The Role of Ontology in Integrating Semantically Heterogeneous Databases
DB-Fusion 2002

Chris Partridge[1 & 2]

[1] BORO Program, 83 Sixth Avenue, Queens Park Estate, London, W10 4HF, England
partridge@BOROProgram.org
http://www.BOROProgram.org/
[2] National Research Council, Institute of Systems Theory and Biomedical Engineering (LADSEB-CNR), Group of Conceptual Modeling and Knowledge Engineering, Corso Stati Uniti, 4, I-35127 Padova, Italy
partridge@ladseb.pd.cnr.it
http://www.ladseb.pd.cnr.it/infor/ontology/BusinessObjectsOntology.html

Abstract. More and more enterprises are currently undertaking projects to integrate their applications. They are finding that one of the more difficult tasks facing them is determining how the data from one application matches semantically with the data from the other applications. Currently there are few methodologies for undertaking this task – most commercial projects just rely on experience and intuition. Taking semantically heterogeneous databases as the prototypical situation, this paper describes how ontology (in the traditional metaphysical sense) can contribute to delivering a more efficient and effective process of matching by providing a framework for the analysis, and so the basis for a methodology. It delivers not only a better process for matching, but the process also gives a better result. This paper describes a couple of examples of this: how the analysis encourages a kind of generalisation that reduces complexity and how ontological relativity can be used to enhance this. Finally, it suggests that the benefits are not just restricted to individual integration projects: that the process processes models which can be used as to construct a universal reference ontology – for general use in a variety of types of projects.

1 Introduction

Not only are there an increasing number of enterprise integration projects, but the need for wider and closer integration keeps on growing – driven by technologies such as the Internet and business goals such as straight through processing. Enterprises undertaking these projects are finding that one of the more difficult tasks facing them is working out how the data from the various applications matches up semantically.

Integration projects come in a variety of forms: for example, building a data warehouse or a middleware messaging system involves a significant amount of integration. However underlying this variety of forms is a common semantic task –

what can be called the 'matching of semantically heterogeneous data' or more simply 'semantic matching'. There is a reasonably clear recognition that the analysis stage of this task needs to focus on identifying the entities that the data describes – the 'real-world semantics'[1]. In theory and practice, this identification currently relies mostly on experience and intuition.

This is a situation familiar to metaphysicians. In ordinary life most people rely on experience and intuition to build up their picture of the 'real world'. Metaphysicians suggest that this ordinary everyday picture is based upon, among other things, implicit ontological assumptions. And that we tend to make avoidable errors about what entities we are talking about when we do not try and understand what these assumptions are. What adds poignancy to the metaphysicians' claim is that most people are unaware of the issue and seem positively uninterested in becoming aware – perhaps because they see no benefits in doing so.

1.1 This paper

This paper suggests that in the particular case of matching of semantically heterogeneous data, the (metaphysicians') ontology is helpful: not only in clarifying applications' ontological assumptions - but also in providing a common framework across applications for analysing what entities their data describes. This acts as both an explanatory framework and a foundation for a methodology for the analysis.

The paper makes its case in five sections which:

- Clarify the context for semantic matching task that ontology is intended to help – outlining what its prime source - semantic heterogeneity - is, how this arises out of a combination of design autonomy and diversity and how it is resolved.
- Clarify the meaning of the terms used to describe the proposed approach: ontology, semantics, ontological model, semantic divergence and ontological paradigm.
- Use these terms to characterise the ontological process – and illustrates it with examples of some of how the analysis approaches some of the main types of semantic divergence.
- Describes, using examples, two key ways in which this process can lead to better results: how the analysis encourages a kind of generalisation that reduces complexity (a serious problem in large integration projects) and how ontological relativity can be used to enhance this.
- Finally, describes how the potential benefits are not just restricted to individual integration projects: that the models built by integration projects

[1] "All these schema integration techniques require either explicitly or implicitly that (the relationship) between the real-world semantics of the classes to be integrated is known. This is a reasonable assumption in tightly-coupled approaches, but … in a federation of databases from multiple modelling contexts this may be surprisingly difficult." Vermeer, M. W. W. and Apers, P. M. G. *On the Applicability of Schema Integration Techniques to Database Interoperation*. ER 1996: 179-194.

can be used as to construct a universal reference ontology – which can then be used in a variety of types of projects.

2 Context

It is important to be clear about what the semantic matching task, that the ontological analysis is intended to help, is. This section starts by explaining why database integration has been chosen to illustrate the task and then outlines the current perception of the main focus of this task: semantic heterogeneity. It outlines where it comes from, what it is and the basis for resolving it.

2.1 The focus on database integration

As noted earlier, although application integration comes in a variety of forms, these share a common task – the 'matching of semantically heterogeneous data'. Our interest here is not in the variety of forms, but on how ontology helps in this common task. So it makes sense to focus on a single form that will most clearly illustrate this. Database integration is a reasonably common and straightforward form of integration, one in which the task clearly manifests itself – so this has been selected as the prototypical situation (the other forms can then be regarded as variations of this).

Simplifying slightly, the database integration design process can be regarded as[2]:

- taking the multiple databases – schemas and data as input, and
- producing as output a single unified database – schemas and data – and a mapping from the individual databases to the unified database.

The task of 'matching of semantically heterogeneous data' takes place during the analysis done in the initial stages. This determines how the elements of the multiple databases are matched with each other.

2.2 Prime source of semantic heterogeneity

What usually makes the task of matching onerous is a high level of heterogeneity between the databases. A prime reason of this heterogeneity is quite easy to identify, it

[2] Basically the same point is made in the Introduction of Parent, C. and S. Spaccapietra (2000). *Database Integration: The Key to Data Interoperability*. (Chapter 10 of *Advances in ObjectOriented Data Modeling*. M. Papazoglou and S. Spaccapietra. Cambridge, Mass., MIT Press.) - "The kernel of design issues … is the database integration problem. Simply stated, database integration is the process which: takes as input a set of databases (schema and population), and produces as output a single unified description of the input schemas (the integrated schema) and the associated mapping information supporting integrated access to existing data through the integrated schema."

is what Sheth and Larson call on p. 187 of [1] *design autonomy*. This is "the ability of a component DBS to choose its own design with respect to any matter". As they note, this includes "The conceptualization or semantic interpretation of the data (which greatly contributes to the problem of semantic heterogeneity)". In fact, they say: "Heterogeneity [in general] … is primarily caused by design autonomy among component DBSs."

Of course, autonomy by itself does not lead to heterogeneity. There is in principle no reason why two autonomous designers should not end up with the same design. However, in practice, autonomy allows what I shall call *design diversity* to manifest itself. This diversity is partly the result of the different requirements of the applications. But it also, partly, the result of the large amount of judgment exercised by the designers. This is reflected in the fact that different designers will (as a result of different judgements, different trade-offs) come up with different database designs for two similar applications. It can be quite surprising how different the designs can be[3].

2.3 Nature of semantic heterogeneity

It is important to clarify what semantic heterogeneity is. Sheth and Larson on p. 187 of [1] provide a description that is a useful starting point. They suggest that heterogeneity occurs "… when there is a disagreement about the meaning, interpretation or intended use of the same or related data [in different databases]." But they noted that "… this problem is poorly understood, and there is not even an agreement regarding a clear definition of the problem."

There are two interpretations of 'disagreement' in this description that illuminate different aspects of the integration task: disagreement between communities and disagreement in form.

Disagreement between communities. *Disagreement between communities* occurs when two or more communities are not agreed on the meaning of a piece of data in a database. There is no a priori reason for heterogeneity of this type to exist when integrating applications. Where a single, small community is solely responsible for the design of a number of applications, it will usually have naturally developed an understanding of how they fit together. When this community is made responsible for their integration, even if the designs are very different, there is no semantic heterogeneity in the community sense.

However, in practice, there usually is heterogeneity in this sense. This is understandable, as the applications often belong to different communities who have no real knowledge of the details of each other's applications. In these cases, there is typically no basis for a common agreement about what the data in the various systems mean – though this is more of a *lack* of common agreement than an actual

[3] For example, the various chapters of Papazoglou, M., S. Spaccapietra, et al. (2000) *Advances in object-oriented data modeling* show markedly different designs for a standard car example. As its Chapter 10 (Parent, C. and S. Spaccapietra (2000) *Database Integration: The Key to Data Interoperability*) notes there are a surprisingly wide variety of designs.

disagreement. This is problem that dogs most large integration projects. And a significant part of the work of integration is developing this agreement.

The usual way of developing agreement is for the communities to agree first on what entities their data describes – and then investigate how the data in the other databases describes these entities[4]. Ontology can help by providing a framework for analysing and, importantly, explaining, what these entities are.

Disagreement in form. *Disagreement in form* is where the "same or related" piece of data in different databases has a semantically different form. Sheth and Larson on p. 187 of [1] offer as an example two attributes with the same name - MEAL-COST. The first is related to the entity RESTAURANT in database DB1 and describes the average cost of a meal per person in a restaurant without service charge and tax. The second is related to the entity BOARDING in database DB2 and describes the average cost of a meal per person including service charge and tax. Despite their surface similarities these are semantically heterogeneous – where "the heterogeneity is due to differences in the definition (i.e., in the meaning)". The two attributes do not refer to exactly the same entity in the real world, but they are closely related: BOARDING's notion of MEAL-COST is a component of RESTAURANT's notion of MEAL-COST[5]. This kind of close miss is characteristic of semantic heterogeneity. Ontology's role is to help unbundle the objects and making clear the relation between them. From now on the focus is on semantic heterogeneity in this sense.

2.4 Basis for agreement

The last example also provides an illustration of what usually makes integration hard work. The databases in the example do not, by themselves, give us enough information. They do not tell us enough about the two MEAL-COST attributes for us to determine whether they describe the same or different things. And as databases tend to be semantically heterogeneous, we cannot even presume that having the same name is a good indication that they are describing the same kind of thing.

It is generally accepted that databases do not by themselves contain enough information for semantic matching[6]. And this connects the two senses of

---

[4] This point has been made through most of the history of database integration – for example, from Bill Kent's (1978) *Data and reality* through to Sheth's (1999) *Changing Focus on Interoperability in Information Systems: From system, syntax, structure to semantics*.

[5] It is worth noting that this, Sheth and Larson's and, presumably, the database designers' analysis assumes that the meaning of the individual attributes in not restricted to the associated entity – for example, that BOARDING's notion of MEAL-COST is not restricted to hotel meals. This assumption of a more general meaning is the beginnings of generalisation – a topic discussed later in the paper.

[6] See, for example, (Sheth and Larson 1990)'s comment that: "Typically, DBMS schemas do not provide enough semantics to interpret data consistently." As they also note: "Heterogeneity due to differences in data models also contributes to the difficulty in identification and resolution of semantic heterogeneity. It is also difficult to decouple the

‘disagreement’. The process of getting hold of the information needed to determine how the forms are related involves securing agreement from the parties involved.

Requirement for translation. Database integration (and this MEAL-COST example) can be regarded as a special case of a general requirement for translation. The need for securing correct agreement about what the data (language) describes is clearly visible in the extreme situation where people try to develop a means of communicating when there is no known translation between their languages. The need for sufficient information to be able to make a translation is one of the key points in Quine’s [2].

This point is well illustrated by the (possibly mythical) story of the meaning in English of the aboriginal word kangaroo. Apparently this meant “I don't understand”, and when the British explorers asked, presumably loudly and slowly in English, pointing to what we now know as a kangaroo, the aborigines replied “Kangaroo.”

The situation is not so extreme for database integrators as they can discuss what the databases describe in ordinary everyday language with the applications’ communities.

As [3] note in their Introduction “…existing database schemas provide basic knowledge about the semantics of data, which may be easily enhanced … through interviews of current users and data administrators.”

---

heterogeneity due to differences in DBMSs from those resulting from semantic heterogeneity.”

Limitations of expertise. Neither is the situation so straightforward. What experience of integration reveals is the extent to which experts have know-how but not what I shall call, know-what. They know how to do their job, but are less clear on what the entities involved actually are – the know-what[7]. This is a common observation in metaphysics[8]. And the point is made even more strongly in relation to the institutional and social facts that are the subjects of most enterprise data[9]. It is a less common observation in application design where the responsibility for knowing-what and checking this is passed over to the user-experts.

Real world semantics. So what one needs to know (secure agreement about) for integration, and many authors note this, is what the database representations mean, what entities they describe – their real world semantics. As [4] notes "…schema integration techniques require either explicitly or implicitly that (the relationship) between the real-world semantics of the classes to be integrated is known."[10]

Once we know what RESTAURANT's and BOARDING's MEAL-COSTs real-world semantics are – in other words, what the database representations describe, we are in a good position to start integrating them. And it is important to keep reminding ourselves that what is being identified here is not in the database but the entities that the database is describing.

An ontological framework for analysis. What ontology provides is a framework within which the "interviews with users and data administrators" can be focused on the entities the database describes and the information this provides analysed and organised. This framework helps both to explain what is happening when one analyses the 'real-world semantics' and to suggest a systematic process for undertaking it. This paper attempts to illustrate both of these.

3 Clarifying the terms

Before we look at the details of ontological analysis, we need to clarify the use here of some basic terms: firstly, ontology and semantics.

---

[7] There are examples of this 'lack of know-what' in. and M. Stefanova's *A Synthesis of State of the Art Enterprise Ontologies: Lessons Learned* (2001) and Partridge, C's *LADSEB-CNR - Technical report 07/02 - STPO - Synthesis of a TOVE Persons Ontology* (forthcoming) and *LADSEB-CNR - Technical report 04/02 - What is Pump Facility PF101?* (forthcoming).

[8] For example, Strawson makes this point in the Introduction to his *Analysis and metaphysics* (1992).

[9] See, for example, Searle's *The construction of social reality* (1995) and Gilbert's *On social facts* (1992).

[10] As it also notes: "One of the central problems … is that the definition of relationships between local and imported data is far from trivial in a situation where information on the meaning of a remote schema is limited. … [I]n a federation of databases from multiple modelling contexts this may be surprisingly difficult."

3.1 Ontology

For the purposes of database integration, the traditional philosophical (metaphysical) notion of ontology is useful – where this is "the set of things whose existence is acknowledged by a particular theory or system of thought." [11]

This view was famously summarised by Quine, who claimed that the question ontology asks can be stated in three words 'What is there?' – and the answer in one 'everything'. Not only that, but tongue in cheek, he also said "everyone will accept this answer as true" though he admitted that there was some more work to be done as "there remains room for disagreement over cases."[12]

Quine's description of ontology captures a common position in many disciplines, where it is unthinkingly assumed that the answer to the question "What is there?" will be a set of things. From the perspective of database integration, each database can be regarded as a 'theory' that acknowledges the existence of a set of objects – its ontology.

Some care needs to be taken to distinguish this traditional metaphysical use of the word ontology from one that has recently developed in Computer Science. Here an ontology is regarded as a "specification of a conceptualisation" [5] and has been applied to a wide range of things, including dictionaries. This sense of the word does not give us a fine-grained enough tool for our needs: it regards a database as simply an ontology – and so it cannot make sense of talking about the ontology underlying it, let alone underlying a set of databases.

A similar point can be made about the conceptual schema described in ANSI/X3/SPARC [6]. This is a representation of the conceptual perspective, and reflects how we conceive of the world – which is, in ways important for database integration, not the same as what our conceptualisation assumes exists in the world.

3.2 Semantics

Along with the traditional philosophical sense of ontology there is a related notion of semantics – where this is the relationship between words (data) and the world – the things the words (data) describe[13]. This needs to be distinguished from the different, but related, sense of the word in linguistics where it means the study of meaning[14].

These notions of ontology and semantics can then be used to describe two other useful notions – that of an ontological model and semantic divergence.

---

[11] E. J. Lowe in the Oxford Companion to Philosophy.

[12] In W.V. Quine's *On what there is* (1948), Review of Metaphysics, Vol. II, No. 5, reprinted in *From a logical point of view* (1961).

[13] Or as Nelson Goodman put it in the Introduction to Quine's lectures published as *Roots of Reference* – "… an important relation of words to objects – or better – of words to other objects, some of which are not words – or even better, of objects some of which are words to objects some of which are not words."

[14] "Semantics – the study of meaning" from the Concise Oxford Dictionary of Linguistics, © Oxford University Press 1997.

3.3 Ontological model

An ontological model is a model that directly reflects the ontology. There is a simple semantics where each object in the ontology has a direct relationship with the corresponding representation in the model[15].

One of the characteristics of an ontological model is that the representations in it can be regarded as the names of the objects in the ontology – from a Fregean perspective as reference and no sense (from a Millian perspective as denotation without connotation). In [7], Ruth Barcan Marcus (explicitly following in the footsteps of Mill and Russell) calls this 'tagging'.

3.4 Semantic divergence

Semantic divergence occurs where the semantic relationship between the ontology and the representation is not direct and straightforward. This is related to the notion of ontological model – as these have no semantic divergence. The kind of ontological analysis proposed here involves the extraction of an ontological model from a database, and this can be characterised as identifying and removing semantic divergences.

As the examples later in the paper will illustrate, working databases are rarely straightforward ontological models – that is they have semantic divergences. While there are sometimes quite deep reasons for these, they do not have to be profound. It can also just be the exigencies of constructing an application that meets the enterprise's requirements – and then maintaining it – that cause them.

The notion of semantic divergence and semantic heterogeneity overlap – but do not coincide. By itself, semantic divergence does not necessarily lead to semantic heterogeneity. If two databases that need to be integrated have identical semantic divergences, then they are not semantically heterogeneous (vis a vis semantic divergence). However, databases with different semantic divergences (for the same objects) would be semantically heterogeneous.

In practice, much of the semantic heterogeneity in databases has its sources in differing semantic divergences. As the number of databases increases, the likelihood of this kind of semantic heterogeneity also increases. So, in practice, most database integration projects have to deal with significant semantic divergence.

---

[15] This is called *strong reference* in Partridge's *Business Objects: Re - Engineering for re - use* (1996).

Different matching strategies. The distinction between semantic heterogeneity and diversity can be used to characterise the way in which the ontological matching strategy proposed here differs from that typically adopted. Currently many integration projects view the semantic matching process as a mechanism for dealing with semantic heterogeneity – focusing on resolving the semantic differences between the databases. And they analyse these differences using 'real world semantics'. The unified database is then a combination of the homogenous and resolved heterogeneous data. Note that there is no reason why the homogeneous data should be homogeneous with respect to any future applications that need to be integrated. I call this the *heterogeneity strategy*.

The *ontological strategy* focuses on purging the semantic divergence from each of the databases. It analyses the semantics of the databases, making their semantic divergences explicit and mapping the underlying ontology. (This assumes there is a single common ontology underlying the various databases – a point we return to below.) This ontology then provides a basis for designing the "single unified database" that is the output of the integration. Another important benefit is that (unlike the heterogeneity strategy's unified database), this ontology's form is not tied to any of the individual databases, and so should be equally applicable to other applications that cover the same ground – a point that is explored in the later section on reference ontologies.

For simple integrations involving a small number of fixed databases the heterogeneity strategy may be the best solution. However, for more demanding integrations it makes sense, other things being equal, to adopt the ontological strategy.

3.5 Ontological paradigm

There is one more phrase whose use here needs to be clarified: *ontological paradigm*. The term *paradigm* is used here in a sense taken from Kuhn's [8]. He describes how scientific paradigms provide world views determining what kinds of things exist – for example, on p.41 he describes the Cartesian paradigm: "As metaphysical, it told scientists what kinds of entities the universe did and did not contain: there was only shaped matter in motion. As methodological, it told them what ultimate laws and fundamental explanations must be like: laws must specify corpuscular motion and interaction." For Kuhn, paradigms are the frameworks of background assumptions that scientists working in a field share. These are typically make a ruling on matters that cannot (at the time) be resolved so that 'normal' scientific work can focus on areas where it can more easily make progress[16].

People's ontological paradigm works in a similar way. It is the background of ontological assumptions that they share. These typically characterise what kinds of things can exist – and how they can exist. They are often formalised into a number of categories into which the rest of the things that exist fall. These categories are also sometimes known as the top ontology.

---

[16] This point is made on p.6 of Partridge's *Business Objects: Re - Engineering for re – use* (1996).

4 Ontological analysis for semantic matching

The preceding terms can now be used to characterise what ontological analysis for semantic integration is. Ontology provides a framework and suggests a process for the analysis needed for semantic matching. This process focuses on the semantics of the database, identifying semantic divergence. It aims to purge this divergence to produce an ontological model. One key aspect of this model is that it explicitly contains at its top level the categories that form the ontological paradigm.

To give you a flavour of this kind of analysis, this section the paper briefly describes with examples showing how it reveals some of the main forms of semantic divergence. The examples should reinforce the point made earlier, that semantic divergence is ubiquitous in database systems. The first form described here is one of the most persistent and pernicious: divergence at the categorical level – the level of the ontological paradigm.

4.1 Categorical semantic divergence

Databases have a number of top-level categories or metatypes that form a framework. In some texts these database's categories (entity, object, attribute etc.) are – mistakenly – presented as if they were also ontological ones. If this were the case, then an object would always be represented in the same database category in every database. If a car is represented as an entity in one database, it would also be represented as one in ever other database that used the entity category.

Entity type as a design decision. In practice, this usually turns out not to be so. This is a simple point, but it can be difficult to appreciate, so I quote two extracts below to ensure that it is clear. These are from data modelling textbooks and turn on the idea that it is a design decision whether to represent a thing as an entity or an attribute[17].

> "One of the difficulties in designing data entities and identifying their relationship is that it is not always clear whether a data item should be included as an attribute of an entity or constitute an entity in its own right. A simple example to illustrate this point is the data item LOC (location) which is included as an attribute in the EMPLOYEE entity. This could equally be set up as an entity, in which case it would have a one-to-many relationship with EMPLOYEE, as is shown in Figure 7-6. The LOCATION entity may be just a key field, or it may contain attribute data fields. In a like manner the employee address field could also be set up as an entity. No hard and fast rules can be laid down concerning the identification of entities in a database." pp. 194-5 [9].
>
> "There is no absolute distinction between entity types and attributes. Sometimes an attribute can exist only as related to an entity type. In a different context, it can be an entity type in its own right… For example, to a

[17] Part 2 of Partridge's *Business Objects: Re - Engineering for re – use* (1996) has an extended example making this particular point by showing two different designs for the same entities.

> car manufacture, a colour is merely an attribute of one of its products; to the company that makes the paint, a color may well be an entity type." p. 26 [10].

Database categories as a mode of representation. The root of the difference is that the database categories are modes of representation – rather than modes of existence (ontological). In other words, they are categorising the representation rather than what is being represented. Though these may map directly onto the ontological categories – and in an ontological model they would – there is no reason why they should. The database category is not determined by what (category of thing) is being represented, but by how the designer chooses to represent that thing. Some data modelling texts quite explicitly note this difference – as the following extract shows.

> "This cannot be emphasised too strongly. An entity cannot be equated to a record, nor can a record be equated to an entity. There are many facets to this inequality. It should always be borne in mind when thinking about data in a computer because the data ultimately will be *represented* as a collection of records. The data ultimately, however, must be *interpreted* as a set of entities." [11] p. 22.

Saving appearances. Another way to illustrate the representational nature of the database categories is to look at what manoeuvres can be made to save the appearance that they are ontological. Many textbooks take a simple extensional view of the semantics of individual entity types. It may seem that one could use a similar extensional view to reify objects for the database categories can refer to. This provides them with a simple semantics where the representation *entity type* in a database refers to the set of entity types that the particular database's *individual entity type* representations refer to.

One result of this is that it indexes the notion of entity type to the database, with the result that there is no general entity type category, but only a series of categories relative to individual databases. For example, if colour is an entity type in database DB3, then it is a DB3 entity type – and there is a different DB4 entity type, which may or may not have colour as an instance. But there is no general entity type. Under this interpretation, the question whether colour is an entity type does not make sense. This is a substantial revision of database designers' intended meaning. They assume there are entity types per se – irrespective of the individual databases. The interpretation of database categories as representational respects this.

Equivocating type-instance relations. The perception of database categories as ontological is often fatally attractive. One of the main reasons for this is the use of the same names for the representational and ontological categories and an equivocation on the type-instance relation. On the representational level, individual entity types in the database have type-instance relations with their (database) instances. They are also subject to type-instance relations with the (representational) entity type category. There is a natural interpretation of the higher-level entity type to individual entity type-instance relations as representations of corresponding ontological type-instance relations. However, there is no such interpretation for the entity type category to entity type type-instance relations (as there is no natural ontological interpretation of the entity type category).

At the representational level it makes sense to group the lower and higher levels of type-instances together. And this can (falsely) suggest that all the members of this group have a similar natural interpretation - which would then imply that there is an ontological entity category corresponding to the representational one – to act as the ontological type for higher level type-instance relations. This false suggestion is usually reinforced by equivocation on the type-instance relation – switching between talking about the representational and the ontological levels without clearly signposting that this is being done.

Clarifying the semantic divergence of the categories. The semantic divergence of representational categories from ontological ones provides the basis for a substantial amount of ontological analysis. The first task of this analysis is to establish the implicit ontological categories. Often the database and ontological categories will have similar names (in ordinary language) and structures – so they need to be clearly distinguished. The next task is to determine what falls under the ontological categories – as the database categorisation is not a reliable guide.

Examples of categorical divergence. A couple of examples where the choice of representation has been guided by the requirements of the application – rather than ontological considerations – will help to give a flavour of the kind of divergences this kind of analysis reveals.

In almost all databases, amounts, dates (and times) and locations are represented as attributes. So much so that within the database, dates and amounts are normally a specific attribute sub-type. However, they do not have to be represented as attributes – and doing so leads to some ontologically odd results.

Let me illustrate this with a simplified example. MegaBank has as customers Tom and Dick, who both live in London. In MegaBank's database the entries for Tom and Dick both have 'London' address attributes. Today Tom comes into the bank and makes a simple foreign exchange (fx) trade – exchanging a £10 note for 20 Euro one. Dick comes in later and exchanges a different 20 Euro note for the same £10 note. In the bank's database two fx trades are recorded. The first has a buy amount attribute of £10, the second a sell amount attribute of £10. Both have trade and value date attributes of today.

From an application point of view representing the location, amounts and dates as attributes has many information processing advantages. However there are some odd aspects of this representation that suggests there is semantic divergence, these are: *ontic redundancy, missing ontic identity* and *multi-faceted representation*.

There are two address attributes, one is Tom's the other Dick's, but there is only one city, London, that they refer to. There is only one £10 note and one day (today) that the two fx amount attributes and various date attributes refer to. These are examples of *ontic redundancy* – a kind of ontic counterpart to data redundancy. There is one object that is represented many times in the database. Ontic redundancy leads to the same problems as data redundancy – a change to one representation does not propagate automatically to the others.

London, the £10 note and today have an identity. We can identify and re-identify them – and distinguish them from each other and other objects. However, the database has no direct mechanism for tracking their identity. For example, just from the data, one cannot tell whether the same £10 is involved in the two fx trades. This is *missing ontic identity* – the objects have identity but the database does not track it.

The £10 note is represented by an attribute that names an amount – a quantity. This quantity attribute implies the existence of the £10 note and its participation in the fx trade. This is an example of a *multi-faceted representation*: where a single representation is used to describe facets of a number of objects.

There are certain types of attribute that inhere in an entity. My personal height or weight is my attribute. For these forms of attribute, the inhered-in entity (me) underwrites the attribute's identity (and so there is no missing ontic identity). It does this because the attribute cannot inhere in any other entity – my height is mine and not yours (so there is no possibility of ontic redundancy).

For these types of attribute we talk directly about being – 'being height x' or 'being weight y'. Amounts, dates and locations are not of this type. In these cases, we talk about being indirectly – about 'being *for* £10', 'being *done* today' and 'being *in* London'. A database design that was more interested in the ontological aspects of these objects, would treat them as entities with relationships. As, for example, a £10 note entity in a participation relationship with the fx trade. And it would represent the same £10 note entity as related to both fx trades[18].

There are some commercial applications that find it practical to take a more ontological view. To my surprise, a couple of years ago, I came across a security settlements application that stored dates as entity types. In the last couple of years, the designers of the universal data models (for example, [12] and [13]) have chosen to represent locations (addresses) as a series of related entities.

What the ontological form of these types of object is, is a more tricky question. Aristotle (in his 'The Categories') suggests that time and place (location) are kinds of attribute – but the form of his attributes are different from that of database attributes[19].

---

[18] For a model of these three types of attributes, which reveals their ontological form, see Part 6 of Partridge's *Business Objects: Re - Engineering for re – use* (1996).

[19] A point made at length in Part 2 of Partridge's *Business Objects: Re - Engineering for re – use* (1996).

The mechanism for characterising these forms is the ontological paradigm – we examine this more in a later section.

4.2 Lack of representational expressiveness

There are a myriad of sources of semantic divergence. I briefly describe three more common types starting with lack of representational expressiveness.

This can occur at various levels. It can happen at the categorical level. A common situation is where the database categories do not support sub-typing: in other words, where there is no sub-type relation. Then a workaround is needed to represent the ontological sub-type relation[20]. A similar situation occurs when the database does not support higher levels of type-instance relation[21]. One result of both these workarounds is that they create categorical divergence ontological. Entity types are represented by individual database entities, and so the database category of individual entity no longer corresponds with the similarly named ontological category.

Lack of expressiveness can create divergence at lower levels. A couple of years ago I was involved in implementing a securities settlements system that was also being used for fx settlements. In order to accommodate the fx trades, a workaround that treated them like security trades had to be adopted. This involved setting up a new fx security for each combination of currencies for each date there was an fx trade. These 'fx securities' were clearly ontologically spurious – they did not refer to any kind of 'real' security.

One task within the ontological analysis is to unwind these kinds of workarounds to arrive at a clearer representation of what actually exists.

4.3 Indexicality

Another common source of semantic divergence is indexicality. This is where the truth of an expression (representation) depends the conditions of its utterance. A classical example is the expression "I am here" – which is usually true, but will refer to different people and places on different occasions. This is clearly a way in which we use language (representation) and not a way in which the world is.

From an integration point of view, a central issue in the analysis of indexicality is determining which representations are indexed to their specific applications. These need to be unbundled before they can be integrated - as the examples later in this section illustrate. The ontological strategy provides a framework for this. It takes indexicality as a kind of semantic divergence that needs to be refined out of the database to get to the ontological model.

The amount of indexicality in applications soon becomes apparent when one tries to integrate across enterprise boundaries. Most enterprises' systems take the

---

[20] Parent, C. and S. Spaccapietra (2000) *Database Integration: The Key to Data Interoperability* discusses an example of this.

[21] Partridge's *Business Objects: Re - Engineering for re – use* (1996) has an example of this involving car types.

perspective of their particular enterprise as their given context. A common example, also often found in textbooks, is: employee modelled as a sub-type of person. Examination of how the data is used usually reveals that what is meant is that the person is an employee of the particular company that 'owns' the application. So an employee in Megabank's payroll application, is an employee of Megabank.

One can see that there is a semantic divergence here by considering how this application would be integrated with Gigabank's payroll application. The two employee files cannot simply be merged – the result would be a substantial loss of information. The merged file would not tell which of the two banks was the employer.

Ontological analysis reveals the indexicality as a kind of multi-faceted representation, which it unbundles as person and an employed by relation. This unbundled relation can be directly merged without any problematic loss of meaning.

Most enterprise applications abound with indexicality. Examples are:

- Securities trading applications that have security purchase and security sale as types of deal – reflecting the way dealers talk about their trades. The indexicality is revealed by the fact that the trader's security purchase is the counterparty's security sale – and vice versa.
- In many applications there is the notion of legal entity, but this is relative (indexed) to a particular legal system. This point is touched upon in [14] and explored in more detail in [15].
- Commercial applications represent roles that are indexical. For example, the role of counterparty (or customer or supplier) is relative to the party (or supplier or customer). Here an application's customer is the customer's application's supplier.

And the issues of indexicality get complicated when the 'owner' of the application is not clear. For example, an application is being used by a department, but the employer is the company of which the department is part. This kind of identity can get even more problematic during mergers and acquisitions[22] – hence the importance of making a good ontological analysis to purge the indexicality.

4.4 Epistemic characteristics

The final source of divergence described here is epistemic characteristics. The designers of databases need to specify what the database has to 'know'. So its form can be the result of designed epistemic characteristics, which conceal the underlying ontological form.

The clearest examples of this are often found in the constraints imposed on relations. For example, an insurance application may allow for the birth date to be entered for a policy holder – but designate this as optional. From the database designer's (epistemic) perspective, it is enough to specify that the application only has to be able to store this, but does not need to know it for each policy holder. However, from an ontological point of view, each person has a birth date – even if the

[22] A point pursued in Partridge, C's *LADSEB-CNR - Technical report 07/02 - STPO - Synthesis of a TOVE Persons Ontology* (forthcoming).

application does not know it. So birth date is ontologically mandatory, but epistemically optional. The ontological analysis aims to reveal this kind of underlying form.

4.5 Clarifying and explaining

The forgoing discussion and examples illustrate a point the metaphysicians have been at pains to point out. That the ontological analysis raises questions whose answers both clarify and explain what things are being represented and what kinds of thing they are. The explanations help, among other things, to ease the process of gaining agreement across communities – helping to resolve the 'disagreement between communities' mentioned earlier.

4.6 Introducing semantic divergence

Though not part of the scope of this paper, to avoid misunderstanding, it may be worth noting that it is not being suggested that semantic divergence should always be avoided. For example, it is likely that the design of the unified database will involve introducing the semantic divergences that the applications require – maybe re-introducing divergences that have been purged. For example, in a security trading application, dealers may wish to classify their deals as (indexical) security purchase and sales. The point is that semantic divergence gets in the way of ground level semantic matching – and so needs to be purged when doing this.

5 A better analysis

This kind of ontological analysis not only offers a better way to do the existing task, it also produces a better result. This section describes two aspects of this. The first is one of the most valuable from the perspective of the enterprise. It is that the analysis encourages a kind of generalisation that reduces complexity, leading to a simpler, more general model. The second aspect described is less immediately obviously a benefit. This is ontological relativity – which, properly harnessed, can be used to enhance benefits such as generalisation.

5.1 Encouraging generalisation that reduces complexity

Complexity is a major issue for large integration projects. As the size of the individual applications and the number of applications grow, the amount of matching required grows exponentially. Ontological analysis helps as it both prepares the ground for and suggests generalisations.

Describing generalisation. Generalisation is not often described in textbooks – so I clarify what is meant here. It can be usefully contrasted with the notion of abstraction. This has its origins in Locke [16], see for example, ii. xi. 9 and 10 and iii. iii. 6 ff. Central to the notion of abstraction is taking an idea, then, by ignoring some of its details, ending up with a more general idea. For Locke it is the psychological mechanism that enables humans to move from less general to more general ideas – from the idea of an equilateral triangle to that of a triangle[23]. More recently, computer science has taken up Locke's notion.

Generalisation works in a different way. It is similar in that it starts with a collection of types and by analysing commonalities generalises them. It is different in that it makes the original collection redundant, not only without losing any information, but also ending up with a smaller simpler collection of types[24]. Though generalisation is not mentioned much in computing textbooks, it is common practice in actual projects[25]. What is less common is an environment, like that provide by ontological analysis, which encourages generalisation and creates the opportunity for high levels of generalisation.

Outside computing generalisation is recognised as a key feature of the growth of knowledge. For example, Kuhn's [8] regards radical generalisation as a feature of scientific revolutions – where the revolutionary theory is both more general, simpler and typically contains *more* information: information that was not explicitly known at the beginning. This generation of extra information is a common feature of generalisation – within scientific theories it is often known as fruitfulness.

It is clear that semantic heterogeneity and divergence can hinder generalisation. If the commonalities of two entities are represented in a semantically different ways, their similarities are more difficult to see. And the semantic differences may create the appearance of commonality where, in reality, none exists. So ontological analysis clears the ground for generalisation, by making what exists clearer.

Examples of generalisation. But it practice it does more than this. Experience of using the ontological analysis for semantic integration shows it actually encourages, often enforces, generalisation. This can be illustrated using the earlier examples.

The example recognition of the indexicality of the notion of security purchase, turned on seeing that the same thing the application classifies as a security purchase, the counterparty's application classifies as a security sale. This suggests, by analogy, that what the application classifies as a security sale will be classified as a security purchase by the counterparty's application. This, in turn, suggests there is a more

---

[23] People were unhappy with Locke's proposal from the beginning. His near contemporary Berkeley criticised the notion: see for example, paras. 6 ff. and paras. 98, 119 and 125 of his *A treatise concerning the principles of human knowledge.* More recently Frege has made some trenchant criticisms. Dummett's *Frege: Philosophy of Mathematics* (1991) has a good summary of the Fregean concerns.

[24] Partridge's *Business Objects* (1996) has an extended description of this with examples.

[25] For example, Doug Lenat, the progenitor of Cyc, in a posting to the SUO mailing list dated Thu 11/04/2002 wrote "… we continually try to police the KB and find ways to generalize and combine assertions, to REDUCE the number of assertions …".

general class of security trade that underlies both security purchases and sales. Interestingly this more general class also encompasses internal trades – where both parties are 'within' the application. Previously these would be a different class of trade as they were, from the application's point of view neither (or both) purchases or sales. This is a simple, though far from trivial, illustration of fruitfulness.

A similar analysis can be made for customer and supplier – where the generalisation also manages to encompass internal customer-supplier relationships. Ditto for counterparty.

As these examples should make clear, within the ontological framework, integration analysis naturally leads to generalisation. And it can also, usefully engender a culture of generalisation.

5.2 Using ontological relativity to improve integration analysis

For the analysis, and in particular, generalisation, to work smoothly there needs to be a single common underlying ontology. This is an aspect of the ontological framework that now needs to be clarified.

It turns out that there is *ontological relativity*: that different applications can be based upon different ontological paradigms. It is perhaps easiest to see this with a simple example

A simple example of ontological relativity. Consider two of MegaBank's applications. One is for trading securities. Fund managers use it to buy and sell securities, which are recorded in its database. The other is a portfolio performance application. This tracks the long-term performance of the portfolios of securities bought and sold by the fund managers. Both applications keep a record of the securities involved. At first sight it might appear that there is an ontological type, securities, that is represented in both applications – and there is some truth in this. However, closer examination of exactly which securities the applications consider exist throws up some fundamental differences that cannot be simply explained away as semantic divergence.

In the trading application, a natural interpretation of what qualifies as a security changes over time. There is a notion of a security currently existing. Before a new security is issued it does not (currently) exist (ignoring grey markets, to keep things simple) and once it matures it no longer (currently) exists. This makes sense, as a trader cannot trade a security that does not currently exist.

In the portfolio performance application, there is a different natural interpretation. Here securities that have matured are still securities. They have an important part to play in the calculation of the performance. In this application, the notions of coming into and going out of existence seem to have no place. Securities have a lifespan, they get issued and mature, but these events are not burdened with the extra load of indicating existence.

The two applications have different natural interpretation of, say, the representation of IBM's ABC bonds. The trading application would say that IBM's ABC bonds exist now, but (simplifying slightly) will no longer exist after they mature next week.

Whereas the portfolio performance application will say they exist simpliciter. They *cannot* be talking about the same bonds.

The different ontological paradigms underlying these two interpretations are well known to metaphysicians. The trading application has an ontological paradigm that includes a position called presentism that takes tenses as real, affecting existence. The portfolio performance application includes a position called endurantism that does not. (One can find explanations of these positions in metaphysics textbooks such as [17]).

Mapping between the paradigms. As noted above, the ontological analysis has, strictly speaking, clarified that the IBM ABC bonds in the trading and portfolio performance applications are different things. Yet no practical person would want to claim that the two applications are talking about different IBM ABC bonds – they intuitively recognise that the two are, in some sense, the same. They also intuitively recognise that they are not identical. The portfolio performance's database of securities cannot simply be installed in the trading application.

From a database integration point of view, there is a need for a single unified view across the two applications. To do this one needs to accommodate both applications within a single ontology. The natural way to do this, in this case, is to provide an interpretation of the trading application within the portfolio performance application by introducing the (indexical) notion of a currently tradable security – and so currently tradable IBM ABC bonds. This manoeuvre re-interprets (slightly less naturally from the trading perspective) the presentist trading ontology as a semantically divergent version of the endurantist portfolio performance ontology.

Understanding the scope of ontological relativity. As the example makes clear, the ontological paradigm is rarely particularly explicit in the database – or in the minds of the community using the database. Ontological analysis requires the relevant parts of the paradigm be made explicit. Therefore it is helpful if one can start the analysis with some idea of the scope of ontological relativity – in other words, how the ontological paradigms can vary. One way of understanding the differences between the paradigms is to consider them the result of different metaphysical choices[26]. These choices, typically closely related, determine what kinds of things count as existing. Overall, once one has made these choices one has settled on an ontological paradigm. However, different choices lead to different ontological paradigms.

For example, the choice between endurantism and perdurantism involves deciding whether change is real (exists). Similarly, the choice between presentism and eternalism involves deciding whether the passing of time is real.

[26] Partridge's *Business Objects* (1996) offers another route to understanding – it explains how the ontological paradigms evolved in response to various problems.

Converting a liability into an opportunity. For an integration project, at first sight the existence of ontological relativity seems a liability. The task of semantically matching takes on an extra level of complexity. Not only is there a need to determine when different databases are talking about the same thing (within a paradigm) – but also of determining how to map the things that exist according to one paradigm into the 'same but different things' that exist according to another (across paradigms). There is also the task of determining which paradigm underlies each database and which should underlie the unified database.

However, opening up the choice of paradigm presents an opportunity to select one that is most suitable for the integration task at hand – taking into account the various costs and benefits[27]. This differentiates the position here from the situation in the traditional metaphysical debate. Here there is clear goal for the paradigm – application integration – and a basis for measuring the success of achieving it.

There are a number of factors to consider. One that has already been mentioned is the need to deal with the complexity that currently dogs most large integration projects – and how ontological analysis can encourage a form of generalisation the reduces this complexity. It turns out that the metaphysical choices that go into making up a paradigm can have a big effect on the degree of complexity generated[28].

One aspect of the metaphysical choices that go into the paradigm that is well known to metaphysicians is that they can have a multiplying or unifying effect. The multiplying choices lead to a wider range of types of entities and the need to explain how these are related. From a complexity management point of view, this is undesirable – and this needs to be taken into account when deciding upon the ontological paradigm.

A good choice of paradigm is even more important when building a reference ontology – the subject of the next section.

6 Using integration to build a reference ontology

It should also be becoming clear that the application independence of ontological models makes them a prime candidate for reference models. By stripping the applications of the semantic divergences introduced to satisfy their requirements or the design judgements of the designers, a common application-independent foundation is revealed. This can act as the basis for the integration of applications (and a variety of other tasks) in the domains that it covers.

6.1 The yet-to-be completed universal reference model

In an ideal world one might be able to set up a project to produce a universal reference model that could form the basis for integrating any application. In the real world, the

[27] David Lewis's (1986) *On the plurality of worlds* is unusual in having an extended discussion of the need to balance the costs and benefits when deciding upon an ontology.

[28] This is one of the major themes of Partridge's *Business Objects* (1996).

elements of this putative model are often going to be developed piecemeal in particular integration projects. This leaves the meta-integration task of combining these elements.

6.2 Piecemeal development

Using integration projects to develop the pieces of a wider reference model has several advantages. The various databases input to the integration project provide vital 'real' information supplemented during the ontological analysis. Also the unified database is an excellent test bed for the unified ontological model. Users will soon make plain if it does not work.

6.3 Meta-integrating a universal reference model

The task of integrating the various piecemeal models can follow a similar process to the individual integration projects. The task is greatly simplified if a 'standard' ontological paradigm is settled on - as this eliminates cross-paradigm matching.

7 Summary

The theme of this paper is that there is an important stage in the process of integration – semantic matching – that has had insufficient attention until now and can be substantially improved though the application of traditional metaphysical ontology. As the paper has been at pains to point out, this can not only help to improve the existing process but also introduce additional benefits – such as the simplifications that generalisation brings. From a wider perspective, it also enables integration projects to become the engines for the production of a universal reference ontology.

Acknowledgements

This report was produced while on sabbatical leave at the Group of Conceptual Modeling and Knowledge Engineering at CNR-LADSEB, Padova. I would like to thank them for their support. I would also like to thank Bob Colomb for reviewing an initial draft of this paper.

References

1. Sheth, A. and J. Larson, Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases. ACM Computing Surveys, 1990. Vol.22(No.3): p. 183 - 236.

2. Quine, W.V., Ontological relativity, and other essays. The John Dewey essays in philosophy, no. 1. 1969, New York,: Columbia University Press. viii, 165.
3. Parent, C. and S. Spaccapietra, Database Integration: The Key to Data Interoperability, in Advances in ObjectOriented Data Modeling, S. Spaccapietra, Editor. 2000, MIT Press: Cambridge, Mass.
4. Vermeer, M.W.W. and P.M.G. Apers, On the Applicability of Schema Integration Techniques to Database Interoperation. ER 1996, 1996: p. 179-194.
5. Gruber, T.R., A translation approach to portable ontology specifications. Knowledge acquisition, 1993. Vol. 5(No. 2): p. pp. 199-220.
6. Tsichritzis, D. and A. Klug, The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management. 1978: AFIPS Press.
7. Marcus, R.B., Modalities : philosophical essays. 1993, New York: Oxford University Press. xiv, 266.
8. Kuhn, T.S., The structure of scientific revolutions. 2d ed. 1970, Chicago,: University of Chicago Press. xii, 210.
9. Brookes, C.H.P., Information systems design. 1982, Sydney: Prentice-Hall of Australia. 477.
10. Tsichritzis, D.C. and F.H. Lochovsky, Data models. Prentice-Hall software series. 1982, Englewood Cliffs, N.J.: Prentice-Hall. xiv, 381.
11. Thompson, J.P., Data with semantics : data models and data management. 1989, New York: Van Nostrand Reinhold. xviii, 468.
12. Hay, D.C., Data model patterns : conventions of thought. 1996, New York: Dorset House Pub. xix, 268.
13. Silverston, L., W.H. Inmon, and K. Graziano, The data model resource book : a library of logical data models and data warehouse designs. 1997, New York: Wiley. xi, 355.
14. Partridge, C. and M. Stefanova, A Synthesis of State of the Art Enterprise Ontologies: Lessons Learned. 2001, The BORO Program, LADSEB CNR, Italy: Padova.
15. Partridge, C., LADSEB-CNR - Technical report 07/02 - STPO - Synthesis of a TOVE Persons Ontology. (forthcoming). 2002, LADSEB-CNR: Padova, Italy.
16. Locke, J. and P.H. Nidditch, An essay concerning human understanding. 1975, Oxford: Clarendon Press. liv, 867.
17. Loux, M.J., Metaphysics: a contemporary introduction. Routledge contemporary introductions to philosophy; 1. 1998, London ; New York: Routledge.
18. Berkeley, G. and J. Dancy, A treatise concerning the principles of human knowledge. Oxford philosophical texts. 1998, Oxford ; New York: Oxford University Press. vi, 237.
19. Dummett, M., Frege: Philosophy of Mathematics. 1991, London: Gerald Duckworth & Company Ltd.
20. Gilbert, M., On social facts. 1992, Princeton, N.J.: Princeton University Press. x, 521.
21. Goodchild, M., et al., eds. Interoperating Geographic Information Systems. 1999, Kluwer Academic Publishers: Boston.
22. Kent, W., Data and reality : basic assumptions in data processing reconsidered. 1978, Amsterdam ; New York New York: North-Holland Pub. Co. ; sole distributors for the U.S.A. and Canada Elsevier/North-Holland. xv, 211.
23. Papazoglou, M., S. Spaccapietra, and Z. Tari, Advances in object-oriented data modeling. Cooperative information systems. 2000, Cambridge, Mass.: MIT Press. xxiv, 367.
24. Partridge, C., Business Objects: Re-engineering for re-use. 1996, Oxford: Butterworth Heinemann.
25. Partridge, C., LADSEB-CNR - Technical report 04/02 - What is Pump Facility PF101? (forthcoming). 2002, LADSEB-CNR: Padova, Italy.
26. Quine, W.V., The roots of reference. 1973, LaSalle, Ill.,: Open Court. xii, 151.

27. Quine, W.V., From a logical point of view: 9 logico-philosophical essays. 2d ed. 1980, Cambridge, Mass.: Harvard University Press. xii, 184.
28. Searle, J.R., The construction of social reality. 1995, New York: Free Press. xiii, 241.
29. Sheth, A., Changing Focus on Interoperability in Information Systems: From system, syntax, structure to semantics., in Interoperating Geographic Information Systems, C. Kottman, Editor. 1999, Kluwer Academic Publishers: Boston.
30. Strawson, P.F., Individuals, an essay in descriptive metaphysics. 1990, London, New York: Routledge. 255.
31. Strawson, P.F., Analysis and metaphysics : an introduction to philosophy. 1992, Oxford ; New York: Oxford University Press. viii, 144.
32. Lewis, D.K., On the plurality of worlds. 1986, Oxford, UK ; New York, NY: B. Blackwell. ix, 276.