Towards Distributed Legal Information Systems: A Discussion of Ontology Mismatches

Pepijn R.S. Visser

Department of Computer Science, University of Liverpool
P.O. Box 147, Liverpool, L69 7ZF, United Kingdom
pepijn@csc.liv.ac.uk, tel: (+44).151.794.3792

Abstract

The information required for solving legal problems cannot be assumed to be contained in one information system anymore. Solving legal problems nowadays requires the integration of information from multiple information systems. In this article we address the development of distributed legal information systems. We focus on the problems arising from differences in domain ontologies in the participating systems. Also, we classify ontology mismatches and study how hard it is to overcome each of the ontology mismatches.

1. The Legacy Problem

Prior to decision making legal practitioners often spend a considerable amount of their time gathering information. The information that is required might be held by different organisations residing at different locations. Deciding on eligibility for social security benefits, for instance, may involve organisations such as the Department of Social Security, the Inland Revenue, and the Vehicle Registration authority. Apart from being at different physical locations these organisations have their own culture regarding the information systems in use. In general these systems are developed by different people, on different platforms using different software tools and with different aims and views on their domain. The differences in software that result from this diversity make it particularly hard to integrate a set of existing information systems. This problem is known as the legacy problem. If we are to support legal practitioners in their task of gathering and integrating information then we have to come to terms with this problem. In this article we address the legacy problem in the context of the development of distributed legal information systems. We focus on the differences in assumptions concerning the conceptualisation of the domain knowledge by studying the ontology mismatches between different heterogenous systems.

The remainder of this article is structured as follows. In section 2 we present an example motivating the need for distributed legal information systems. In section 3 we analyse the mismatches that can occur between the ontologies of two different systems by presenting a classification of ontology mismatches. Then, in section 4 we briefly discuss the difficulty of dealing with each of the ontology mismatches. In section 5 we outline KRAFT, a distributed architecture that deals with heterogeneity in ontologies and show how it can be applied to the legal domain. Finally, in section 6 we draw conclusions.

2. A Possible Scenario for Distributed Legal Problem Solving

Legal problem solving using distributed information can be illustrated by discussing a problem-solving scenario regarding a request for income support. The scenario is an illustration of how different information systems can contribute to solving one particular legal problem, and should not be viewed as typical or desirable for future developments in the automation of law.

Imagine that a request for income support is sent to a Social Security Office. In the office the case is encoded to enable the automatic consultation of a distributed computer system. After the encoding is done the request is forwarded to a knowledge system which has the relevant social security regulations and is capable of determining a person's eligibility. In deciding on the request, the information about previous and current labour relations of the applicant is determined by querying a database of the Inland Revenue. Assume that the system concludes that the person is entitled to income support. In parallel with the legal knowledge system a fraud-prevention knowledge system is put on the case. This system determines whether the case at hand is a 'suspected fraud case' and should be investigated as such. Again, the database from the Inland Revenue is queried but this time to cross check the information supplied by the applicant with the information held by the Inland Revenue. Similarly, the central database of the Social Security Office is consulted to determine any possible other social security benefits, such as disability and sickness payments, or, unemployment benefits. Finally, the database of the Vehicle Registration Office is consulted to determine whether the applicant has registered an exceptionally expensive vehicle. Let us assume that the fraud-prevention system tags the case as a suspect case. The case is now forwarded to a case retrieval system which returns a set of similar cases that the Social Security Office has dealt with before. All information is passed on to a human clerk. After studying the case she decides to overrule the fraud-prevention system and grant the applicant the income support.

In this example we have used two knowledge systems, a case retrieval system and three databases, all of which contribute to the determination of eligibility for income support. Although this set of systems can be extended easily the example suffices to illustrate that different systems (which are or can be thought of as being located at different physical locations) may have to be consulted in deciding on one particular case. As stressed in the introduction we like to investigate the problems concerned with the development of such a distributed system, assuming that the various systems have been developed independently of each other. The focus will be on mismatches between ontologies of the legal domain (for an overview of legal-ontology issues we refer to Visser and Winkels, 1997). In the next section we present a classification of ontology mismatches that could occur between the systems described above. The classification is meant as an instrument to assess whether it is possible to integrate legacy systems (and thus, to determine that some systems have to be rebuilt).

3. An Analysis of Ontology Mismatches

Following Gruber in his original definition that an ontology is an explicit conceptualisation of a domain (Gruber, 1995) we distinguish two sub processes in the creation of an ontology; conceptualising a domain and explicating the conceptualisation (this distinction forms the basis for our classification of ontology mismatches, the idea being that ontology mismatches may be introduced during both sub processes) [1].

1) This section is based on a (non official) publication by Visser et al., 1997 which appeared in the working notes of the AAAI spring symposium on Ontological Engineering (Stanford University).

During the conceptualisation process decisions are made upon classes, instances, relations, functions and axioms that are distinguished in the domain. One could imagine the Vehicle Registration Office to use a system in which, for instance, owner, car make, registration number, number of cylinders, and car weight are distinguished. Usually, the conceptualisation process also involves ordering the classes in a hierarchical fashion, and assigning attributes to them. We assume the outcome of this process to be a conceptualisation. In particular, this conceptualisation consists of a set of ontology concept descriptions $\{C\_\beta,..,C\_n\}$ in which C_i is either a class description, an instance description, a relation description, a function description or an axiom description. In the conceptualisation process, we do not specify the form or the appearance of these descriptions as their specification is the topic of the explication process (although later on, we will for practical reasons assume that it is a natural language expression).

Explicating the ontology-concept descriptions requires an ontology language. We do not choose a specific ontology language but confine ourselves to a generic form of an ontology. An ontology is defined as a 5-tuple O = <CD,RD,FD,ID,AD> in which CD is a set of class definitions, RD is a set of relation definitions, FD is a set of function definitions, ID is a set of instance definitions, and AD is a set of axiom definitions.[2] In defining ontology mismatches we confine ourselves to mismatches between definitions of concepts (CD), definitions of relations (RD), and definitions of instances (ID).[3] Relations defined in RD are divided into relations that contribute to the taxonomic structure of the domain (viz. structuring relations), and those which do not (viz. non-structuring relations) (Guarino, 1995). We consider definitions in CD, RD, and ID to be 3-tuples Def = <T,D,C> in which T is the definiendum, D is the definiens (to avoid confusion with the definiens, we use the letter T – for term – to denote the definiendum), and C is the concept description that is to be defined (this description is the one distinguished during the conceptualisation process). For practical reasons, we here assume C to be expressed in natural language (cf. the DOCUMENTATION relation which is linked to definitions in ONTOLINGUA). T is an atomic formula in a formal language and D is a (compound) formula in a formal language. Distinguishing the concept description to be defined (C) facilitates the expression of the intention of the knowledge engineer in making the definition, thereby allowing us to distinguish between different definitions with the same term and definiens components but with different intentions (see also section 3). An example of a definition (in PROLOG format) is the concept description 'An employee is a natural person having a labour relationship' (C), which is explicated as employee(X) $\leftarrow$ natural_person(P) $\wedge$ labour_relationship(P, L), in which employee(X) is the definiendum (T), and natural_person(P) $\wedge$ labour_relationship(P, L) is the definiens (D).

Semantic heterogeneity has been studied extensively in the database field (e.g., Batini et al., 1986; March, 1990; Ceri and Widom, 1993; Kitikami et al., 1996). Ceri and Widom (1993), for instance, list four categories of semantic conflicts: (1) naming conflicts (different databases use different names to represent the same concepts), (2) domain conflicts (different databases use different values to represent the same concepts), (3) meta-data conflicts (same concepts are represented at the schema level in one database and at the instance level in another database), and (4) structural conflicts (different databases use different data organisation to represent the same concept).

2) These components are the components of an ontology defined in ONTOLINGUA, see Gruber (1992).

3) We do not address functions and axioms in our classification of ontology mismatches since their form is slightly different from the other definitions.

The classification presented below is not necessarily restricted to databases. Because we focus on the underlying ontologies the classification applies to databases as well as knowledge systems, and in fact, to every information system. The four conflict categories listed above can be shown to be contained in the classification of ontology mismatches below. We do not claim the classification to be complete nor disjoint. We deem the classification to be a useful instrument to determine what mismatches are present between ontologies, to determine which mismatches are hard to resolve, and, to define guidelines for the design of interoperable systems. As stated before, there are two basic types of ontology mismatches: (1) conceptualisation mismatches, and (2) explication mismatches.

(M1) Conceptualisation mismatch
The conceptualisation mismatches are mismatches between two (or more) conceptualisations of a domain. The conceptualisations differ in the ontological concepts distinguished or in the way these concepts are related.

(M1.1) Class mismatch
A class mismatch is a conceptualisation mismatch relating to the classes distinguished in the conceptualisation. In particular, this type of mismatch concerns classes and their subclasses.

(M1.1.1) Categorisation mismatch
A categorisation mismatch occurs when two conceptualisations distinguish the same class but divide this class into different subclasses. The conceptualisation of the fraud-prevention system, for instance, might split the category of cars into exclusive and normal, whereas the conceptualisation used for the Vehicle Registration Office might split the category of cars into sub categories lorries and motor cars.

(M1.1.2) Aggregation-level mismatch
An aggregation-level mismatch occurs if two conceptualisations both recognise the existence of a class, but define classes at different levels of abstraction. The conceptualisation used for the Vehicle Registration Office will probably distinguish between lorries and motor cars whereas the conceptualisation for the income support knowledge system might only distinguish cars.

(M1.2) Relation mismatch
A relation mismatch is a conceptualisation mismatch relating to the relations distinguished in the conceptualisation. Relation mismatches concern, for instance, the hierarchical relations between two classes or, to the assignment of attributes to classes (cf. Woods, 1985).

(M1.2.1) Structure mismatch
A structure mismatch occurs when two conceptualisations distinguish the same set of classes but differ in the way these classes are structured by means of ontology relations. The conceptualisations for the fraud-prevention system and the Social Security Office database, for instance, might both distinguish parent and child but the fraud-prevention system might structure these with the relation a-dependent and the database might relate them with has. Although these two relations have a substantial overlap in their meaning they are not equal (this incorporates the dependency mismatch of Batini et al., 1986).

(M1.2.2) Attribute-assignment mismatch
An attribute-assignment mismatch occurs when two conceptualisations differ in the way they assign an attribute (class) to other classes. The Vehicle Registration Office might, for instance, assign the attribute owner to the class vehicle whereas the Social Security Office might assign the attribute owned-vehicle to the class person. Other examples can be found in Visser (1995, p.98).

(M1.2.3) Attribute-type mismatch
An attribute-type mismatch occurs when two conceptualisations distinguish the same (attribute) class but differ in their assumed instantiations. The conceptualisation for the Inland Revenue database, for instance, could assume its attribute married-to to be instantiated with a National Insurance number, whereas the conceptualisation for the database of the Social Security Office could assume the same attribute to be instantiated with a string.

(M2) Explication mismatch
Explication mismatches are not defined on the conceptualisation of the domain but on the way the conceptualisation is specified. We assumed that such an explicit conceptualisation (viz. an ontology) consists of a set of definitions. Recall that each definition is a 3-tuple Def = <T,D,C> in which T is a term, D is a definiens, and C is the ontological concept (from the conceptualisation) that is explicated.

The three components of a definition allow us in principle to distinguish eight different binary relations (match or mismatch) between two definitions, and thus eight different types of mismatches. However, if terms, definiens, and concepts of the two ontologies are all different we assume there to be no mismatch (viz. there is no correspondence). Neither is there a mismatch if terms, definiens, and concepts of the two ontologies are all the same (viz. there is a complete match). This leaves six different types of mismatches. Thus, we assume an explication mismatch to occur when two ontologies have different definitions where their terms, their definiens, or their ontological concepts are identical.

(M2.1) CT mismatch (or Concept and Term mismatch)
A CT mismatch occurs when two ontologies use the same definiens D but differ in both the concept C they define and the term T linked to the definiens. The fraud-prevention ontology could, for instance, label an exclusive car as a high-value vehicle (as in: $\text{highvalue\_vehicle}(X) \leftarrow \text{exclusive}(X) \wedge \text{car}(X)$), whereas the Inland Revenue ontology could label this as a taxable car (as in: $\text{taxable\_car}(X) \leftarrow \text{exclusive}(X) \wedge \text{car}(X)$).

(M2.2) CD mismatch (or Concept and Definiens mismatch)
A CD mismatch occurs when two ontologies use the same term T but differ in the concept C they define and the definiens D used for the definition. Consider the term blind. For the Social Security Office ontology the term blind may be defined as $\text{blind}(X) \leftarrow \text{person}(X) \wedge \text{lacking\_sight}(X)$ whereas the fraud-prevention ontology could define blind as: $\text{blind}(X) \leftarrow \text{action}(X) \wedge \text{aimed\_at\_hiding\_the\_truth}(X)$. Although both ontologies use the same term T it is clear that the ontologies denote a distinct concept and use a different definiens. Note, that a CD mismatch implies that T is a homonym.

(M2.3) C mismatch (or Concept mismatch)
A C mismatch occurs when both ontologies have the same term T and definiens D but differ in the concept they define. For instance, the above mentioned ontologies could both have definitions:

$$\text{single\_blind}(X) \leftarrow \text{blind}(X) \wedge \text{single}(X)$$

while the fraud-prevention ontology defines a one-off attempt to hide the truth

whereas the Social Security Office ontology defines a blind person who is single. Note that, like the CD mismatch, a C mismatch implies that T is a homonym.

(M2.4) TD mismatch (or Term and Definiens mismatch)
A TD mismatch occurs when two ontologies define the same concept C but differ in the way they define it; both with respect to the term T and the definiens D. Compare, for instance, the two following definitions of a male person: male(X) $\leftarrow$ natural_person(X) $\wedge$ gender(X, man) and boy_or_man(X) $\leftarrow$ man(X) $\wedge$ boy(X). Although representing the same concept in the world, these ontologies differ both in their term T and their definiens D. Note, that the TD mismatch implies that the two terms are synonyms (possibly the two definiens contain synonyms as well).

(M2.5) T mismatch (or Term mismatch)
A T mismatch occurs when two ontologies define the same concept C using the same definiens D but refer to it with different terms. This mismatch occurs when ontologies use synonyms. Consider the use of the expression: lorry(X) $\leftarrow$ vehicle(X) $\wedge$ designed_for_heavy_goods(X) by one ontology and truck(X) $\leftarrow$ vehicle(X) $\wedge$ designed_for_heavy_goods(X) by another ontology, both defining the concept of a vehicle that is designed for carrying heavy goods. Note that, like the TD mismatch, the T mismatch implies that the two terms are synonyms.

(M2.6) D mismatch (or Definiens mismatch)
A D mismatch occurs when two ontologies define the same concept C and use the same term T to refer to the concept but use a different definiens. For instance, one ontology may define the concept of term sick with the expression: sick(X) $\leftarrow$ natural_person(X) $\wedge$ state_of_being_ill(X), whereas a second ontology may define the same term with the expression: sick(X) $\leftarrow$ ill(X).

The two main categories of ontology mismatches reflect two different perspectives of looking at ontology mismatches. It should be noted that each conceptualisation mismatch should also appear in the explication of that conceptualisation. Indeed, some conceptualisation mismatches are easily recognised as explication mismatches. Consider, for instance, the attribute-type mismatch. This mismatch type occurs in the explication as a D mismatch (or as a CD mismatch, depending on whether the type is specified in the description of the ontological concept C). In fact, all explication mismatches must occur in some form in the explication (ontology). However, not all explication mismatches necessarily occur in the conceptualisation. For instance, the actual terms (identifiers) to denote concepts are usually chosen in the explication process. This confirms the idea that certain ontological decisions are made only when the conceptualisation is explicated, and thus, that certain ontological mismatches occur only in the explication.

The reason why we adhere to both sets of ontology mismatches is twofold. First, it allows us to tell whether certain types of mismatches arise from the conceptualisation process, or from the explication process (which forms a basis to resolve them, see section 4). Second, some mismatches are better understood at a conceptual level (viz. in terms of classes and their hierarchical relations), whereas some mismatches are better understood in terms of ontology components (viz. in terms of terms and definiens). An example of the former mismatches is the categorisation mismatch which is clearly understood as a class with different subclasses, but less obviously understood when expressed as a CD or D mismatch. An example of the latter mismatches is when one term refers to two different concepts (viz. a C or CD mismatch) which is clearly an explication mismatch, but maybe not recognised as a conceptualisation mismatch (for instance, because they are only used in a certain context).

4. Coping with ontology mismatches

Ontology mismatches impede the integration of information from different information systems. However, some ontology mismatches are less difficult to cope with than others. In this section, we briefly discuss the difficulty of each of the ontology mismatches by considering possible mappings between the ontologies. The scope of this article does not allow us to elaborate on each of the mismatches in full. For a more details we refer to the prolonged discussion as presented in Visser (et al., 1997).

4.1. Conceptualisation mismatch

The categorisation mismatch may be difficult to resolve. Theoretically, the mismatch can be resolved by abstracting the knowledge of the two sets of subcategories to the abstraction level of the common class (see definition of categorisation mismatch). However, the common class could be a very abstract class (e.g., the root of a class hierarchy), which implies that the abstraction process, and thus the mapping function, is a complicated and 'knowledge-intensive process'. The mapping between ontologies that have an aggregation-level mismatch requires the knowledge first to be expressed at the same aggregation level. In general, this can be done either by abstracting the more detailed knowledge, or, by specialising the less detailed knowledge (or indeed by doing both). Providing mapping rules for structure mismatches in the general case is difficult. Having different relations between identical classes can be problematic but may not be in special cases. This depends on the case at hand. The attribute-assignment mismatch occurs when two conceptualisations differ in the way they assign an attribute class to other classes. Whether this type of mismatch is easily mapped again depends on the case at hand. The mapping rules for ontologies which have an attribute-type mismatch are rather straightforward if both systems store exactly the same information in different formats. However, it is likely that two systems do not store precisely the same information content. The mapping then will not be bi-directional.

4.2. Explication mismatch

An explication mismatch occurs when two ontologies have different definitions but their terms, their definiens, or their ontological concepts are the same. The CT mismatch, which occurs if two ontologies use the same definiens D to denote two different concepts C andC' and refer to them with different terms T and T', respectively, probably does not require a mapping since the terms should be considered (and are already) different. In case of the CD mismatch, which occurs if two ontologies use the same term T while using different concepts C and C' and different definiens D and D', it is clear that the expressions T should not be considered equal. To resolve such a mismatch the terms need to be renamed (possibly only one). If two systems use identical terms T and definiens D but differ in the concepts C, as is the case with a C mismatch, then a mapping is required because the terms should be kept distinct in a cooperative context. The mismatch can be resolved by renaming the expressions (as in the CD mismatch). A TD mismatch occurs if two systems represent the same concept C but define it with different terms T and T' and definiens D and D'. There are two competing arguments. On the one hand, it can be argued that the terms refer to the same concept C for which reason the terms T and T' are considered synonymous and can be mapped onto each other. On the other hand, it can be argued that because the definiens D and D' are different the concepts denoted C are, in fact, not the same. Hence, there should be no mapping of the terms. Which solution is more appropriate cannot be stated in the general case. This depends, among others, on the relative importance of the differences between D and D'. A T mismatch occurs if two ontologies have the same definiens D to denote a concept C but refer to it with different terms T and T'. In this case the terms have to be mapped onto each other (T and T' are synonyms). In general, the mapping functions will be bi-directional. The D mismatch occurs if two systems define the same concept C and use the same term T but differ in their definitions. In this case it can be argued that the concepts are in fact not the same (cf. TD mismatch). Hence, in a cooperative context, the terms should be kept distinct. A solution would be to rename the terms. Alternatively, it can be argued that, because both systems are intended to define the same concept C, the terms and definiens should be considered the same, which implies that the terms should remain as they are. A straightforward mapping function (i.e., one that links the same terms in different systems) would suffice.

We can now list the ontological mismatches of our classification and divide them into three categories: (a) manageable, (b) hard, and (c) unknown (the latter meaning that the difficulty depends on the case at hand). The list reflects a first impression of the author on the solubility of the mismatches. It is therefore tentative.

(a) manageable:	T mismatch, CT mismatch, attribute-type mismatch, CD mismatch, C mismatch
(b) hard:	categorisation mismatch, structure mismatch
(c) unknown:	attribute-assignment mismatch, aggregation-level mismatch, TD mismatch, D mismatch

Not all mismatches are equally likely to occur. For instance, a CT mismatch is not likely to occur, but a categorisation mismatch is very likely to occur. Also, the mismatches differ in the ease with which they can be detected. For instance, a C mismatch is probably difficult to detect, whereas T and CT mismatches are easy to detect.

5. A Distributed Architecture

Obviously, the legacy problem can in theory be dealt with by redesigning all participating information systems. In theory, all systems could be rebuilt with one unifying ontology, but this is undesirable as well as infeasible. Ontology mismatches simply would not occur. However, it would be preferable to develop distributed architectures which integrate the (legacy) systems as they are (e.g., Wiederhold, 1994). In such systems expressions based on one ontology have to be mapped onto expressions based on other ontologies. In this section we outline the KRAFT architecture and illustrate how it could be applied to the scenario in section 2. KRAFT (Knowledge Reuse and Fusion / Transformation) is a collaborative project of BT and the universities of Aberdeen, Cardiff, and Liverpool focussing on architectures that integrate heterogeneous information systems. We illustrate where in the architecture ontology mappings are performed.

The KRAFT architecture is based on the idea that legacy systems should maintain their individual ontologies. Communication is performed by providing mapping rules from the individual ontologies to a shared – KRAFT internal – ontology. A KRAFT architecture has four major types of components (see figure 1). The first is the resource (white boxes in the figure) which, roughly speaking, is any system that is to be integrated in the architecture (e.g., knowledge-based systems, databases). In the scenario described in section 2 we identified five different resources: the Fraud-prevention Knowledge System, the Vehicle Registration Database, the Social Security Office Database, the Income Support Knowledge System, the Inland Revenue Database, and the Case Retrieval Information System. For convenience, we

Figure 1: A Distributed Architecture for Legal Problem Solving

here also assume the User Agent to be a resource. The second major type of component of a KRAFT architecture is the wrapper (denoted with a W in the figure). This component is responsible for translations between on the one hand the resource protocol, language and ontology, and on the other hand the KRAFT internal protocol, language and ontology (the large gray area in the figure denotes the boundaries of the shared KRAFT internal protocol, language and ontology). The wrapper thus is responsible for the ontology mappings that are described in section 4 (as well as for the protocol and language mappings). The third major type of component of the KRAFT network is the facilitator (denoted with an F in the figure). This component is a network router which has awareness of the available resources and their problem-solving capabilities.KRAFT components that want to contact other components consult the facilitator for the available resources and their network addresses. This kind of communication is depicted with dotted lines in the figure (as opposed to the solid lines which denote exchange of data and data requests). The fourth major type of component is the mediator (the light gray box within the gray area). This component is responsible for splitting information requests into sub requests, and for collecting and the integrating the results. For more information on the KRAFT architecture we refer to Gray (et al., 1997) or to our web site at http://www.csc.liv.ac.uk/ pepijn/kraft.html.

The architecture outlined here can be used to model the scenario of section 2. The request to determine the eligibility of the applicant is formulated by the user agent. Then, the request is sent to his wrapper which translates the request into the internal (protocol, language and) ontology of the network. The wrapper contacts the facilitator to determine which mediator or resource can deal with the request. The facilitator returns the address of the Income Support (InSu) Request Mediator and the wrapper sends the request to this mediator. Here, the request is split in two sub requests, one request to determine eligibility and one request to determine whether the case is a suspect fraud case. After consulting the facilitator for the appropriate capabilities and addresses, the mediator sends these requests to the Income Support KBS and the Fraud-prevention KBS, respectively. Both these systems require information from databases and queries are sent (after consulting the facilitator) accordingly. The result of both KBS systems is then sent back to the mediator which determines, on the basis of the results, to contact the Case Retrieval IS. After the latter system returns a set of relevant cases the mediator sends all information back to the user agent.

6. Conclusion

We illustrated that each resource maintains its own ontology and is wrapped to the shared ontology of the KRAFT network. The wrapper converts expressions that are based on the resource ontologies into expressions that are based on the shared ontology (and vice versa). Thus, the architecture is, in principle, suited to the integration of legacy systems. However, as discussed in section 3, it is not always possible to integrate two arbitrary ontologies. The ontologies should not be too different with respect to the information that needs to be exchanged. Otherwise stated, there is a trade-off between heterogeneity and interoperability (Visser et al., 1997). It is for this reason that we need to know which ontology mismatches can be resolved and which ontology mismatches cannot be resolved. Provisionally, we can conclude that systems can be integrated in a KRAFT network relatively straightforward if the only mismatches between the ontology of the system and the KRAFT-internal ontology are T mismatches, CT mismatches, attribute-type mismatches, CD mismatches or C mismatches. Whether systems with other mismatches can be integrated cannot be stated in general. This has to be determined on an individual basis. The integration of legacy systems remains a difficult problem and despite the enormous amount of research that is done in this area (e.g., Arens, 1996; Mena et al., 1996) there are still several fundamental issues that need to be resolved.

Acknowledgements

The author expresses his gratitude to his colleagues of the KRAFT project at BT (British Telecommunications Plc) and the universities of Aberdeen, Cardiff, and Liverpool – all of whom made contributions to the ideas presented in this paper. In particular, the author likes to thank Trevor Bench-Capon, Dean Jones, Alun Preece, and Mike Shave. The KRAFT project is funded by the Engineering and Physical Science Research Council (EPSRC) and BT.

References

Arens, Y., C.A. Knoblock and W.M. Shen (1996). Query Reformulation for Dynamic Information Integration, Journal of Intelligent Information Systems, No.6, pp.99-130.

Batini, C., M. Lenzerine, and S.B. Navathe (1986). Comparison of Methodologies for Database Schema Integration,ACM Computing Surveys, Vol. 18, No. 4, pp.323-364.

Ceri, S., and J. Widom (1993). Managing Semantic Heterogeneity with Production Rules and Persistent Queues, Proceedings of the 19th VLDB Conference, Dublin, Ireland, pp.108-119.

Gray, P.M.D., A. Preece, N.J. Fiddian, W.A. Gray, T.J.M. Bench-Capon, M.J.R. Shave, N. Azarmi, M.Wiegand, M. Ashwell, M. Beer, Z. Cui, B. Diaz, S.M. Embury, K. Hui, A.C. Jones, D.M. Jones, G.J.L. Kemp, E.W. Lawson, K. Lunn, P. Marti, J. Shao, and P.R.S. Visser (1997). KRAFT: Knowledge Fusion from Distributed Databases and Knowledge Bases, Proceedings of the Conference on Database and Expert System Applications (DEXA '97), Toulouse, France.

Gruber, T.R. (1992). Ontolingua: A Mechanism to Support Portable Ontologies, Version 3.0, Knowledge Systems Laboratory, Stanford University, Stanford, California, United States.

Gruber, T.R. (1995). Toward principles for the Design of Ontologies Used for Knowledge Sharing, Int. Journal of Human-Computer Studies, Vol.43, pp.907-928.

Guarino, N. (1995). Formal Ontology, Conceptual Analysis and Knowledge Representation, International Journal of Human-Computer Studies, special issue on The Role of Formal Ontology in Information Technology, N. Guarino and R. Poli (eds.), Vol 43, No. 5/6.

Kitakami, H., Y. Mori, and M. Arikawa (1996). An Intelligent System for Integrating Autonomous Nomenclature Databases in Semantic Heterogeneity, Database and Expert System Applications, DEXA'96, Zürich, Switzerland, pp.187-196.

March, S.T. (1990). Special Issue on Heterogeneous Databases, ACM Computing Surveys, ACM Press, Vol. 22, No. 3.

Mena, E., V. Kashyap, A. Sheth and A. Illarramendi (1996). OBSERVER: An approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS '96), Brussels, Belgium.

Visser, P.R.S. (1995). Knowledge Specification for Multiple Legal Tasks; A Case Study of the Interaction Problem in the Legal Domain, Computer / Law Series, No. 17, Kluwer Law International, The Hague, The Netherlands.

Visser, P.R.S., D.M. Jones, T.J.M. Bench-Capon and M.J.R. Shave (1997). An Analysis of Ontology Mismatches; Heterogeneity versus Interoperability, Working notes of the AAAI 1997 Spring Symposium on Ontological Engineering, Stanford University, California, USA, pp.164-172.

Visser, P.R.S. and R.G.F. Winkels (1997). Proceedings of the First International Workshop on Legal Ontologies (LEGONT'97), University of Melbourne, Law School, Melbourne, Victoria, Australia.

Wiederhold, G. (1994). Interoperation, Mediation, and Ontologies, Proceedings International Symposium on Fifth Generation Computer Systems (FGCS94), Workshop on Heterogeneous Cooperative Knowledge Bases, Vol. W3, pp .33-48, ICOT, Tokyo, Japan.

Woods, W.A. (1985). What's in a Link: Foundations for Semantic Networks, Readings in Knowledge Representation, R.J. Brachman, and H.J. Levesque (eds.), pp.217-241, Morgan Kaufmann, San Mateo, California, United States.