Chapter 12
Methodological Guidelines for Matching Ontologies

Jérôme Euzenat and Chan Le Duc

Abstract Finding alignments between ontologies is a very important operation for ontology engineering. It allows for establishing links between ontologies, either to integrate them in an application or to relate developed ontologies to context. It is even more critical for networked ontologies. Incorrect alignments may lead to unwanted consequences throughout the whole network, and incomplete alignments may fail to provide the expected consequences. Yet, there is no well-established methodology available for matching ontologies. We propose methodological guidelines that build on previously disconnected results and experiences.

12.1 Motivation

Ontology matching is the activity of establishing correspondences between ontologies. Correspondences express relationships supposed to hold between entities in ontologies, for instance, that a 'district' in one ontology is the same as a 'kreis' in another one or that 'fishery' in an ontology is a subclass of 'company' in another one. An alignment may be used to link an ontology with its background, i.e. set it in a more general context: This is typically what is achieved by providing an alignment with an upper-level ontology. An alignment can also be used to link an ontology with its previous versions or alternative ontologies in other applications.

We use interchangeably the terms matching operation (focussing on the input and result), matching task (focussing on the goal and the insertion of the task in a wider context) and matching activity (focussing on the internal processing).

J. Euzenat (✉)
INRIA & LIG, F-38330 Montbonnot Saint-Martin, France
e-mail: Jerome.Euzenat@inria.fr

C. Le Duc
Université Paris 8, 93200 Saint-Denis, France
e-mail: Chan.Leduc@iut.univ-paris8.fr

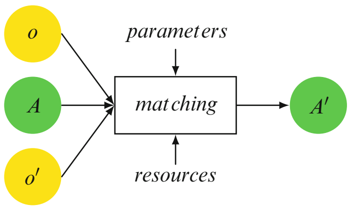


Fig. 12.1 The matching activity (From Euzenat and Shvaiko 2007)

The ontology matching process may be summarised as in Fig. 12.1 by a process taking two ontologies (o and o') as input and returning an alignment (A'), i.e. a set of correspondences. In addition, this process can take as input an initial alignment and various parameters. Ontology matching can be used with more than two ontologies. However, in this chapter, we restrict ourselves to matching two ontologies. As simple as it seems, ontology matching is an unsolved problem and a delicate activity which requires care (Euzenat and Shvaiko 2007). Many matching methods exist, and not one fits all needs.

Ontology matching is a very important operation in modern ontology engineering because of the networked environment in which ontologies are engineered and supposed to work. Methodologically, it is worthwhile to express relations between ontologies because this allows (1) for working with small and self-sufficient modules instead of monolithic ontologies, (2) for expressing the links between two versions of the same ontology and thus to upgrade data from one ontology to another or (3) for putting back an ontology in the context of an upper-level ontology, allowing it to play better with other ontologies. Networked ontologies are sets of ontologies together with alignments relating the entities of these ontologies. These ontologies may be related because they are complementary, two independent domain ontologies, e.g. sales and tyres, refinement, a domain ontology specialising a top-level ontology, or supplementary, a version replacing another version or two ontologies about the same domain. In networked ontologies, alignments are as important as the ontologies themselves because relationships between ontologies are the basis of networks.

Hence, methodological guidance for ontology matching is particularly required and needs to be supported for helping ontology engineers to develop semantic applications. Contrary to ontology building which is an open-ended (design) activity, ontology matching is an inductive activity bounded by the ontology to be matched. Hence, it requires a more focussed methodology.

Yet, very little support exists for such an activity at the methodological or at the tool level. Even in the database field, where similar but simpler problems have existed for years, there is no consensus methodology on how schema matching can be conducted. This chapter provides guidance for matching ontologies based on existing partial guidelines and overall experience collected so far in the field.

We do not consider ontology matching as an independent activity. On the contrary, we consider it as related to ontology management: When ontologies evolve, alignments must follow this evolution. Moreover, as proposed in the work

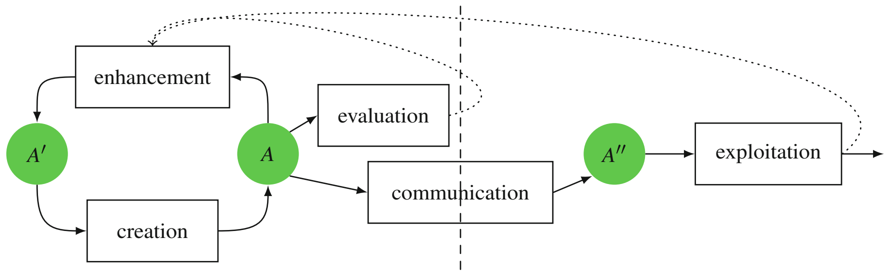


Fig. 12.2 The ontology alignment life cycle (Adapted from Euzenat et al. 2008)

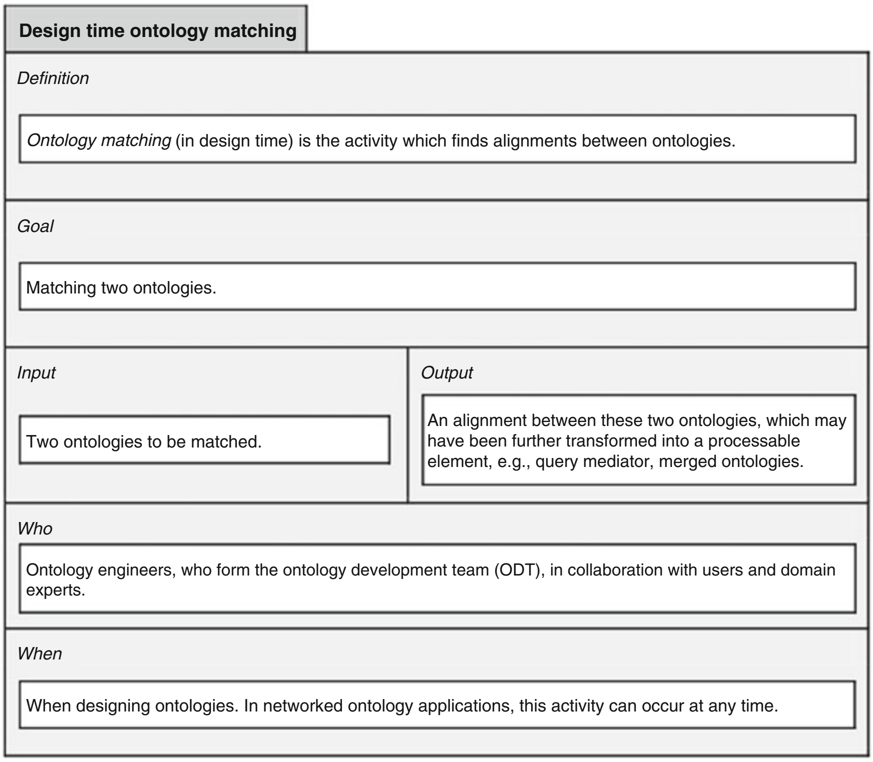
of Euzenat et al. (2008), ontology matching should be considered in a dynamic perspective in which the result of matching has its own life cycle and will have to be maintained and evolved. This is illustrated by Fig. 12.2, representing the alignment life cycle. This life cycle takes into account the evolution of alignments as well as the importance of considering alignment as first-class objects which can be shared. As such, alignments can be manipulated to better suit the needs of users. We consider this ontology alignment life cycle and further investigate the methodological guidelines for supporting it. In the spirit of NeOn (see Chap. 1), these guidelines put the emphasis on networks of ontologies as well as reusing ontologies and alignments.

In what follows, we first introduce synthetic descriptions of the ontology matching activity (Sect. 12.2). Then, we discuss the issue of the format in which alignments have to be delivered in order to support reusable matching (Sect. 12.3) before considering step by step the proposed methodological guidelines (Sect. 12.4). Then we present support offered by tools for the proposed methodological guidelines (Sect. 12.5). Finally, examples are given (Sect. 12.6) before concluding.

12.2 Ontology Matching Filling Cards

We present below two different ontology matching activities. These depend on the time at which ontology matching is supposed to take place. If ontology matching is supposed to occur at design time, then its goal is to match two ontologies for connecting them in a network; if it is to occur at runtime, then the goal of the activity is to generate a matching process that achieves ontology matching at runtime.

This distinction between runtime and design time ontology matching is very important in practice because the output of the two operations is not the same. At design time, the resulting alignment is used for relating the different ontologies which will be used at runtime, for instance, for transforming queries. At runtime,



Design time ontology matching

Definition

Ontology matching (in design time) is the activity which finds alignments between ontologies.

Goal

Matching two ontologies.

Input

Two ontologies to be matched.

Output

An alignment between these two ontologies, which may have been further transformed into a processable element, e.g., query mediator, merged ontologies.

Who

Ontology engineers, who form the ontology development team (ODT), in collaboration with users and domain experts.

When

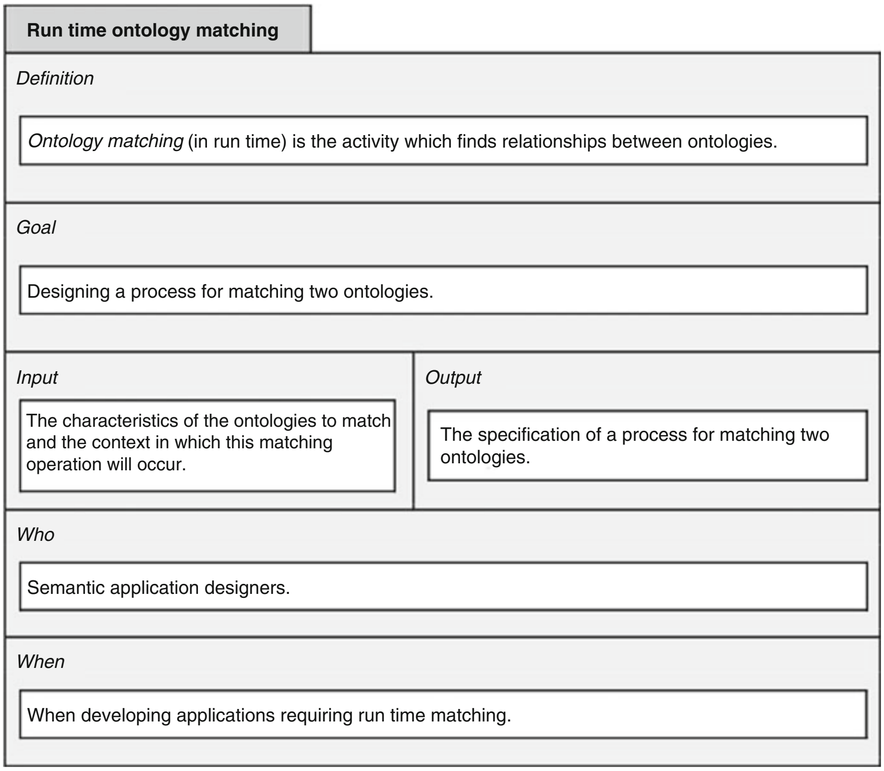
When designing ontologies. In networked ontology applications, this activity can occur at any time.

Fig. 12.3 Design time ontology matching

ontology matching is used for finding alignments between ontologies which were not known at design time. This could be for composing semantic web services using different ontologies, for instance.

When ontology matching is performed at design time (see Fig. 12.3), only the resulting alignment is available at runtime: no more matching is necessary. So, there is no runtime constraint on matching. When it is performed at runtime, no design time alignment is available; the matching will occur at runtime. So, the goal of the designer is to design a matching process instead of an alignment (see Fig. 12.4). In this case, runtime constraints (speed, memory) may apply to matching.

However, functionally, these two operations can also be seen as the same since they, in practice, generate an ontology matching process which is executed at different moments. Hence, the guidelines that we apply are the same in both cases because it consists of choosing software components which are applied at different time.


Run time ontology matching

Definition

Ontology matching (in run time) is the activity which finds relationships between ontologies.

Goal

Designing a process for matching two ontologies.

Input

The characteristics of the ontologies to match and the context in which this matching operation will occur.

Output

The specification of a process for matching two ontologies.

Who

Semantic application designers.

When

When developing applications requiring run time matching.

Fig. 12.4 Runtime ontology matching

12.3 Alignments and Formats

Although formats should not be a main concern for methodology, it is here very important because the input and output of most of the tasks is an alignment. Hence, choosing a common alignment representation makes tasks interoperable and allows for better sharing and reusing the product of the ontology matching activity.

Ontology alignments are sets of relationships between ontology entities. Alignments may be expressed in various languages. For instance, the two relations mentioned in the introduction can be expressed in OWL (Horrocks et al. 2003) through `owl:equivalentClass` and `rdfs:subClassOf`, but they can also be expressed in SKOS (Miles and Bechhofer 2009) through `skos:exactMatch` and `skos:broaderMatch`. Other applications may mandate a different form like views in databases, mediators in web services frameworks or even merged ontologies. The advantage of such representations is that they can be processed.

However, application-specific output is not particularly interoperable. It is not easy to transform a database view into OWL axioms or SKOS statements into ODEDialect (Corcho and Gómez-Pérez 2007). Indeed, when the alignment is expressed in OWL, its only possible use is to 'merge' two OWL ontologies. It cannot easily be used to import data from one ontology to another or to translate queries. Moreover, such formats are not easy to share and retrieve (see Sect. 12.4.7) or to manipulate (see Sect. 12.4.6), e.g. for merging the results of several matchers if they do not use a format that supports such manipulations.

Hence, in order to avoid early commitment to a particular type of usage, it is to be preferred to keep the alignments in a declarative language. Such a language allows for manipulating and composing alignments as well as for generating the required representation (OWL, SKOS and others) when necessary.

Using a neutral and declarative representation (Euzenat 2004) provides the opportunity to distribute and share alignments among applications. This is why, in the remainder, only 'alignments' are considered.

12.4 Detailed Guidelines

Ontology matching has been the focus of a lot of attention in the recent years. However, little work has been carried out on the methodological support for finding alignments. We provide here the outline for such methodological guidelines. It can be summarised by the workflow of Fig. 12.5. Each task of this workflow will be described in subsections.

12.4.1 Identifying Ontologies and Characterising Needs

The first task in finding alignments is to identify the ontologies to be matched and to characterise the need. Indeed, the type of required alignment will be different if the

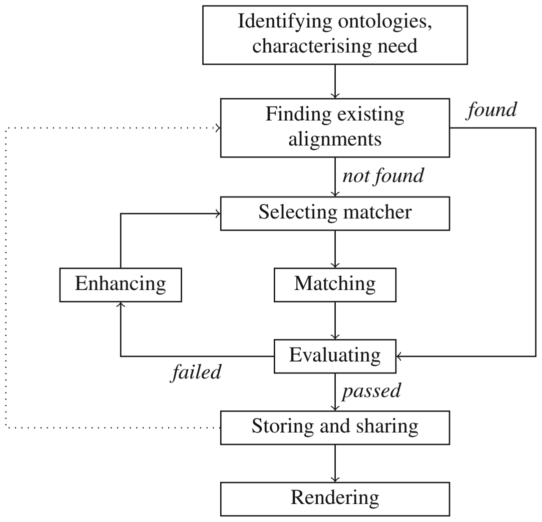


Fig. 12.5 The matching methodology workflow. It goes step by step through characterising the problem, selecting existing alignments, selecting appropriate matchers, running the matchers, evaluating the results and correcting the choices made before (matchers, parameters), documenting and publishing good results and finally using them

goal is to merge two ontologies in a knowledge-based system or to add yet another data source to a query mediator. In the former case, the alignment will have to be strictly correct, otherwise the system may draw incorrect inferences, but the relationships can be diverse: subsumption and disjointness assertions can be very useful. In the latter case, lack of completeness is not a problem since other sources may return the missing answers, but relations other than equivalence are not straightforwardly used in query mediation. This first task is similar to Activity 3 of the work of Corcho (2005), called 'design of [a] translation system', which specifies how to characterise source and target ontologies for ontology translation.

It is also useful to characterise the kind of ontologies: Are they labelled in the same natural language? What is their expressiveness? Are individuals related to the ontologies available?

Characterising the situation in which matching will be performed should not be neglected. It will determine the choice of matchers or alignments as well as the parameters to care for. Euzenat and Shvaiko (2007) identified several parameters:

- Are instances available at match time?
- Is matching performed under time constraints?
- Has matching to be performed automatically?
- Must the alignment be correct?
- Must the alignment be complete?
- What type of operation (merging ontologies, transforming queries) is to be performed?

These characteristics of the situation are requirements for the ontology matching process. There has been research attempting to refine such requirements. Mochol (2009) gave a very precise description of the type of ontologies to be matched depending on their size, expressiveness, language and role, e.g. domain ontology or upper-level ontology.

12.4.2 Finding Existing Alignments

Finding existing alignments which satisfy the need of the application is the second task. Alignments may be found on the web or through specialised directories. Reusing existing alignments should be privileged because of the cost of generating such alignments. For that purpose, the task of sharing (see Sect. 12.4.7) prepares alignment retrieving.

Ideally, alignments should come with annotations characterising their level of trustworthiness, the purpose for which they have been built and the type of relations they use.

These alignments must concern the ontologies to be matched, and they have to satisfy the constraints related to the alignment established in Sect. 12.4.1. In particular, correctness and completeness are criteria to use for selecting among various alignments.

These criteria may be assessed manually, on a sample, or can be inferred through the properties of their generation methods. In particular, one can use metadata attached to such alignments. They can reveal the method used for matching the ontologies (in particular, if these are automatic or manually generated alignments), they can cover manual assessments about the alignment (people publishing them can annotate the alignments to tell what they are good for) or they may contain indications of their intended use which can be matched with that of the current situation.

So, practically, selecting an alignment requires:

- Finding alignment repositories
- Finding those alignments between the ontologies to match
- Assessing the capacity of these alignments to address the needs previously identified, either based on metadata, or on the content of the alignments
- Deciding for one alignment based on this assessment

If apparently suitable alignments are available, the user can directly go to the validation task (Sect. 12.4.5). Otherwise, it is necessary to create a new alignment from the ontologies, as is explained in Sect. 12.4.3.

12.4.3 Selecting a Matcher

In order to build a new alignment, a suitable matcher has to be found. Many matchers have been developed over the years, and they provide different results on different types of data sets and matching contexts. Hence, the criteria elicited in the characterisation phase (Sect. 12.4.1) are also used for selecting the most appropriate matcher.

There have been several studies about how to choose a matcher depending on the characteristics of the ontologies and those of the expected alignments. They are worth taking into account.

Euzenat et al. (2006) provide a simple method for weighting matcher capabilities (speed, automaticity, precision and recall as measured in matcher evaluations) against the application requirements defined as the answers to the questions of Sect. 12.4.1.

The work of Mochol (2009) uses a deep classification of matchers and the matching context in order to assess which matcher will be more adapted to a particular context. This assessment is made using the Analytic Hierarchy Process (AHP) which guides the decision process of choosing a matcher. It can work on automatic or manual mode.

The OntoMas system (Huza et al. 2006) has been developed for helping and teaching how to carry out matching. For choosing a matcher, it processes a set of symbolic rules over a classification of tools and a characterisation of tasks.

The problem of such methods is that they require extensive information about available matchers which is not always available or always accurate when the assessment comes from the matcher developers. An important source of information is the result of the various matcher evaluation campaigns that have been run. The most important one is the Ontology Alignment Evaluation Initiative (OAEI) campaigns[1]. They have evaluated many matchers in a variety of situations. So their results can be taken into account when choosing a matcher. They are currently further developed in the context of the SEALS project[2].

So, in practice, choosing a matcher can be achieved by:

- Finding available matchers
- Assessing their capacity to generate alignments that fill the identified requirements by reading their documentation or comparing their performances in similar tasks during evaluations
- Deciding for one matcher based on this assessment

Other works try to automate this task, or the choice of matcher parameters, on the fly (Sayyadian et al. 2005). Such work can be used in runtime matching processes.

12.4.4 Matching Ontologies

The next task consists of running the matcher against the ontologies and collecting the resulting alignment. It may seem like the simplest task, methodologically speaking, because matchers have been designed exactly for this purpose.

But the user should not hesitate to run the matcher several times or to run several matchers, trying different sets of parameters and different thresholds. It is also useful to process matching incrementally by curating the returned alignment and feeding it again to the matcher for improving it.

In fact, all the procedure that can be applied at the enhancing phase (see Sect. 12.4.6) can also be directly applied during the matching phase without any prior evaluation.

Hence, this task can be further decomposed into a more complex sub-workflow (see Fig. 12.6). Section 12.4.6 provides some refinements of the matching workflow.

12.4.5 Evaluating Alignments

Once an alignment has been obtained, it is necessary to perform a final screening and validation. Evaluation can be applied on alignments that have been retrieved as

[1] http://oaei.ontologymatching.org

[2] http://www.seals-project.eu

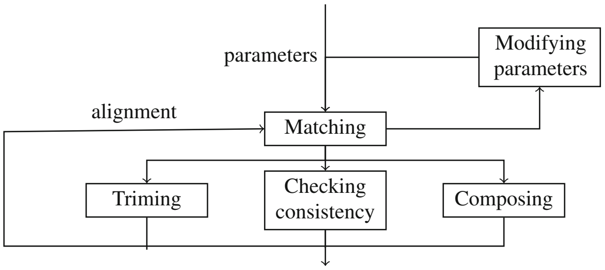


Fig. 12.6 The sub-workflow of fine-tuning matchers (all tasks but matching are optional). After matching, it is possible to apply automatically some alignment manipulation that can trim the alignment under a threshold, check and restore the consistency of an alignment or compose the alignment with another alignment. The result of these manipulations can be fed back as input to the matching operation or can be the final result of the workflow. Alternatively, it is possible to modify the parameters of the matcher and to run it again. These operations can be triggered manually or automatically

well. This task corresponds to the *evaluation* task of Fig. 12.2. Very precise methodological guidelines for evaluation of ontology networks are provided in Chap. 9 which may be applied here as well (note that the 'identify evaluation criteria and frame of reference' task corresponds to our 'identifying ontologies and characterising needs' task, see Sect. 12.4.1). We consider here what is specific to alignment evaluation during the matching activity. The evaluate/enhance loop in Fig. 12.5 corresponds to the feedback after evaluation in Fig. 12.2.

Evaluation consists of assessing properties of the obtained alignment. It can be performed either manually or automatically. Manual evaluation can be achieved by running a dry test of the final application or by asking an independent expert to assess the quality of the alignment and perform some manual assessment. For that purpose, graphical tools which allow to navigate quickly both in the alignment and in the ontologies are invaluable.

An often overlooked functionality of matching algorithms is their ability to give explanation for the provided alignments. Explanations can be obtained by interacting with the matcher or by accessing metadata about a stored alignment.

Automated quantitative evaluation can be performed by using techniques for evaluating alignments used in matcher evaluation campaigns such as OAEI[1] or SEALS[2]. These would require to extract samples from the results and computing measures like precision and recall which would provide an approximation of correctness and completeness.

There is no definitive answer as to what is a good result for evaluation. The evaluation must be performed so as to assess evaluation criteria. The characterisation of the problem (Sect. 12.4.1) aims at defining such success criteria. For some applications, high recall is required, while for some others, recall is not important. Moreover, the meaning of 'high' is not the same for all applications: A critical application which can break if some correspondence is missing will require 100% recall while a non critical application may be satisfied with 98%.

If the evaluation results are positive, i.e. the alignment satisfies these success criteria, then the obtained alignment can go through the next task, storing and sharing (Sect. 12.4.7); otherwise, the alignment can be improved (Sect. 12.4.6) before being input to the matcher and/or another matcher and/or parameters can be chosen.

12.4.6 Enhancing Alignments

Enhancement can be obtained either through manual modification of the alignment, e.g. with the help of an alignment editor, or the application of refinement procedures, e.g. selecting correspondences by applying thresholds. This enhancing task can lead to:

- The selection of another matcher, as in Fig. 12.5, by going back to Sect. 12.4.3
- The selection of another set of parameters to use with the same matcher, as in Fig. 12.6
- The manipulation of the alignment through trimming under a particular threshold or combining several alignments, as in Fig. 12.6

Among these procedures, the most straightforward one consists of trimming the alignment under some thresholds. There are many different ways to apply automatic thresholds (Euzenat and Shvaiko 2007). Some work has introduced double thresholding: Above the upper threshold, correspondences are selected, under the lower threshold, they are discarded, and the remaining correspondences are brought to the attention of the user (Lambrix and Liu 2009).

It may also restore consistency when the resulting alignment has been detected inconsistent in the evaluation (Sect. 12.4.5). By consistency checking, we do not necessarily mean logical consistency checking, but rather that the result does not violate particular constraints which may be:

- Acyclicity
- Syntactic anti-patterns (Roussey et al. 2009)
- Full logical consistency

Enhancing may then consist of selecting a subset of the correspondences in an alignment which satisfies the constraints. Algorithms developed in (Meilicke and Stuckenschmidt 2009) are particularly suited for that purpose.

Alignments obtained from various sources, such as other matchers or alignment libraries, may be composed in a single alignment through various operators: composition, meet, join and union.

12.4.7 Storing and Sharing

An extra task is to save and share the obtained alignment in a declarative format and to give it proper annotations to record its provenance and purpose. This task is very often overlooked but it is vital if one wants to find alignments in the corresponding task (Sect. 12.4.2): carefully annotating alignments will help others to reuse them. This task corresponds to the *communication* task of Fig. 12.2, and the dotted arrow in Fig. 12.5 corresponds to the availability of stored alignments after communication.

When an alignment is deemed worth publishing, then it can be annotated, stored and communicated to other parties interested in such an alignment.

Annotations of alignments should record the information that is useful for the 'finding existing alignment' task (Sect. 12.4.2). In particular, what is the purpose of this alignment, what is the assessment of its quality? Noy et al. (2008) discuss various kinds of metadata that are useful to record with correspondences. There are a few normalised vocabularies for doing this, and in particular the ontology metadata vocabulary (Hartmann et al. 2005). Other useful information like the algorithm used for computing it, the time it took or the source alignments and the date of matching can generally be recorded automatically.

Below is a sample of metadata associated with an alignment in the Alignment API:

dc:date	2009/10/23
align:method	fr.inrialpes.exmo.align.impl.methods.StringDistAlignment
align:time	421
omwg:purpose	Query mediation
align:creator	JohnDoe

while correspondence annotations can be:

align:measure	.7768
align:note	"manualy validated"

Storing an alignment requires some type of persistent storage. This is usually achieved through the use of a database management system, but a web site based on a file system may be sufficient. However, alignments must be properly indexed to retrieve them when necessary on various characteristics (one ontology, pairs of ontologies, arity, etc.). Indexing can be direct through a URI identifying alignments or indirect through queries looking for alignments based on their metadata. In general, it is preferable that both access modes be available.

Finally, these alignments may be shared by interested communities. For that purpose, they should be accessible on the web through HTML interfaces or web services.

There are several software supporting sharing alignments on the web. The Alignment server[3] and Cupboard (d'Aquin et al. 2009) are general-purpose servers providing alignments in the Alignment format. BioPortal[4] is specialised in biomedical ontologies and provides individual correspondences (called mappings in this system).

[3] http://alignapi.gforge.inria.fr

[4] http://bioportal.bioontology.org

12.4.8 Rendering Alignments

Finally, the alignment is transformed into another form or interpreted for performing actions like mediation or merging.

This task corresponds to the *exploitation* task of Fig. 12.2. It is the natural outcome of matching. The exploitation of the alignment may be denoted by a different activity name, e.g. merging or query translating, taking directly the alignment as input. However, it may happen that ontology matching is considered as an activity in itself in which case it will deliver its output in an appropriate format for another task. This is what is called 'rendering'.

Rendering may deliver the alignment as such in RDF in order to be processed by an interpreter such as a query mediator. But it also can be transformed, as discussed in Sect. 12.3, into OWL axioms, SKOS relations or sets of `owl:sameAs` statements.

The dotted arrow on Fig. 12.2 expresses the feedback after using the alignment which may contribute to enhance it.

12.5 Support for Matching Ontologies

We think that methodological guidelines are more useful and better accepted if they are supported by tools rather than delivered as rules to be applied. So far, existing support is available in the alignment manipulation part rather than the requirement analysis part.

12.5.1 Independent Tools

Some tools offer alignment manipulation that can be used for alignment enhancement (Sect. 12.4.6).

Foam (Ehrig 2007) is a framework in which matching algorithms can be integrated. It mostly offers matching and processor generation. It does not offer online services or alignment editing, but it is available as a Protégé plugin and has been integrated in the KAON2 ontology management environment. COMA++ (Aumüller et al. 2005) and Harmony (Mork et al. 2008) are stand-alone (schema) matching work benches that allow for integrating and composing matching algorithms. They support matching, evaluating, editing, storing and, for COMA++, processing alignments.

The Alignment server, associated with the Alignment API[5] (David et al. 2011), offers matching ontologies, manipulating, evaluating, storing and sharing alignments as well as processor generation. It can be accessed by clients through an API, web services, agent communication languages or HTTP. It does not support editing.

Most of the software developed for editing alignments are candidates for design time matching. The same alignment editor can be used for manipulating more precisely the obtained alignments. They should provide a convenient display of the currently edited alignments and the opportunity to discard, modify or add correspondences. Ideally, each design time function should be available from an alignment editor. Since ontologies and alignments can be very large, it is very challenging to offer intuitive alignment editing support. There exists such editor prototype such as OnaGui[6] or iMerge (El Jerroudi and Ziegler 2008).

12.5.2 Integrated Tools

Model management has been promoted in databases for dealing with data integration in a generic way. It offers a high-level view to the operations applied to databases and their relations. Rondo[7] is such a system (Melnik et al. 2003). It offers operators for generating the alignments, composing them and applying them as data transformation. It is a stand-alone programme with no editing functions.

Integrated tools integrate alignment management to ontology management.

The Web Service Modeling Toolkit (WSMT) (Kerrigan et al. 2007) is an Integrated Development Environment (IDE) for Semantic Web services which also provides ontology engineering capabilities. Among other capabilities, WSMT proposes a set of tools for manually creating, editing and storing ontology alignments. It offers a set of methods and techniques that assist ontology engineers in their work, such as different graphical perspectives over the ontologies, suggestions of the most related entities from the source and target ontology and guidance throughout the matching process (Mocan et al. 2006). WSMT and the ontology engineer work together in an iterative process which involves cycles consisting of suggestions from the tool side and validation and creation of correspondences from the user side.

Protégé is an ontology edition environment which offers design time support for matching. In particular, it features Prompt[8] (Noy and Musen 2003), an environment that provides some matching methods and alignment visualisation. Prompt allows to match, compare and merge ontologies. Since alignments are expressed in an

[5] http://alignapi.gforge.inria.fr

[6] http://sourceforge.net/projects/onagui/

[7] http://infolab.stanford.edu/modman/rondo/

[8] http://protege.stanford.edu/plugins/prompt/prompt.html

ontology, they can be stored and shared through the Protégé server mode. Prompt can be extended through a plugin mechanism.

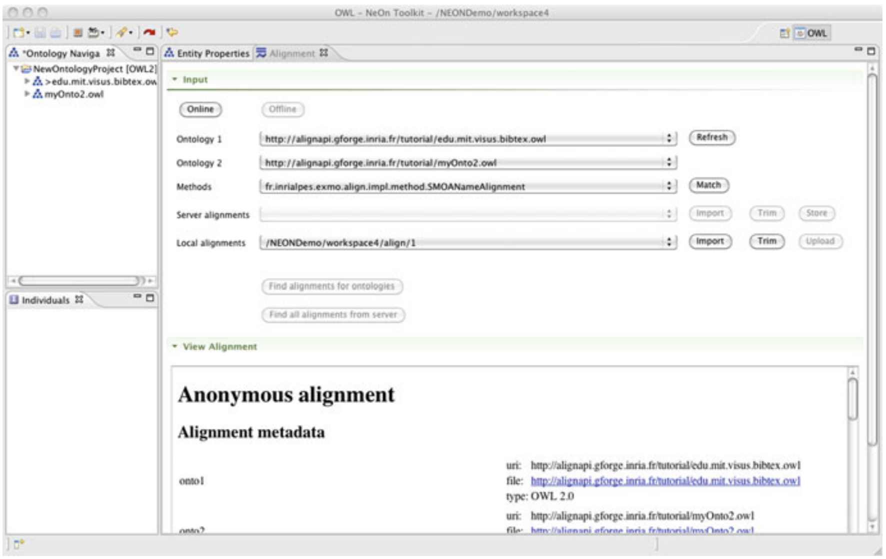
12.5.3 The NeOn Toolkit Alignment Plugin

The NeOn[9] project produced a toolkit for ontology management (see Chap. 13) which features runtime and design time ontology alignment support.

NeOn supports ontology alignments in both the NeOn Toolkit and the Cupboard ontology server.

The NeOn Toolkit Alignment plugin works in two modes: an offline mode in which the user can work locally on the alignments. The user can run the matchers which are embedded in the toolkit against ontologies in the NeOn Toolkit and manipulate alignments which are in the local environment. Figure 12.7 shows two selected ontologies and a matching method that can be applied to them. It also shows a local alignment between these two ontologies to which operations such as trimming under a threshold or rendering in OWL ('import') can be applied.

The online mode connects the NeOn Toolkit to an alignment server allowing to share ontologies and to apply these operations on alignments stored on the server.



Ontology 1 http://alignapi.gforge.inria.fr/tutorial/edu.mit.visus.bibtex.owl

Ontology 2 http://alignapi.gforge.inria.fr/tutorial/myOnto2.owl

Methods fr.inrialpes.exmo.align.impl.method.SMOANameAlignment

Server alignments

Local alignments /NEONDemo/workspace4/align/1

Anonymous alignment

Alignment metadata

onto1	uri: http://alignapi.gforge.inria.fr/tutorial/edu.mit.visus.bibtex.owl file: http://alignapi.gforge.inria.fr/tutorial/edu.mit.visus.bibtex.owl type: OWL 2.0
onto2	uri: http://alignapi.gforge.inria.fr/tutorial/myOnto2.owl file: http://alignapi.gforge.inria.fr/tutorial/myOnto2.owl

Fig. 12.7 The NeOn Toolkit Alignment plugin interface

[9] http://www.neon-project.org

Of course, alignments can move back and forth between the server and the local environment.

Both online and offline modes provide the functions of the Alignment API: retrieving alignments, matching ontologies, trimming alignments under various thresholds, storing them in permanent stores and rendering them in numerous output formats. These operations support the whole alignment life cycle (Fig. 12.5).

The Alignment plugin allows one to automatically compute and manage ontology alignments. More precisely, it offers the following functionalities:

- Find alignments between ontologies or those available on the server
- Match ontologies
- Trim alignments by applying thresholds to existing alignments
- Retrieve and render alignments in a particular format
- Upload and store an alignment permanently on the server

Alignments stored in the server can be further shared through the Cupboard ontology server. It allows for indexing alignments available from alignment servers. Hence, these alignments can be available to each Cupboard user to be stored in her cupboard and, as for ontologies, be rated and annotated. Cupboard provides direct access to alignments as well as indirect access to the Alignment server to generate new alignments when they are missing.

12.6 Examples

In this section, we consider a user having to connect an ontology designed for drug and prescription to existing ontologies outside. These examples are closely related to the application presented in Chap. 20.

12.6.1 Identifying Needs

More precisely, the newly proposed Semantic Nomenclature ontology (presented in Chap. 20), designed from schemas of pharmacological firm databases, has to be matched to ontologies available on the web. This could help searching for literature about concerned drugs or exporting drug interactions as linked data for other applications to take them into account.

The requirements for this matching activity allow it to be performed offline, without time constraints, so the use of the NeOn Toolkit and user supervision is perfectly suited. The ontologies, having been developed independently and for different purposes, are not expected to match exactly. Correct correspondences are expected, completeness is secondary. The type of operation to be performed with the resulting alignments is data export (for exposing linked data) and query translation (for connecting to the literature).

Table 12.1 Characteristics of the considered ontologies

Organisation	Ontology	Lang.	Form.	Classes	Relations	Properties
ATOS	Semantic Nomenclature	English	OWL	67	20	26
UMLS	Semantic Network	English	OWL	199	105	0
LDIS	DrugOnt	English	OWL	28	26	32
NLM	RX nomenclature	English	OWL	10	16	0

12.6.2 *Identifying Ontologies*

Watson (d'Aquin and Motta 2011) allows for finding further ontologies that may be useful. These are:

- The LDIS Drug ontology[10] which has been designed for prescription application in hospitals (related with electronic patient record)
- The UMLS Semantic Network ontology[11] which is well used for literature indexing because of its extensive coverage
- The RxNorm ontology[12] which is used for classifying drugs and is suited to search the literature

A quick study of these ontologies shows the characteristics displayed in Table 12.1.

More ontologies on this topic are available, and a comparison can be found in Herrero Cárcel and Pariente (2009).

The ontologies are relatively homogeneous being in English (with some Spanish comments in Semantic Nomenclature) and OWL. They have comparable sizes with the notable exception of UMLS.

12.6.3 *Finding Existing Alignments*

Finding available alignments may be achieved by using an alignment server. In the present case, there is no alignment available between these ontologies.

12.6.4 *Selecting a Matcher*

The user then proceeds by selecting a matcher suited to match these ontologies. In this case, given that ontologies are about a very close and normalised domain, they

[10] http://lsdis.cs.uga.edu/projects/asdoc/DrugOnt_schema.owl

[11] http://swpatho.ag-nbi.de/owldata/umlssn.owl

[12] http://www.nlm.nih.gov/research/umls/rxnorm/

are written in the same natural language; the user may select very simple matchers based on the strings naming entities. There are several matchers available, either under the NeOn Toolkit or the Alignment server; the best way is to try them and to see the results (see Sect. 12.6.5).

In a second iteration, tests have been performed with more elaborate matchers such as a simple use of WordNet which would use synonyms to match terms and distance in the hypernym graph. Or it can use the Aroma matcher which will attempt at determining association rules between concepts before extracting an alignment between them (David et al. 2007).

12.6.5 Matching

The simple StringDistAlignment method with different string distances is run, and results are displayed in Table 12.2.

The user first ran the method with Levenshtein measure (edit distance) and SMOA measure which tries to better interpret the way people label things, e.g. by using syntactic variations. The threshold has been put to .75 so as to avoid considering far-reaching similarity between strings. Later, a threshold of .85 has been applied in order to further ensure correctness (because a higher threshold will eliminate unlikely matches).

12.6.6 Evaluating

There is no automatic way to evaluate these results. They have to be manually looked into by the user to assess their quality (they can be displayed by alignments editors).

Concerning the drug ontology, the small returns with the Levenshtein distance are obviously correct. The use of SMOA provides mostly new correct matches, such as interacts/hasInteraction. The only non–fully correct matches are the matching of DrugInteraction and OtherInteraction to interaction. Using SMOA with a .75 threshold provides reasonable results. Some more matches, such as

Table 12.2 Number of matched entities in Semantic Nomenclature depending on matching method and threshold. The example has used two iterations displayed by the horizontal line

Method	Threshold	DrugOnto	RxNorm	UMLSSN
Levenshtein	.85	3	3	9
–	.75	3	4	11
SMOA	.85	8	8	20
–	.75	11	11	34
WordNet	.75	5	6	12
Aroma		8	5	30

isIndicated/has_indication_text, could have been found, but not many. The small number of matches can be explained as follows: Semantic Nomenclature is more oriented towards the drug production and commerce processes, while the drug ontology is targeting the consumption process.

Concerning UMLSSN and RxNorm, Levenshtein was better than SMOA which was returning quite a lot of unwanted matches, such as isProducedBy/produces or Clinical_Finding/Clinical_Drug. After a closer examination, there is no real reason to find more correspondences than those provided by Levenshtein, so the user may want to use these. Especially with UMLSSN, it seems that the labels have been chosen so that they correspond to those of UMLS so they exactly match.

Using the more elaborate matchers has confirmed this. They have only returned plausible but not necessarily valid correspondences, such as Physical_Entity/ Physical_Object.

12.6.7 Enhancing

Enhancement may be achieved by two means: either by manual edition of the resulting alignment or by running a new matcher, using new parameters or applying different threshold to the results. This is what has been done by using different thresholds and testing the more elaborate matchers, i.e. starting back to Sect. 12.6.4. In the end, once the SMOA alignment with the drug ontology has been found acceptable with respect to other results, this alignment is manually edited and selected.

Both means can be interleaved: It is possible to edit an alignment and to use it as further input for a matcher.

12.6.8 Storing and Sharing

Once an alignment of sufficient quality is established, especially if it has been curated by hand, it must be better documented, for instance, by adding metadata explaining how it has been obtained, who has curated it and what is the reached confidence in each correspondence. This is illustrated in Sect. 12.4.7. Then, it can be uploaded to an Alignment server so that it would be visible to other people (in the previous step of Sect. 12.6.3).

12.6.9 Rendering

Finally, the obtained alignments have to be used. We have considered that the data expressed in the Semantic Nomenclature ontology could be converted in the drug ontology so as to communicate critical information about interaction. This may be achieved either by generating an XSLT transformation applying to the data expressed in XML for obtaining the interactions under the drug ontology or a more elaborate process may take advantage of the alignment to generate links between instances of both ontologies.

On the other side, if RxNorm or UMLSSN is used to query bibliographical databases, the alignments may be used for translating queries expressed with respect to the Semantic Nomenclature into queries expressed in the two other ontologies and eventually evaluate them in parallel.

12.7 Conclusions

Establishing relations between ontology entities is part of modern ontology engineering and a very important activity for networked ontology engineering. This activity remains difficult though there are many solutions for carrying it out. We proposed methodological guidelines for ontology matching which integrates with the alignment life cycle and can cooperate with ontology engineering methodologies. In particular, we paid a particular attention to alignment sharing and reuse. These guidelines are based on research work on particular tasks: Some of these have been investigated in depth and others have not. Similarly, some tools cover parts of these guidelines, but none is able to support them entirely.

Hence, more work is necessary to achieve a fully instrumented ontology matching methodological support, and no doubt it will raise some demands for improvement in the proposed methodological guidelines.

12.8 Further Readings

There are few methodological accounts of ontology matching. Mochol (2009) is the exception: a whole thesis dedicated to matcher selection. Corcho (2005) has considered more specifically the methodology for designing an ontology translation method, including a matcher. Euzenat and Shvaiko (2007) covers many facets of ontology matching, but not extensively methodology. It provides insights of most of the tasks of the above methodological path. Euzenat et al. (2008) is more methodological but not focussed on the individual act of matching.

Acknowledgements We thank Pavel Shvaiko for his comments on a previous version of this chapter. This work has been partly supported by the European Commission IST project NeOn (IST-2006-027595).

References

Aumüller D, Do H-H, Maßmann S, Rahm E (2005) Schema and ontology matching with COMA++. In: Proceedings of the 24th international conference on management of data (SIGMOD), software demonstration, Baltimore, MD, USA, pp 906–908

Corcho Ó (2005) A layered declarative approach to ontology translation with knowledge preservation. Ios Press, Amsterdam

Corcho Ó, Gómez-Pérez A (2007) ODEDialect: a set of declarative languages for implementing ontology translation systems. J Univers Comput Sci 13(12):1805–1834

d'Aquin M, Motta E (2011) Watson, more than a semantic web search engine. Semant Web J 2:55–63

d'Aquin M, Euzenat J, Le Duc C, Lewen H (2009) Sharing and reusing aligned ontologies with cupboard. In: Proceedings of 5th ACM KCap poster session, Redondo Beach, CA, USA, pp 179–180. URL ftp://ftp.inrialpes.fr/pub/exmo/publications/daquin2009a.pdf

David J, Guillet F, Briand H (2007) Association rule ontology matching approach. Int J Semant Web Inf Syst 3(2):27–49

David J, Euzenat J, Scharffe F, Trojahn dos Santos C (2011) The Alignment API 4.0. Semant Web J 2(1):3–10. URL http://iospress.metapress.com/content/4164891n48p5v826/

Ehrig M (2007) Semantic web and beyond: computing for human experience. In: Ontology alignment: bridging the semantic gap. Springer, New York. Acitrezza, Italy, ISBN 0–387–32805-X

El Jerroudi Z, Ziegler J (2008) iMERGE: interactive ontology merging. In: Proceedings of the 16th EKAW demonstration track, Acitrezza, Italy, pp 52–56

Euzenat J (2004) An API for ontology alignment. In: Proceedings of 3rd international semantic web conference (ISWC), Hiroshima, Japan, Lecture notes in computer science, vol 3298. Springer, Berlin/Heidelberg, pp 698–712

Euzenat J, Shvaiko P (2007) Ontology matching. Springer, Heidelberg

Euzenat J, Ehrig M, Jentzsch A, Mochol M, Shvaiko P (2006) Case-based recommendation of matching tools and techniques. Deliverable 1.2.2.2.1, knowledge web. URL ftp://ftp.inrialpes.fr/pub/exmo/reports/kweb-126.pdf

Euzenat J, Mocan A, Scharffe F (2008) Ontology alignment: an ontology management perspective. In: Hepp M, De Leenheer P, De Moor A, Sure Y (eds) Ontology management: semantic web, semantic web services, and business applications. Springer, New York, pp 177–206

Hartmann J, Palma R, Sure Y, Haase P, Suárez-Figueroa MC, Haase P, Gómez-Pérez A, Studer R (2005) Ontology metadata vocabulary and applications. In: Meersman R, Tari Z, Herrero P et al (eds) Proceedings of the International conference on ontologies, databases and applications of semantics (ODBASE-2005), Lecture notes in computer science, vol 3762. Springer, Berlin/Heidelberg/New York, pp 906–915

Herrero Cárcel G, Pariente T (2009) Revision of ontologies for semantic nomenclature: pharmaceutical networked ontologies. Deliverable 8.3.2, NeOn project

Horrocks I, Patel-Schneider P, van Harmelen F (2003) From SHIQ and RDFto OWL: the making of a web ontology language. J Web Semant 1(1):7–26

Huza M, Harzallah M, Trichet F (2006) OntoMas: a tutoring system dedicated to ontology matching. In: Proceedings of the 1st ISWC international workshop on ontology matching (OM), Athens, GA, USA, pp 228–323

Kerrigan M, Mocan A, Tanler M, Fensel D (2007) The web service modeling toolkit – an integrated development environment for semantic web services. In: Proceedings of the 4th European semantic web conference (ESWC) system description track, Innsbruck, Austria, pp 303–317

Meilicke C, Stuckenschmidt H (2009) An efficient method for computing alignment diagnoses. In: Proceedings of the 3rd international conference on web reasoning and rule systems (RR-2009), Chantilly, VA, USA, pp 182–196

Melnik S, Rahm E, Bernstein P (2003) Rondo: a programming platform for model management. In: Proceedings of the 22nd international conference on management of data (SIGMOD), San Diego, CA, USA, pp 193–204

Miles A, Bechhofer S (2009) SKOS simple knowledge organization system: reference. Recommendation, W3C. URL http://www.w3.org/TR/skosreference

Mocan A, Cimpian E, Kerrigan M (2006) Formal model for ontology mapping creation. In: Proceedings of the 5th international semantic web conference (ISWC), Athens, GA, USA, Lecture notes in computer science, vol 4273. Springer, Berlin/Heidelberg/New York, pp 459–472

Mochol M (2009) The methodology for finding suitable ontology matching approaches. PhD thesis, Freie Universität Berlin. URL http://www.diss.fuberlin.de/diss/receive/FUDISS_thesis_000000008124

Mork Peter, Seligman Len, Rosenthal Arnon, Korb Joel, Wolf Chris (2008) The harmony integration workbench. J Data Semant XI:65–93

Noy N, Musen M (2003) The PROMPT suite: interactive tools for ontology merging and mapping. Int J Hum-Comput Stud 59(6):983–1024. ISSN: 1071–5819. doi:http://dx.doi.org/10.1016/j.ijhcs.2003.08.002

Noy N, Griffith N, Musen M (2008) Collecting community-based mappings in an ontology repository. In: Proceedings of the 7th international semantic web conference (ISWC), Karlsruhe, Germany, pp 371–386

Patrick Lambrix, Qiang Liu (2009) Using partial reference alignments to align ontologies. In: Proceedings of the 6th European semantic web conference (ESWC 2009), Heraklion, Germany, Lecture notes in computer science, vol 5554. Springer, Berlin/Heidelberg/New York, pp 188–202

Roussey C, Corcho Ó, Vilches Blázquez LM (2009) A catalogue of owl ontology antipatterns. In: Proceedings of the 5th international conference on knowledge capture (KCap-2009), Redondo Beach, CA, USA, pp 205–206

Sayyadian M, Lee Y, Doan A-H, Rosenthal A (2005) Tuning schema matching software using synthetic scenarios. In: Proceedings of the 31st international conference on very large data bases (VLDB), Trondheim, Norway, pp 994–1005