communicate together, in particular through their web service interface. They can exchange the alignments they found and select them on various criteria. This can be useful for alignment servers to 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 server.

3.4. Indexing for sharing alignments

The applications among which alignments are shared may have diverse needs and various selection criteria. Hence, it is necessary to search and retrieve alignments on these criteria. Alignment metadata used for indexing alignments are thus very important. So far, alignments contain information about:

- the aligned ontologies;
- the language in which these ontology are expressed;
- the kind of alignment it is (1:1 or n:m for instance);
- the algorithm or operation that provided it (or if it has been provided by hand);
- the confidence in each correspondence;
- the initial alignment from which an alignment is derived if any.

This information is already very precious and helps applications selecting the most appropriate alignments. It is thus necessary that ontology matchers be able to generate these metadata.

The Alignment server can store them at the alignment level as well as the correspondence level. Metadata are represented as key-value pairs whose keys are URIs for faster retrieval. They are preserved through the serialisation of alignments in RDF. Moreover, the metadata scheme is extensible and other valuable information may be added to alignment[1], such as:

- the parameters passed to the generating algorithms;
- the properties satisfied by the correspondences (and their proof if necessary);
- the certificate from an issuing source;
- the limitations of the use of the alignment;
- the arguments in favor or against a correspondence [13,14].

All such information may be useful for evaluating and selecting alignments and thus should be available from alignment servers.

4. Plug-ins

The Alignment server is designed to be extensible. Hence, there are several documented access points that can be filled with plug-ins. In each case below, some standard plug-ins are available in the Alignment server and more can be added. Plug-ins are detected at launch time and loaded without recompilation, if their java archives are in the path.

4.1. Drivers

Drivers allow to interact with the server through the server protocol. They usually handle low level communication with the clients and translate messages and answers to the protocol manager. Currently, four communication drivers are available for the server:

HTML/HTTP for interacting through a browser. This is an easy and portable way to interact with a human user;

FIPA ACL/JADE for interacting with agents: has been used for ambient intelligence [9];

SOAP/HTTP and REST/HTTP for interacting as a web service. They use the same HTTP transport layer as HTML and is the favorite way to communicate with other programs, including other Alignment servers. The interface is described in WSDL, so the binding can be made dynamically.

Since everything is managed by the same protocol, it is possible to start a conversation through one of these drivers and dynamically switch to another.

4.2. Matchers

All matchers developed within the framework of the Alignment API (OLA, TaxoMap, oMAP, etc.) can be used in the Alignment server. The Alignment API itself contains several sample matchers.

The matcher interface is that of the Alignment API (`init()` and `align()`). Matchers can be provided with additional parameters.

For matching tasks which take long time, there is the opportunity run them asynchronously. In this case, the URI of the alignment is returned immediately by the Alignment server, and the Alignment is made available at that URI upon completion.

4.3. Renderers

Renderers are also part of the Alignment API. They generate alignments in a particular output format, either for communication or for processing them further. The Alignment API makes extensive use of our Alignment format [6]. However, from the client standpoint,

[1] Metadata identifiers are registered at `http://alignapi.gforge.inria.fr/labels.html`.

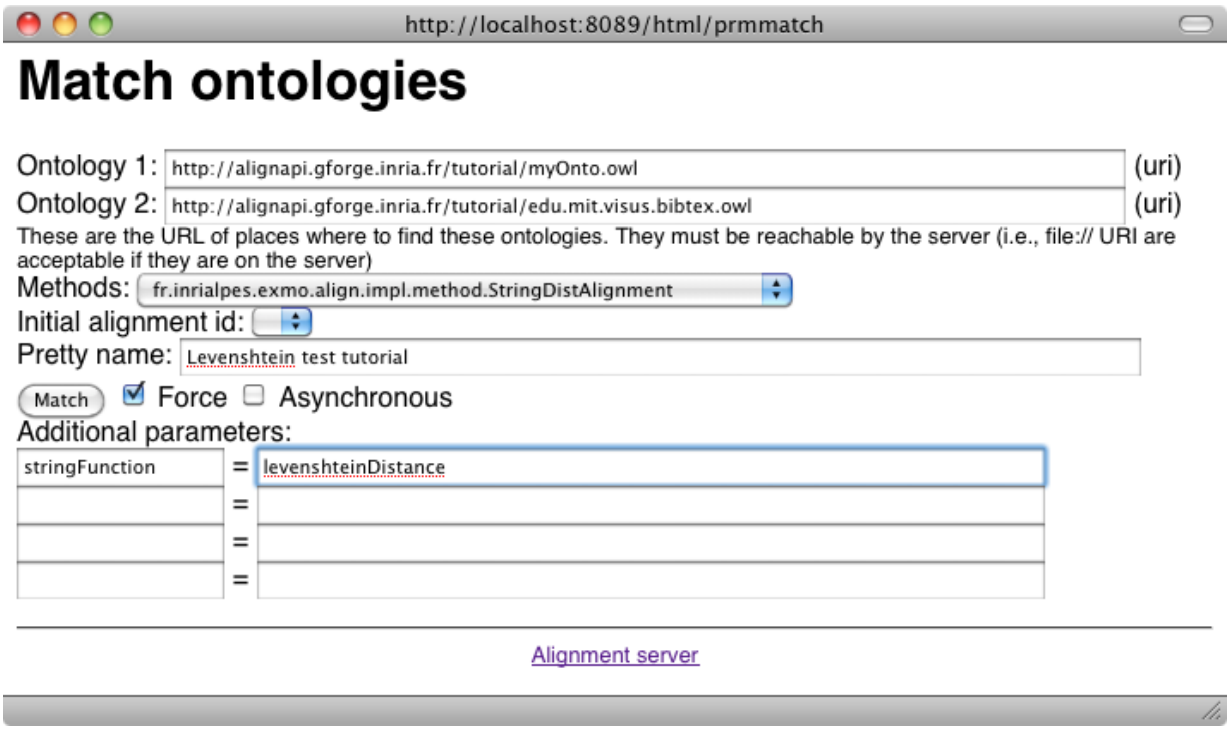

Fig. 3. Screendump of the HTML/HTTP plug-in providing access to the Alignment server.

this is not important since various formats can be delivered through renderers. Other formats for representing alignments are Mapping Documents [16] or the Portable Ontology Alignment Fragments [12]. Several of these are included in the Alignment API and used in the server:

HTML is the way alignments are rendered under the HTML interface (the output is actually in XHTML+RDFa, preserving the full alignment data);

RDF/XML is the Alignment API interchange format and the preferred format for interacting with web services; this renderer also generates Mapping Documents [16];

OWL, SKOS, owl:sameAs are standard formats for expressing relations within ontologies, thesauri and data;

SWRL, C-OWL, XSLT are processable formats which can be directly fed into processors for manipulating data.

4.4. Directories

The Alignment server can connect to directories for two purposes: (1) being visible and found by other servers, (2) finding other server instances which can run different matchers or provide different alignments. Servers can either register themselves to directories so that potential clients know that they are available, but they can also register their alignments so that clients know what they have to offer. The directory interface has been designed with UDDI in mind. It is currently connected to the Oyster peer-to-peer ontology metadata sharing system [15]. Oyster has been extended for featuring some metadata about alignments and all alignments are declared to Oyster.

The directory interface is very simple (connect, disconnect and declare alignment) so that it is easy to add new directories to which the server will connect.

5. Applications

The Alignment server has been used for dynamically providing alignments between device requests and sensor profiles in ambient computing [9]. It has also been integrated in a prototype ontology-based peer-to-peer systems for translating queries through alignments.

It is accessible within the NeOn toolkit, the Cupboard system [4] and the WebContent infrastructure [1].

6. Availability

The Alignment server code is open source and freely available with the Alignment API[2]. Instructions for running a server are given in the documentation. The server requires a JDBC-compliant database server, e.g., mysql, and associated drivers.

A sample server, not to be used for production, is usually available through the HTML and REST interface at `http://aserv.inrialpes.fr`.

7. Related works

The Alignment server, is a basic support component for storing and sharing alignments. It does not pretend to be more than that and, in particular, to function as:

alignment editor [14]: there is no extensive support for graphically visualising and manipulating alignments;

collaborative tool : the server only provides the basis for collaborating: storing and sharing, but does not provide help to negotiate or rate alignments [14];

ontology support and reasoner : the server cannot directly reason about alignments;

mediator : the server does not act as a mediator between applications.

There are a couple of related initiatives. In databases, several systems have been designed for offering a variety of matching methods and a library of "mappings" [3,5]. However, they were meant only as a component for design time integration and not a server that can be used at run time.

More recently, there has been some interest in supporting collaborative ontology alignment creation. The system described in [14] is a collaborative system for sharing both ontologies and mappings (which are our correspondences: alignments between two concepts only). It allows users to share, edit and rate these mappings and is available as a web service.

The main difference with the Alignment server is thus the lack of support for alignments themselves which require specific manipulations on the client side for obtaining a coherent set of mappings. Support for mapping manipulation is also restricted to uploading them, annotating them and selecting them based on metadata. There is no way to match ontologies or to render selected mappings in other ways than as instances of a particular ontology. The strengths of this system are a user friendly interface with the possibility to annotate alignments and the direct connection with the ontologies which helps users to navigate.

The CATCH repository [17] takes a similar approach as ours for thesauri linked by alignments under our format [6]. There are however notable differences: the repository aims at storing both the alignments and the thesauri and it does not provide methods for creating or manipulating the alignments, only for publishing them. In that respect, CATCH is more similar to Cupboard [4].

8. Conclusion

Alignment servers should become a very important piece of the semantic web infrastructure. Beside storing and sharing, they also provide other capabilities such as manipulating the alignments and rendering them under various formats.

Hence, the Alignment server lightweight and focussed design makes it a natural component on which to plug applications and other such tools. For instance, the NeOn toolkit adds alignment support to its ontology support, thanks to the alignment plug-in. Any such tool can take advantage of the Alignment server and focus on their specifics. In particular, the result of their activities can be stored either as new alignments (alignment editor) or as alignment metadata (rating and commenting) and shared with the web at large or a smaller community.

The Alignment server is continuously improved. We are developing the cooperation within a network of alignment servers through their web service interface. This will allow directly sharing alignments and matchers among servers and contributing to robustness.

References

[1] S. Abiteboul, T. Allard, P. Chatalic, G. Gardarin, A. Ghitescu, F. Goasdoué, I. Manolescu, B. Nguyen, M. Ouazara, A. Somani, N. Travers, G. Vasile, S. Zoupanos, Webcontent: Efficient p2p warehousing of web data, in: Proc. VLDB demonstration papers, 2008.

[2] `http://alignapi.gforge.inria.fr`

[2] S. Bechhofer, R. Volz, P. Lord, Cooking the semantic web with the OWL API, in: Proc. 2nd International Semantic Web Conference (ISWC), vol. 2870 of Lecture notes in computer science, Sanibel Island (FL US), 2003.

[3] P. Bernstein, A. Halevy, R. Pottinger, A vision of management of complex models, ACM SIGMOD Record 29 (4) (2000) 55–63.

[4] M. d'Aquin, J. Euzenat, C. Le Duc, H. Lewen, Sharing and reusing aligned ontologies with cupboard, in: Proc. 5th ACM K-Cap poster session , Redondo Beach (CA US), 2009.
URL ftp://ftp.inrialpes.fr/pub/exmo/publications/daquin2009a.pdf

[5] H.-H. Do, E. Rahm, COMA – a system for flexible combination of schema matching approaches, in: Proc. 28th International Conference on Very Large Data Bases (VLDB), Hong Kong (CN), 2002.

[6] J. Euzenat, An API for ontology alignment, in: Proc. 3rd International Semantic Web Conference (ISWC), vol. 3298 of Lecture notes in computer science, Hiroshima (JP), 2004.

[7] J. Euzenat, Alignment infrastructure for ontology mediation and other applications, in: Proc. International Workshop on Mediation in Semantic Web Services (MEDIATE), Amsterdam (NL), 2005.
URL http://ceur-ws.org/Vol-168/MEDIATE2005-paper6.pdf

[8] J. Euzenat, A. Mocan, F. Scharffe, Ontology alignment: an ontology management perspective, in: M. Hepp, P. D. Leenheer, A. D. Moor, Y. Sure (eds.), Ontology management: semantic web, semantic web services, and business applications, chap. 6, Springer, New-York (NY US), 2008, pp. 177–206.

[9] J. Euzenat, J. Pierson, F. Ramparani, Dynamic context management for pervasive applications, Knowledge engineering review 23 (1) (2008) 21–49.

[10] J. Euzenat, P. Shvaiko, Ontology matching, Springer, Heidelberg (DE), 2007.

[11] S. Jupp, S. Bechhofer, R. Stevens, A flexible API and editor for SKOS, in: Proc. 6th European Semantic Web Conference (ESWC), Heraklion (GR), vol. 5554 of Lecture Notes in Computer Science, 2009.

[12] Y. Kalfoglou, P. Smart, D. Braines, N. Shadbolt, POAF: portable ontology alignment fragments, in: U. Sattler, A. Tamilin (eds.), Proc. workshop on ontologies: reasoning and modularity (WORM), Tenerife (ES), 2008.
URL http://ceur-ws.org/Vol-348/

[13] L. Laera, I. Blacoe, V. Tamma, T. Payne, J. Euzenat, T. Bench-Capon, Argumentation over ontology correspondences in MAS, in: Proc. 6th International conference on Autonomous Agents and Multiagent Systems (AAMAS), Honolulu (HA US), 2007.

[14] N. Noy, N. Griffith, M. Musen, Collecting community-based mappings in an ontology repository, in: Proc. 7th International semantic web conference (ISWC), Karlsruhe (DE), 2008.

[15] R. Palma, P. Haase, Oyster: Sharing and re-using ontologies in a peer-to-peer community, in: Proc. 4th International Semantic Web Conference (ISWC), vol. 3729 of Lecture notes in computer science, Galway (IE), 2005.

[16] F. Scharffe, J. de Bruijn, A language to specify mappings between ontologies, in: Proc. IEEE Conference on Internet-Based Systems (SITIS), Yaounde (CM), 2005.

[17] L. van der Meij, A. Isaac, C. Zinn, A web-based repository service for vocabularies and alignments in the cultural heritage domain, in: Proc. 7th European semantic web conference (ESWC), Hersonissos (GR), vol. 6088 of Lecture notes in computer science, 2010.