GOALS – A TEST-BED FOR ONTOLOGY MATCHING

Paulo Maio and Nuno Silva
GECAD – Knowledge Engineering and Decision Support Group
School of Engineering – Polytechnic of Porto
Porto, Portugal
{pam,nps}@isep.ipp.pt

Keywords: Ontology, Matching, Mapping, Alignment

Abstract: State-of-the-art ontology matching systems rely on the combination of basic matching techniques but good results are only achieved when processing particular classes of ontologies. Furthermore, they are quite restrictive with respect to their internal configuration, as they are committed to a pre-defined architecture and workflow. Additionally, the skilful selection of matchers and the respective combination and configuration process is difficult and time consuming. Additionally it is hard to test and evaluate. This paper presents a test-bed system that eases the creation of new matching systems. It promotes the reusability, the combination and the configuration of existing matchers, encouraging the development of new matching algorithms able to fill specific open matching gaps exploiting existing methods and algorithms.

1 INTRODUCTION

Ontologies are artifacts that provide a shared vocabulary and its meaning about a domain of interest that can be conveyed between people and application systems (Berners-Lee, Hendler, & Lassila, 2001). Because ontologies are targeted and fitted to describe the structure and the semantics of information, they play a key role in many application scenarios, such as the Semantic Web (Berners-Lee et al., 2001), Knowledge Management and e-commerce (Fensel, 2001), information integration (Halevy et al., 2005) and peer-to-peer systems (Staab & Stuckenschmidt, 2006).

Yet, because different entities adopt different ontologies for their descriptions, heterogeneity problems arise between communication partners. Ontology matching, also known as ontology mapping, is perceived as an appropriate approach to overcome this terminological and semantic gap (Euzenat & Shvaiko, 2007).

Ontology matching is the process whereby an alignment is established between a source and target ontology, i.e. a set of correspondences between semantically related ontology entities (e.g. concepts, properties, instances) of different but overlapping ontologies (Euzenat & Shvaiko, 2007). However, the alignment specification is a time consuming and knowledge demanding task, whose result is error prone even when domain experts are part of the process (Doan, Madhavan, Domingos, & Halevy, 2004).

Despite an impressive number of research initiatives in the matching field, containing valuable ideas and techniques, current matching approaches are (implicitly) restricted to processing particular classes of ontologies and thus they are unable to guarantee a predictable quality of results on arbitrary inputs (Mochol, Jentzsch, & Euzenat, 2006). Furthermore the current trends in research suggest solving small parts of "global" problems in the matching field or fill some open matching gaps (Fürst & Trichet, 2005). Yet, when implementing a new matching system, the corresponding algorithm is typically built from scratch and no attempt to reuse existing methods is made (Mochol et al., 2006). Additionally, existing approaches usually act as a black-box and consequently they are quite restrictive with respect to their internal configuration, allowing one to set up some parameter values only (e.g. thresholds and weights of constituent matching techniques).

It is our conviction that the combination of simple matching algorithms into more complex ones and consequently into more skillful systems is the correct approach. However, this encompasses a combination and configuration process of matchers that is very difficult and time consuming, hard to test and evaluate.

This paper presents our approach to overcome these problems by proposing and describing a test-bed system that eases the creation of new matching systems, promoting the reusability, the combination and the configuration of existing matchers, but also encouraging the development of new matching algorithms able to fill specific open matching gaps in cooperation with existing methods.

The rest of this paper is organized as follows: the next section introduces ontology matching techniques and the composition process. Section 3 presents our test-bed for ontology matching, which is complemented in section 4 with an example of use. Finally, section 5 draws conclusions and comments on future work.

2 ONTOLOGY MATCHING

Basic matching techniques (referred to as matchers) are grouped into two distinct groups: (i) those where focus is at the element-level (e.g. string-based and language-based methods) and (ii) those that put the focus at the structure-level (e.g. taxonomy-based and graph-based methods). Semantic grounded methods can be focused at element-level (e.g. sameClassAs) or in the structure-level as S-Match (Giunchiglia, Shvaiko, & Yatskevich, 2004). Independently of which algorithm is used, the result is a set of mapping elements (also referred to as matches or correspondences), where each match is a 5-tuple: < *id, e, e', R, n* > where *id* is a unique identifier, *e* and *e'* are source and target ontology entities (respectively), *R* is a relation (e.g. equivalence, more general) and *n* is a confidence value, typically in the [0-1] range.

State-of-the-art ontology matching systems (Euzenat & Shvaiko, 2007; OAEI'2008, 2008) apply at least two of these basic matching techniques yielding different and complementary competencies, to achieve better results. In the following, we introduce relevant strategies used by those systems to combine matchers.

From an architectural perspective, systems follow two distinct approaches: (i) a sequential approach, i.e. a matcher computes a set of correspondences (referred to as similarity matrix) used to seed the next matcher and so on (the process includes as many matchers as needed); (ii) a parallel approach, i.e. each matcher individually computes one similarity matrix, whose results are aggregated through a function into one single matrix. It should be noticed that both approaches can co-exist in the same system (hybrid architecture).

Several aggregation functions might be used in the parallel approach. The most common and popular functions are min, max, linear average and weighted average (Ji, Haase, & Qi, 2008). However, weight-based functions have two major drawbacks with respect to their definition: (i) the weights must be set up by the user or (ii) they must be learned through machine learning (ML) methods. Recently, the ordered weight average (OWA) operator has been proposed (Ji et al., 2008), which instead of associating weights to specific similarity measures (matchers) it suggests weighting the similarity value according to its (ordered) relative position. That is, the best match is weighted differently (weight 1) than the second best match (weight 2) and so on (weight n).

Besides the adopted architecture, the result of matching is a large set of correspondences. Therefore, the satisfactory set of correspondences that will be part of the resulting alignment remain to be extracted. This is the role of specialized extraction methods, which acts on similarity matrices or on some pre-alignments already extracted. Methods applying thresholds are seen as the simplest approach. Several kinds of thresholds have been identified and are summarized in (Euzenat & Shvaiko, 2007). Here, the big issue is to find out the right threshold value. However, more complex methods such as strengthening and weakening functions (e.g. sigmoid functions) proposed by (Ehrig & Sure, 2004) the local optimization methods (also known as stable marriage) or global optimization methods might be applied too. Moreover, *a priori* knowledge about the cardinality of expected resulting alignment might be useful.

Despite the matching systems allowing some parameterization, such as (i) threshold values, (ii) matcher weights (for a specific aggregation function) or even (iii) choosing the list of matchers to participate in the alignment; the fact is that they are restrictive with respect to the internal working of the systems. That is, one cannot set up the architecture of the system and the corresponding workflow. Furthermore, since matchers should not be chosen only with respect to the given data but also adapted to the problem to be solved, the selection of the most suitable matcher is still an open issue.

Another important aspect of ontology matching is assessing the quality of resulting alignment. For that purpose, measures can be classified as (i) compliance measures and (ii) formal or logic-based measures. Compliance measures are those that compare system outputted alignment with a reference alignment (or gold standard) which should be the complete set of all correct correspondences. The most used are Precision and Recall (originating from information retrieval), or their harmonic mean, referred to as F-Measure. Precision corresponds to the ratio of correctly found correspondences over the total number of found correspondences while Recall corresponds to the ratio of correctly found correspondences over the total number of expected correspondences. Yet, in order to improve these measures, a Relaxed Precision and Relaxed Recall have been proposed (Ehrig & Euzenat, 2005). These are based on the idea that a proposed correspondence not existing in the reference alignment might be similar to an existing one. Instead of considering it incorrect one can measure the correction effort to transform such correspondence into a correct one. Semantic Precision and Semantic Recall are formal measures based on the comparison of deductive closure of both alignments (i.e. proposed and reference alignment) instead of a syntactic comparison (Euzenat, 2007). Recently, a set of logic-measures based on the incoherence of correspondences has been proposed (Meilicke & Stuckenschmidt, 2008). But a major drawback to all of these measures (except incoherence-based ones) is that they are grounded in the existence of one reference alignment which might not be available in real-world scenarios. It should be noted that, there are other measures concerned with resource consumption (e.g. speed, memory, scalability), referred to as performance measures, that can be used to compare systems instead of the resulting alignments.

Given the impressive number of existing matching techniques, their diversity and all the resulting combination possibilities, a tool that eases the matching combination process and the respective evaluation through a simple, fast and flexible way without being committed to a pre-defined workflow is required.

3 THE GOALS TEST-BED

After the characterization and systematization of the ontology matching technologies described in the previous section, the following dimensions have been identified and represent the core concepts of our GOALS (**G**ECAD **O**ntology **AL**ignment **S**ystem) matching test-bed: (i) data-entities, (ii) components, (iii) workflow specification and (iv) the execution engine. Data entities represent any kind of data structures that components manipulate as input and/or output. Currently, the system supports three different data entities: (a) OntModel which corresponds to one ontology, (b) Matrix which corresponds to a set of matches (or correspondences) and (c) an Alignment which corresponds to a set of mappings. Main difference between Matrix and Alignment is grounded in the notion of match and mapping. While one match establishes a relation (typically '=', '<', '>') between one source ontology entity and one target ontology entity with a given confidence value (e.g. 'firstName' is '<' than 'name' with a confidence value of 0.8), one mapping establishes a more complex relation (might even be language dependent) between one set of source ontology entities and one set of target ontology entities (e.g. 'name' = 'firstName' + " " + 'lastName' with a confidence value of 0.95). Therefore, and according to Euzenat terminology (Euzenat, 2004) any Matrix might be converted in one Alignment of level 0. However, GOALS allows level 1 and 2 alignment extraction. Components (or actors) are objects acting as black boxes that play one or more roles in the ontology matching process. Each component explicitly defines a set of shared and common data entities as inputs and outputs and a specific functionality. Particularities of each component are configurable through a set of parameters. Workflow specification is about choosing which components take part in the matching process, its parameters and roles and how data entities flow between components. The result of a workflow specification is a new complex matcher, referred to as the meta-matcher. The execution engine is able to tackle the workflow specification and automatically run the meta-matcher configuration. An important aspect of the system is that it is not limited or committed to any built-in component. In fact, instead of built-in components there is a well established Java API that any component must implement. To ease the connection between GOALS API and external systems (e.g. matchers) the GoF Adapter pattern (Gamma, Helm, Johnson, & Vlissides, 1994) is applied. Thus,

development of components is independent of the overall system. By exploiting this feature, most of the available components are implemented as adapters. Current GOALS release provides adapters for several well know matchers such as Falcon-OA (Jian, Hu, Cheng, & Qu, 2005) or FOAM (Ehrig & Sure, 2005).
Several generic components were identified and categorized according to their input/output data entities, which constrain the possible combination of components, and therefore the workflow. Figure 1 depicts all possible flows (arrows) of data entities (parallelograms) between the generic components (rectangles).

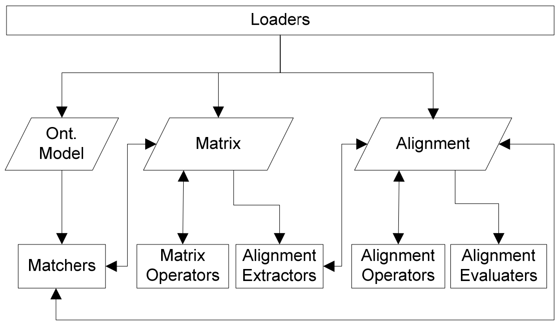

Figure 1: Generic components data flow.

Loader components are those that load any kind of data entity from a given location (set by URI parameter). Matcher components represent any basic or complex matching algorithm whose output is a set of correspondences, i.e. one Matrix or one Alignment between two input ontologies. Deductive matchers might need one seeding Matrix or Alignment too. Matrix operators are those that receive one or more Matrices, process them in some way and the output is one Matrix. Typically, these methods are classified as filters (e.g. applying one threshold) or as aggregators (e.g. applying an OWA operator). Similar to these components are the Alignment operators but instead of working with matrices they work with alignments. Alignment extractors components are responsible for extracting one Alignment from a set of Matrices and/or from a set of pre-Alignments. Alignment evaluators are components that compare one alignment (typically the result of meta-matcher) with another one (referred to as the reference alignment) applying evaluation metrics (e.g. precision, recall). Furthermore, the evaluation results might help one change and refine the meta-matcher specification in order to achieve better results. Note that, other component that do not fit in the above generic component description might also exist in the system and be included in a workflow specification. In that sense, we point out the components that are responsible for data-entities adaptation, i.e. converting a given data-entity from the common representation format to the appropriate format required for an external method or vice-versa.

The workflow infrastructure exploits components' common interface to query each component about required and optional input data entities and respective output data entities in order to ensure data flow executability. Furthermore, because this infrastructure is completely configurable through an XML file (i.e. script mode) or programmatically, one might also create workflow-based components in order to ease and improve the meta-matcher specification. This is made through a special component that encapsulates any previously defined workflow and infers required input and output data entities. As such, any meta-matcher will act as any other matcher component. Therefore, these components need to request that the execution engine run their internal workflow before providing their outputs to the main workflow.

The execution engine tackles any workflow specification through the following steps: (i) reads the list of components that are present in the workflow creating one instance for each component; (ii) each component instance knows what components responsible for providing their input data entities are; (iii) particular parameters of each component instance are set up with the defined values in the workflow specification phase. When these steps are complete, the execution engine is ready to run, i.e. execute the meta-matcher. Two running options are available: (i) run the entire meta-matcher or (ii) run the meta-matcher partially to a given stop-point (component) of the workflow. However, both options are grounded in the notion of terminator components, i.e. components whose output is not used as input of any other component. In that sense, while the first option automatically identifies the terminator components through a workflow inspection, the second option temporarily considers as unique terminator component the component used as the stop-point. Finally, a running command is sent to each terminator component, previously identified, which must adopt a pull-strategy behaviour to get their required inputs and further process those inputs in order to generate the intended outputs. Note that, each component knows which components are responsible for providing their inputs and therefore it is up to the components to request their own inputs.

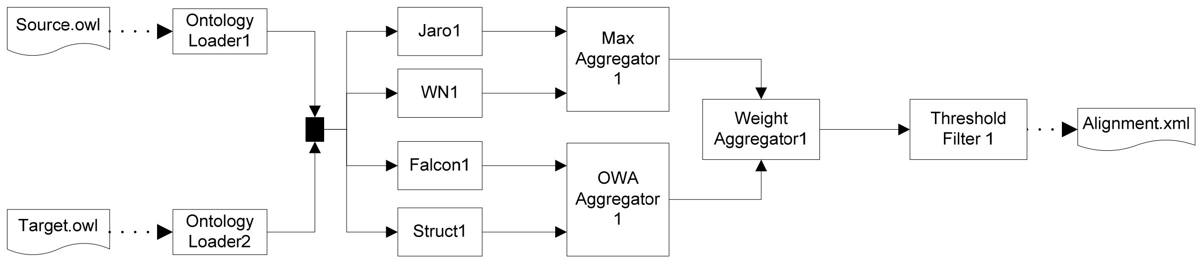

Figure 2: GOALS meta-matcher example.

4 WORKED EXAMPLE

In order to better explain our approach we present a worked example. Consider the scenario of a matching process intended to align a pair of ontologies using several matchers, to combine matchers' results through aggregation techniques and finally to obtain an alignment filtering aggregation's result by the means of a threshold (depicted on Figure 2).

The following components will be used to model this scenario:

- OntologyLoader1 and OntologyLoader2 to load source and target ontologies respectively from a resource location (e.g. Source.owl and Target.owl). Both are instances of the same component, i.e. "OntologyLoader";
- Jaro1, WN1, Falcon1 and Struct1, each representing an instance of a specific matching algorithm able to generate the alignment between both ontologies. Such techniques are Jaro of similarity package SimPack, WordNet Synonyms (Miller, 1995), Falcon-AO (Jian et al., 2005) and ClassStructure from the INRIA AlignAPI project respectively. Notice that, instead of these concrete algorithms, any other matching technique could be used;
- MaxAggregator1 to combine similarity matrices generated by Jaro1 and WN1 components. It is an instance of an aggregation technique that uses the max function to combine several matrices into one;
- OWA-Aggregator1 is an instance of an aggregation technique that uses the ordered-weight average function to combine similarity matrices into one. It requires as many weight-values as inputs matrices. In this case, there are two input matrices generated by Falcon1 and Struct1 components. Those values might be manually set by users or automatically proposed by the system;
- WeightAggregator1 is an instance of an aggregation technique that uses the weighted-average function to combine several matrices into one. One specific weight-value might be set for each input Matrix. In our example, it combines similarity matrices generated by MaxAggregator1 and OWA-Aggregator1 components;
- ThresholdFilter1 is an instance of the filtering-by-threshold method to filter WeightAggregator1's results applying a given threshold value (e.g. 0.75). Further, it saves filtered results to a physical location (e.g. Alignment1.xml).

These components together with the connection arrows correspond to a single workflow specification, i.e. a meta-matcher. The resulting specification is represented in a XML file, partially depicted in Figure 3.

Notice that the GOALS matchers' selection task and parameters specification is similar to many existing matching systems (OAEI'2008, 2008). The novelty is that GOALS allows the specification of the data flow between different components and all the possibilities that arise with this feature. This is possible because GOALS is not committed to any pre-defined system architecture (i.e. sequential, parallel or hybrid). In order to exemplify this feature, suppose that the alignment resulting from the previous workflow is poor. Furthermore, suppose that the main reason for that is the poor Matrix's quality resulting from MaxAggregator1. To solve this, one decides to improve the results by (i) filtering the Matrix using a high threshold value and (ii) applying the FOAM matcher seeded (i.e. to receive anchors) with the filtered Matrix. Figure 4 partially depicts the reconfigured meta-matcher.

Running this new meta-matcher, the resulting alignment (i.e. the Alignment.xml file) would be updated with a new alignment (hopefully improved). Remember that the meta-matcher might also have components concerned with evaluation, by comparing the resulting alignment with a reference alignment.

```
(...)
<ListOfComponents>
 <!-- Creating Jaro1 component -->
 <Component>
  <Name>Jaro1</Name>
  <Class>
  pt.ipp.isep.gecad.goals.components.matchers.stringbased.JaroMatcher
  </Class>
 </Component>
 <!-- Creating ThresholdFilter1 component -->
 <Component>
  <Name>ThresholdFilter1</Name>
  <Class>
pt.ipp.isep.gecad.goals.components.filtersAggregators.MatrixFilterByThres
hold
  </Class>
  <ListOfParameters> <!—Begin of ThresholdFilter1 Parameters -->
  <ParameterSet>
  <Parameter>
  <Name>SaveToFileNameOrURI</Name>
   <Class>pt.ipp.isep.gecad.goals.engine.parameters.URIParameter</Class>
   <Value>./Alignment.xml</Value>
   </Parameter>
   <Parameter>
   <Name>WantToSaveData</Name>
   <Class>pt.ipp.isep.gecad.goals.engine.parameters.BooleanParameter
   </Class>
   <Value>true</Value>
   </Parameter>
   <Parameter>
   <Name>ThresholdType</Name>
   <Class>pt.ipp.isep.gecad.goals.engine.parameters.StringParameter
   </Class>
   <Value>HARD</Value>
   </Parameter>
   <Parameter>
   <Name>ThresholdValue</Name>
   <Class>pt.ipp.isep.gecad.goals.engine.parameters.DoubleParameter
   </Class>
   <Value>0.95</Value>
   </Parameter>
   </ParameterSet>
  </ListOfParameters> <!—End of ThresholdFilter1 Parameters -->
 </Component>
(...)
</ListOfComponents>
<DataFlow>
 <!-- Sending Source Ontology to Jaro1 Component -->
 <Connection>
   <From>OntologyLoader1</From>
   <To>Jaro1</To>
   <DataEntity>ONTOLOGY</DataEntity>
   <Role>SOURCE_ONTOLOGY</Role>
   <Parameter/>
 </Connection>
 <!-- Sending Matrix from WeightAggregator1 to ThresholdFilter1 -->
 <Connection>
   <From>WeightAggregator1</From>
   <To>ThresholdFilter1</To>
   <DataEntity>MATRIX</DataEntity>
   <Role/>
   <Parameter/>
  </Connection>
  (...)
</DataFlow>
```

Figure 3: Fragments of an workflow specification

Finally, one might repeatedly reconfigure the meta-matcher specification easily and quickly by adding, changing and removing components in order to improve the achieved result without being committed to any pre-defined system architecture or constraints.

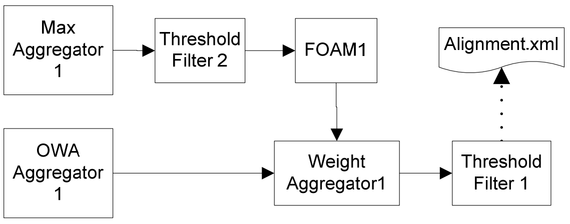

Figure 4: Partial improved meta-matcher example.

5 CONCLUSIONS

GOALS encourages the reusability of existing basic matchers and also the more complex ones. GOALS provides and supports prototyping and/or building complex ontology matching algorithms and facilitates testing and evaluation, promoting an iterative and incremental process of complex matchers development. Its plug-in-based architecture provides an incremental approach: the resulting matching system can be applied as a new operation component in a new matching system.

GOALS by itself, is not an ontology matching system, but a very flexible, adaptable and powerful tool for building ontology matching systems. In that sense, no comparative tests with existing ontology matching systems were neither done, nor are relevant. Additionally, results would be dependent of the workflow specification.

In order to use, test and explore all available features, GOALS is available for download at (Maio & Silva, 2009). To improve the user experience, we are working to provide GOALS with a GUI which it is expected to be available shortly.

GOALS is the first step of a larger effort concerning two complementary approaches: automaticity and complexity of the matchers. With respect to automaticity, we are planning to enrich GOALS with a module that will be able to automatically generate a workflow specification based on a full matching scenario characterization, according to several dimensions, such as domain of ontologies, time constraints and envisaged application. With respect to complexity, GOALS will provide the test-bed for carrying out our research efforts concerning the development of new algorithms that will address more accurate and complex mappings (e.g. “firstName + lastName” = “fullName”), especially devoted to data integration. Because it is very flexible and adaptable, GOALS is well suited for testing and evaluating complex approaches in a very immediate way.

ACKNOWLEDGEMENTS

This work is partially supported by the Portuguese MCT-FCT project COALESCE (PTDC/EIA/74417/2006). Thanks to Owen Gilson for his revisions of the paper.

REFERENCES

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, *284*(5), 34—43.

Doan, A., Madhavan, J., Domingos, P., & Halevy, A. (2004). *Ontology matching: A machine learning approach*. Handbook on Ontologies.

Ehrig, M., & Euzenat, J. (2005). Relaxed Precision and Recall for Ontology Matching. In *Proc. K-CAP Workshop on Integrating Ontologies* (p. 25—32). Banff (CA). Retrieved from http://ceur-ws.org/Vol-156/paper5.pdf.

Ehrig, M., & Sure, Y. (2004). Ontology mapping — an integrated approach. In *Proc. 1st European Semantic Web Symposium (ESWS)*, Lecture notes in computer science (Vol. 3053, p. 76—91). Hersounisous (GR). Retrieved from http://www.aifb.uni-karlsruhe.de/WBS/meh/publications/ehrig04ontology_ESWS04.pdf.

Ehrig, M., & Sure, Y. (2005). FOAM - Framework for Ontology Alignment and Mapping; Results of the Ontology Alignment Initiative. In B. Ashpole, M. Ehrig, J. Euzenat, & H. Stuckenschmidt (Eds.), *Proceedings of the Workshop on Integrating Ontologies* (Vol. 156, p. 72—76). CEUR-WS.org. Retrieved from http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-156/paper11.pdf.

Euzenat, J. (2004). An API for ontology alignment. In *Proc. 3rd International Semantic Web Conference (ISWC)*, Lecture notes in computer science (Vol. 3298, p. 698—712). Hiroshima (JP).

Euzenat, J. (2007). Semantic Precision and Recall for Ontology Alignment Evaluation. In *In Proc. of 20th International Joint Conference on Artificial Intelligence*. Hyderabad, India.

Euzenat, J., & Shvaiko, P. (2007). *Ontology Matching* (1st ed., Vols. 1-1, Vol. 1, p. 334). Heidelberg, Germany: Springer-Verlag.

Fensel, D. (2001). *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Heidelberg, Germany: Springer-Verlag.

Fürst, F., & Trichet, F. (2005). Axiom-based ontology matching. In *Proceedings of the 3rd international conference on Knowledge capture* (pp. 195-196). Banff, Alberta, Canada: ACM. doi: 10.1145/1088622.1088665.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software* (illustrated edition.). Addison-Wesley Professional.

Giunchiglia, F., Shvaiko, P., & Yatskevich, M. (2004). S-Match: an algorithm and an implementation of semantic matching. In *Proc. 1st European Semantic Web Symposium (ESWS)*, Lecture notes in computer science (Vol. 3053, p. 61—75). Hersounisous (GR).

GOALS. (n.d.). . Retrieved June 26, 2009, from http://www.dei.isep.ipp.pt/~pmaio/goals/.

Halevy, A., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., et al. (2005). Enterprise information integration: successes, challenges and controversies. In *Proc. 24th International Conference on Management of Data (SIGMOD)* (p. 778—787). Baltimore (MD US).

Ji, Q., Haase, P., & Qi, G. (2008). Combination of Similarity measures in Ontology Matching using the OWA Operator. In *Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Base Systems (IPMU'08)*.

Jian, N., Hu, W., Cheng, G., & Qu, Y. (2005). Falcon-AO: Aligning Ontologies with Falcon. In *Proc. K-CAP Workshop on Integrating Ontologies* (p. 87—93). Banff (CA).

Maio, P., & Silva, N. (2009). GOALS. *GOAlS - Gecad Ontology Alignment System*. Retrieved June 26, 2009, from http://www.dei.isep.ipp.pt/~pmaio/goals/.

Meilicke, C., & Stuckenschmidt, H. (2008). Incoherence as a basis for measuring the quality of ontology mappings. In *3rd International Workshop on Ontology Matching (OM-2008)* , Worksop Proceedings (Vol. 431). Karlsruhe, Germany: CEUR. Retrieved from http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-431/om2008_Tpaper1.pdf.

Miller, G. (1995). WordNet: A lexical database for English. *Communications of the ACM*, *38*(11), 39—41.

Mochol, M., Jentzsch, A., & Euzenat, J. (2006). Applying an analytic method for matching approach selection. In *Proc. 1st ISWC International Workshop on Ontology Matching (OM)* (p. 37—48). Athens (GA US).

OAEI'2008. (2008). Ontology Alignment Evaluation Initiative. *2008 Campaign*. Retrieved February 22, 2008, from http://oaei.ontologymatching.org/2008/.

Staab, S., & Stuckenschmidt, H. (2006). *Semantic Web and Peer-to-Peer: Decentralized Management and Exchange of Knowledge and Information* (1st ed.). Springer.