Grounding the Foundations of Ontology Mapping on the Neglected Interoperability Ambition

Hamid Haidarian Shahri [*], James Hendler [+], Donald Perlis [*]

[*] Department of Computer Science
University of Maryland, College Park, USA
{hamid, perlis}@cs.umd.edu

[+] Tetherless World Research Constellation
Department of Computer Science
Rensselaer Polytechnic Institute, Troy, USA
hendler@cs.rpi.edu

Abstract

The problem of ontology mapping has attracted considerable attention over the last few years, as the usage of ontologies is increasing. In this paper, we revisit the fundamental assumptions that drive the mapping process. Based on real-world use cases, we identify two distinct goals for mapping, which are: (i) ontology development and (ii) facilitating interoperability. Most of current research on ontology mapping has been focused on *ontology development* and is rooted in the seminal work of McGuinness and Noy in 2000. For example, the well studied problem of ontology merging is an ontology development task. Note that with the increase in the number of information systems that utilize ontologies, *facilitating interoperability* between these systems is becoming more critical. We compare interoperability to the information integration problem in databases. As a result of this comparison, class matching is emphasized, as opposed to the matching of other entities in an ontology. To the best of our knowledge, this is the first work that distinguishes facilitating interoperability, from ontology development and merging.

1. Introduction

The need for communicating between autonomous and distributed information systems is increasing with the wide usage of the Web. Nowadays, the issue of sharing data across resources and enterprises is no longer a desirable feature, but a necessity. Considerable amount of research on data integration and schema mapping over the past decades have lead to significant improvements in this area [Rah01]. The difficulty of finding correspondences between schemas originates from the fact that the conceptual models, used for data representation, do not capture the semantics of the data with enough precision. For example, it is very difficult to infer that *area* in one schema and *location* in another schema refer to the same real-world entity, as the meaning of attributes in the schema is not encoded explicitly. This problem is referred to as semantic heterogeneity.

Ontologies encode the specification of concepts more accurately, than schemas. The rich set of relationships defined between concepts in ontologies, help in mitigating the semantic heterogeneity problem. Since different ontologies exist and are being used by various organizations, it is necessary to find *correspondences* between these ontologies. The terms: ontology mapping, matching, alignment, integration, and merging, in the research literature, relate to this issue in various ways! In fact, unifying the interpretation of this diverse terminology is quite challenging. Usually, the goals of the task of finding correspondences in ontologies are not explicitly stated. Moreover, there is considerable vagueness on how the task should be performed, as the problem is often stated for some *specific* setting, or a *theoretical* one. Generally, there exists no consensus on what solution to use and under what circumstances, as evident by the variety of the terminology used. Nevertheless, previous research [Kal03, Noy04, McG00] is very valuable for developing the foundations of the ontology mapping problem.

In our opinion, this vagueness can only be resolved by observing the *use cases* of the problem. To the best of our knowledge, this is one of the very first attempts to put the previous research, on ontology mapping, in a unified context. This study revisits the ontology mapping problem in *various settings*, to furnish generality, and at the same time avoids *theoretical assumptions*, by adhering to real-world use cases.

The contributions of this paper are as follows: (i) Different *use cases* of ontology mapping are explored and clarified with real-world motivating examples. (ii) Two separate *goals* of the ontology mapping problem are identified, based on the use cases. They are interoperability and ontology development. (iii) *Interoperability* is highlighted as a major goal in ontology mapping, and the problem is revisited in this context, as opposed to the usual ontology development context. (iv) We provide an in-depth comparison to the information integration problem in databases. Based on this comparison, *class matching* is emphasized as the main ingredient in ontology mapping for facilitating interoperability. This is different from finding all matching entities, which is the focus of ontology development efforts.

2. Revisiting Ontology Mapping Goals with Motivating Examples

Currently, there are many ontologies that have been designed by different organizations and communities, and hence there is a need for a mapping between them. There are two quite distinct goals for ontology mapping. These goals are based on the types of use cases that we have identified, and will clarify in this section with motivating examples. Although, there are similarities between the use cases, one can differentiate the subtle requirements that arise from these examples, with careful observation. One possible goal of mapping is *ontology development*, when an ontology is being designed or engineered by an organization. The other possible goal of mapping is *interoperability*, when there are various parties, which are using different ontologies and the parties need a mechanism to be able to communicate and exchange information. *This distinction has seldom been addressed, in previous research on ontology mapping.* Clearly, interoperability is of considerable importance, as will be explained in this section.

2.1. Ontology Development

Since ontology is an abstraction for representing knowledge and all concepts that fit into the domain of human knowledge are connected together in some fashion, it is very hard to limit an ontology in terms of what it represents. This decision is usually made based on business needs, i.e. the ontology designer decides not to include some concepts, as they seem irrelevant to current organizational demands. Assume that an organization is currently using a *host ontology*, *H*.

Over time, as business models change and evolve, the ontology *H* also needs to be changed and often extended. Sometimes, the new business models, or some fragments of the changes that are required in the ontology, have already been captured by ontologies that are being used in other organizations. In this case, the required extensions to host ontology *H*, are existent in some other *guest ontology*, *G*. Now, the ontology designer of *H*, needs to: 1) find the correspondences between ontologies *H* and *G*, 2) decide on what concepts, relations, and instances of *G*, need to be added to *H*, based on the correspondences found in the previous step. This use case closely resembles the problem that has been analyzed in the context of merging two ontologies, in the literature [Noy00, McG00, Stu01].

Example 1: Consider two organizations offering various products and using two different ontologies O_1 and O_2 shown in Figure 1(a) and 1(b), respectively. O_1 is shown with ovals, while O_2 is shown with rectangles. The orange color represents the corresponding concepts between O_1 and O_2. In Figure 1, since class *Videos* in ontology O_1 is defined in a very similar context to class *Movies* in ontology O_2, it is conceivable to merge the two ontologies and produce a more comprehensive ontology. In essence, O_1 is being extended with O_2 and the merged ontology is a mix of ovals and rectangles, as shown in Figure 1(c).

Both organizations may need to make changes in their operation, in order to use the merged ontology. Furthermore, merging can be problematic, if the ontologies are defining the classes in different contexts, as merging would easily lead to irresolvable inconsistencies. Assume that *Electronic Equipment* in O_1 also has *Toys* as a subclass. Now, the merged ontology would have two *Toys* concepts, one of which is a subclass of *Sale Items* (Figure 1c) and the other is a subclass of *Electronic Equipment* (not shown). Even combining the two *Toys* concepts may not have the desired effect.

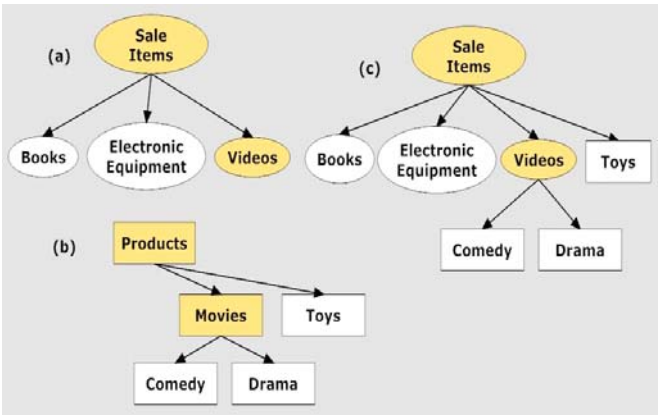

Fig. 1. Two ontologies O_1 and O_2 shown in (a) and (b). The merged result is shown in (c).

Important points arise from the study of this use case, which are as followed:

1. When ontologies are being merged, there is potential for inconsistencies and the ontology designer needs to make complex decisions in various steps of the process. Hence, the merging process can only be s*emi-automated* [Noy03] and no algorithmic solution exists. Moreover, the process must be *interactive*, to allow the designer to verify the changes.
2. The nature of the merging problem is such that the host ontology is usually not only being *extended*, but also needs to evolve, to accommodate the neighboring classes of the corresponding class in the guest ontology. For example, if the class *Movies* did not have a parent class, a simple extension would have sufficed, but now we must accommodate the *Products* class as well.
3. Finding correspondences between ontologies is necessary for ontology development, as illustrated by this use case.

2.2. Interoperability

Different enterprises use their proprietary and autonomous systems and are often not willing to change their business models and operations. On the other hand, they also need to exchange information. In many circumstances, users need to query these distributed and autonomous sources of

information, and retrieve data from all of them, as if all the information resides in a unified source.

Let us define this scenario more formally. Various autonomous ontologies, O_1, O_2, O_3, …, O_n, are designed and being used by n different organizations, also known as *parties*. Each ontology O_i is designed based on the business model that governs the operations of the organization that it belongs to. Hence, the ontology being used by each party can not be changed or extended. To facilitate interoperability, in this scenario, two steps are required: 1) correspondences between the ontologies of different parties have to be determined, 2) a skeleton S, must be developed, to represent these correspondences.

Example 2: Consider two universities in which faculties and departments within the faculties are structured differently, as shown in Figure 2(a) and 2(c). The ontologies O_1 and O_2 for the two universities are represented with ovals. There are six corresponding concepts in O_1 and O_2, namely: *University*, *Science*, *Maths*, *CS*, *Physics*, and *Chemistry*, shown with an orange color. Note that these six concepts appear in different places in O_1 and O_2. These six concepts are used in skeleton S, as shown in Figure 2(b), and represented with rectangles.

When creating the skeleton, the shape of the skeleton is determined by one of the original ontologies (parties). The shape of the skeleton is in fact the relationship between the concepts in the skeleton. In this example, the shape of the skeleton is the same as ontology O_1. Then, each concept in skeleton S is connected to its corresponding concepts in the original ontologies O_1 and O_2, with a subclass relationship. Figure 2 only shows such connections for the *University* concept, with red dotted arrows, and other such connections are not drawn for more readability. In Example 2, each organization's ontology (i.e. O_1 and O_2) remains intact, unlike Example 1.

Notice that the two use cases in Section 2.1 and 2.2 are very different. In Figure 2, consider that each of the departments of *CS* in O_1 and *Computer Science* in O_2 contain *instances* of courses being offered in those departments. In the interoperability use case, we would like to query for all courses related to computer science and retrieve results from both universities. In Figure 2, with the skeleton, we can query for *CS* courses in ontology O_1 and using query expansion, we move to the corresponding concept in the skeleton (which is *CS*), and then also retrieve the relevant courses from the *Computer Science* class in ontology O_2. Therefore, the query would return the results, as if all data resides in a unified source. Now assume that course *abc* is offered in the *CS* department in O_1, while a different course, named *efg*, is offered in the *Computer Science* department in O_2. Merging of these two departments (as done in Figure 1, for the ontology development goal) would imply that *abc* is being offered in both $Univeristy_1$ and $University_2$, which is not correct. Using the *owl:equivalentClass* construct (instead of creating a skeleton) for the purpose of interoperability is not acceptable for the same reason. Stating that $Class_1$ and $Class_2$ are equivalent classes using *owl:equivalentClass*, implies that every instance of $Class_1$ is also an instance of $Class_2$. This is a very strong statement, and not generally applicable for facilitating interoperability between systems.

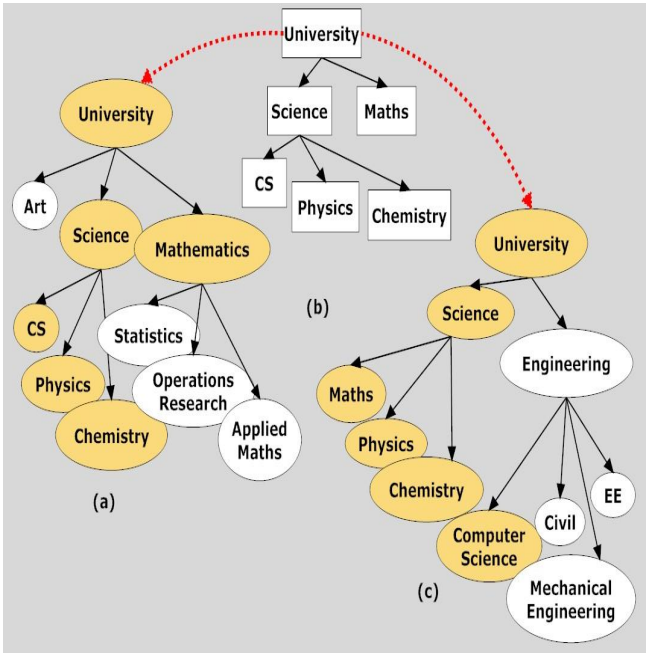

Fig. 2. O_1 and O_2 shown in (a) and (c) are the ontologies of two autonomous organizations. Skeleton S connecting the ontologies is shown in (b), in the middle.

The following observations can be made by careful examination of Example 2, and comparing it to Example 1:

1. *Isolation*: Creating the skeleton S, to represent ontology mappings is much more flexible than merging, and the autonomous ontologies O_i, are isolated from any further changes. This is very desirable, since autonomous organizations, which are using the ontologies, are usually not willing to change their business practices for the sole purpose of communicating with other organizations. Hence, interoperability must be facilitated by other means.
2. *Class Matching*: For interoperability, determining correspondences between two ontologies should be focused on the *classes* that match, in the original ontologies. Section 3 will elaborate on why class matching should be the focus. Nevertheless, matching of corresponding properties and instances can provide auxiliary information for the ultimate task of class matching.
3. *Tractability*: The creation of skeleton S, is more tractable and comprehensible, and leads to fewer inconsistencies than the merging process. Therefore, it can be streamlined and tackled algorithmically.

The above use cases and the discussion in this section demonstrate that *interoperability* (i.e. facilitating the

exchange of information between organizations) is a very important goal in ontology mapping. This goal is quite similar to what the database community is trying to achieve, in the context of information integration research and schema matching [Rah01, Len02]. However, current solutions to the ontology mapping problem have not addressed this goal, and are primarily focused on the merging of ontologies. The merging process is geared towards the *development of an ontology*, which is the other goal identified in this section. In the following section, we will concentrate on the interoperability goal, compare it to information integration, and describe its implications on the ontology mapping problem.

3. Class Matching: The Main Ingredient of Ontology Mapping for Interoperability

In the previous section, by observing the use cases, we illustrated that correspondence between ontologies need to be determined. In this section, we will show that for interoperability, the process of determining correspondences should be focused on *classes*. Note that the class matching objective does not imply that the matching of other entities is not *used for* class matching.

3.1. Information Integration

The problem of combining heterogeneous data sources under a single query interface is commonly known as "data integration" or "information integration" in the database community. Here, the idea is to provide a uniform query interface over a mediated schema. This query is then transformed into specialized queries over the original databases. This process can also be called view based query answering, because we can consider each of the data sources to be a view over the mediated schema. Formally such an approach is called Local As View (LAV), where "Local" refers to the local sources/databases. An alternate model of integration is one where the mediated schema is designed to be a view over the sources. This approach is called Global As View (GAV), where "Global" refers to the global (mediated) schema [Len02].

Figure 3 shows an example of the information integration problem in databases. Here, the goal is to generate a mapping between columns (*Town* and *City*) in different local schemas (*S* and *T*), by mapping them to some global schema. The local schemas usually reside in separate autonomous data sources (*DataSource1* and *DataSource2*). Figure 3 illustrates one example mapping between schema *S* and schema *T*. More details about the information integration process in databases can be found in [Len02]. This is a simple, but critical example, and will be used later in this section to demonstrate how ontology mapping should be performed to facilitate interoperability, and how ontology mapping for interoperability relates to schema mapping (i.e. information integration).

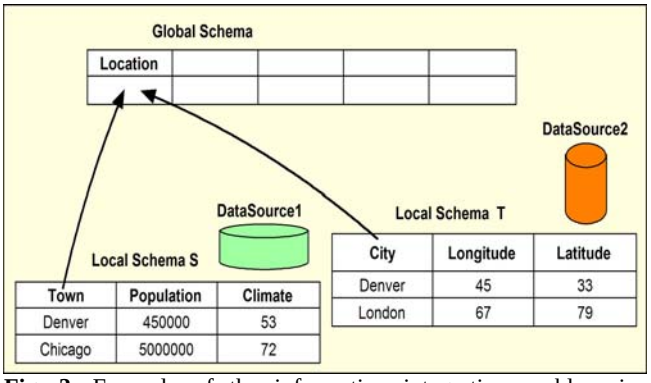

Fig. 3. Example of the information integration problem in databases. The goal is to generate a mapping between columns (*Town* and *City*) in different local schemas (*S* and *T*), by mapping them to some global schema. The local schemas usually reside in separate autonomous data sources (*DataSource1* and *DataSource2*).

3.2. Interoperability

Following the description of information integration above, it is important to point out that, the term "integration" is vague to some extent, since it may be interpreted as some type of "merging" of schemas. This is not what actually occurs in databases, as the schemas in each local data source are handled *autonomously* and need to be kept separately. The local data sources are often administered by different organizations and are not merged (integrated). In fact organizations are not willing to change their business models and everyday operations.

The ultimate goal of information integration is to provide *interoperability* between various systems, which is the exact same goal that we identified in section 2.2. The term "interoperability" is much clearer, for describing the motivations and objectives of the process. By analogy, in ontology mapping, there is no merging of ontologies involved, when we are trying to achieve interoperability between organizations, which use different ontologies (see Section 2.2).

3.3. Expression of Simple Facts in the RDF Model

Simple facts in the RDF model indicate a relationship between two things. Such a fact may be represented as an RDF triple in which the predicate (i.e. property) names the relationship, and the subject and object denote the two things. Figure 4(a) shows the predicate *hasAuthor*, which is the relationship between the subject and the object. The subject is an instance of **class** *Book*, while the object is an instance of **class** *Author*. The classes are depicted as ovals. For example, *The Art of Computer Programming* (which is a book) hasAuthor *Donald Knuth* (who is an author).

The use of extensible URI-based vocabularies in RDF facilitates the expression of facts about arbitrary subjects; i.e. assertions of named properties about specific named things. A URI can be constructed for any thing that can be named, so RDF facts can be about any such things. The

use of Universal Resources Identifiers (URIs) in the RDF model provides a very powerful mechanism for facilitating interoperability. Consider that the success and scalability of the current WWW infrastructure, is a vivid illustration of the tremendous potential of the idea of "using links" (which seems like a simple idea at first glance).

3.4. Expression of Simple Facts in the Relational Model

A familiar representation of a fact in the relational model in databases is a row in a table. The terms row and table are also known as tuple and relation, respectively. A table has a number of columns (also known as attributes). Figure 4(b) shows the *hasAuthor* table. The table has two **columns**, namely *Book* and *Author*. A row for example indicates that, *The Art of Computer Programming* (which is a book) hasAuthor *Donald Knuth* (who is an author).

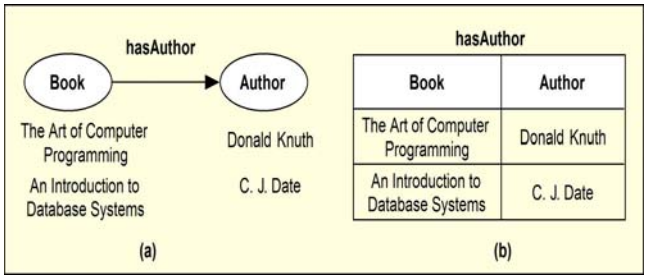

Fig. 4. Correspondence between the RDF and relational models. (a) The predicate *hasAuthor*, which is the relationship between the instances of class *Book* and the instances of class *Author*, in the RDF model. (b) The table *hasAuthor*, which has two columns, namely *Book* and *Author*, in the relational model.

3.5. The Analogy between the RDF and Relational Models

Comparing the explanation of the RDF and relational models above, and observing Figure 4, demonstrate that the **classes** *Book* and *Author* (in RDF) correspond to the **columns** *Book* and *Author* (in relational). The correspondence between classes and columns is an important one and will be used later in this section. The correspondence is also depicted in Figure 5. The other substantial correspondence is between instances of a class in RDF, with column values in relational.

In the above discussion, description of the relational model was constrained, such that a table only contained two columns. Now, we consider the general case where a table contains more than two columns. In Figure 6(b), the table in the relational model has three columns, namely *Book*, *Author* and *Publisher*. Then, the RDF model would also have three corresponding classes, as shown in Figure 6(a). The correspondence between classes and columns still holds. It is essential to realize that the name of the table in the relational model is arbitrary. We used TableName in Figure 6(b). Additionally, two predicates, namely *hasAuthor* and *hasPublisher*, are now used in the RDF model (Figure 6(a)). The name of the two predicates in RDF is arbitrary and could be anything.

Notice that from Figure 4, we do not infer a correspondence between the name of the table (*hasAuthor* in relational) and the name of the predicate (*hasAuthor* in RDF), as this correspondence has no real substance. The comparison in Figure 6 actually eliminates the role of the table name in the relational model.

Therefore, the substantial correspondence is between classes and instances in RDF, with columns and column values in relational, respectively. There is no correspondence for predicate (i.e. property) names in the relational model. This is a side effect of the fact that: In the RDF model, relations are encoded explicitly. These issues have rarely been mentioned in the ontology mapping literature in the Semantic Web community, and are essential for a better understanding of the ontology mapping process, as explained in Section 3.1.6.

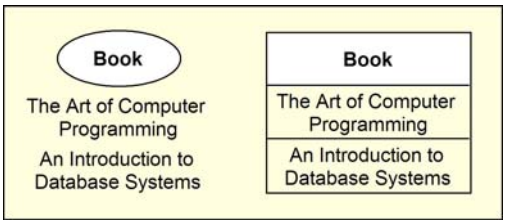

Fig. 5. The correspondence between classes (in the RDF model) and columns (in the relational model). There is also a correspondence between instances of a class (in RDF) and column values (in relational).

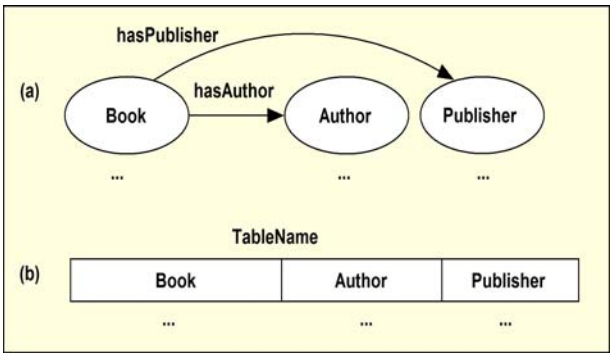

Fig. 6. A more general correspondence between the RDF and relational models. (a) The three classes in RDF are *Book*, *Author*, and *Publisher*. The name of the two predicates (*hasAuthor*, *hasPublisher*) in RDF is arbitrary and could be anything. (b) A table in the relational model, which has three columns, namely *Book*, *Author*, and *Publisher*. The name of the table (*TableName*) is arbitrary.

3.6. Class Matching: Why

Section 3.1 showed that in databases, the final output of the information integration process is a mapping between *columns* in different local schemas (see Figure 3). In Section 3.5, we illustrated that *columns* in the relational

model correspond to *classes* in the RDF model (see Figure 4 and 5). Therefore, to facilitate interoperability between ontologies, the *classes* in the ontologies need to be mapped to each other.

In RDF, the data is the instances of classes. The ultimate objective of interoperability is to query and correctly retrieve these data instances, across various ontologies. The data resides in classes in the ontologies. Notice that *as long as* a correct mapping between the classes in the ontologies exists, users can query and correctly retrieve the data instances, across various ontologies. For example, in Figure 2, users would like to retrieve course instances from both *CS* class in $Univeristy_1$ and *Computer Science* class in $Univeristy_2$. By analogy, in the relational model, the data is the values in the columns. A correct mapping between the columns in the schemas enables users to correctly retrieve column values across various schemas.

The detailed comparison in this section clarifies that the main ingredient of ontology mapping for facilitating interoperability is the matching of classes. The class matching objective directly facilitates interoperability. Identifying this objective is a helpful *guideline*, when performing the mapping, and has not been emphasized previously in the literature. This is one of the critical implications of focusing on the context of interoperability in ontology mapping. The class matching objective does not imply that other entities are not *used for* class matching. In fact, matching of other entities is usually *helpful for* the matching of classes.

4. Related Work

Many of the solutions to ontology mapping produce a merged ontology as the final output [McG00, Noy03, Stu01], and all this work is in the context of ontology development. Our work on ontology mapping for interoperability does not merge the ontologies. However, finding the matching classes is necessary in our approach, which is somewhat similar to the matching of various entities, in ontology merging. [Kal03] provides a good survey of various ontology mapping systems. Many systems look at finding lexical matches between ontologies and use dictionaries for this task. Chimaera is one of the early ontology merging tools, which considers structures such as subclass and superclass relations and slot attachments [McG00]. [Noy03] provides interactive support for merging ontologies and uses the graph structure of ontologies to provide suggestions. [Stu01] uses the set of shared instances or the set of shared documents annotated with concepts of two ontologies and generates a lattice to relate the concepts of the ontologies using formal concept analysis. [Dou03] is a system that merges ontologies and is proposed to be used by agents on the semantic web. [Mit00] proposes the use of rules across ontologies to create linkage between systems, and handle user queries. These rules are generated by a domain expert semi-automatically, and represented using a graph oriented model which is extended with set operators.

5. Conclusions

In this paper, we identify two goals for ontology mapping and distinguish them, for the first time, using real-world use cases. The goals are ontology development and facilitating interoperability. Much of current research in ontology mapping has been focused on ontology development and is rooted in the seminal work of [McG00, Noy00] in 2000. Clearly, *today*, providing interoperability between autonomous organizations is critical, considering the increase in the number enterprises that use ontologies in their information systems. Unfortunately, the ontology mapping problem has been mainly studied in the context ontology merging (i.e. ontology development).

We showed that the merging of ontologies is an ontology development task. Moreover, we compared the interoperability goal to the information integration problem in databases. As a result, for facilitating interoperability, class matching was emphasized, as opposed to the matching of other entities in an ontology.

References

[Dou03] Dou, D., McDermott, D., Qi, P., "Ontology Translation on the Semantic Web". *Proceedings of International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE'03)*. LNCS 2888, 2003.

[Kal03] Kalfoglou, Y., Schorlemmer, M., "Ontology Mapping: The State of the Art". *Knowledge Engineering Review*, Vol. 18(1), pp. 1-31, 2003.

[Len02] Lenzerini, M., "Data Integration: A Theoretical Perspective". *Proceedings of 21st ACM Symposium on Principles of Database Systems (PODS'02)*, June 3-5, Madison, Wisconsin, USA, pp. 233-246, 2002.

[McG00] McGuinness, D.L., Fikes, R., Rice, J., Wilder, S., "An Environment for Merging and Testing Large Ontologies". Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'00), Breckenridge, Colorado, USA, 2000.

[Mit00] Mitra, P., Kersten, M., and Wiederhold, G., "A Graph-Oriented Model for Articulation of Ontology Interdependencies". *Proc. 7th Int. Conference on Extending Database Technology (EDBT'00)*, Konstanz, Germany, 2000.

[Noy00] Noy, N.F, Musen, M., "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment". *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00)*, Austin, TX, USA, July 2000.

[Noy03] Noy, N.F, Musen, M., "The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping". *International Journal of Human-Computer Studies*, Vol. 59(6), pp. 983-1024, 2003.

[Noy04] Noy, N.F., "Semantic Integration: A Survey of Ontology-Based Approaches". *SIGMOD Record*, Vol. 33(4), pp. 65-70, 2004.

[Rah01] Rahm, E., Bernstein, P.A., "A survey of approaches to automatic schema matching". *VLDB Journal*, Vol. 10(4), pp. 334-350, 2001.

[Stu01] Stumme, G., Maedche, A., "FCA-merge: Bottomup merging of ontologies". *IJCAI 2001*, pp. 225-234, 2001.