Ontology Mapping to Support Semantic Interoperability in Product Design and Manufacture

N. Chungoora[1] and R.I.M. Young[1]
[1] Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Loughborough, LE11 3TU, UK
{N.Chungoora, R.I.Young}@lboro.ac.uk

Abstract. Ontological engineering is currently being used by a range of functional domains to support the capture and sharing of information and knowledge. It has long been recognised that ontologies provide a basis for sharing meaning. However, several reasons explain how the management of knowledge contained in ontologies can be biased in a number of ways, which inevitably leads to the creation and use of heterogeneous ontologies. This situation is being witnessed in the design and manufacture stages of the product lifecycle, and raises an issue whenever disparate ontologies have to be made interoperable with each other to promote design and manufacturing knowledge sharing among stakeholders. Ontology mapping provides a convenient direction to overcome the problem of ontology heterogeneity. This paper identifies the nature of semantic mismatches and essential elements that need to be taken into account for ontology mapping. Simple examples are provided at various stages to support arguments.

Keywords: Semantic Mismatches, Ontology Mapping, Semantic Interoperability, Manufacturing Knowledge Sharing.

1 Introduction

In the field of product design and manufacture, due to the dispersibility of product and manufacturing knowledge at various stages of the product lifecycle, different functional domains inevitably construct their product and manufacturing ontologies tailored to their needs. The continuing diversity of ontologies is also partly related to ontologies being aligned with particular views of the world, hence resulting in biases and subjective features [1]. Since the definition of concepts in design and manufacture is dependent of the context or view being taken, this clearly identifies that an all-embracing common basis for ontology construction to be adopted by all parties can prove to be very difficult and time-consuming to realise. These incommensurate views of the same functional domain imply incompatible systems, and incompatible systems imply no data sharing, no knowledge transfer, and a necessary duplication of effort [2].

Adopting an all-embracing ontology as a basis for sharing meaning, and as a foundation over which to build up information and knowledge exchanges, remains a very unlikely scenario [1], since in practice, multiple ontologies and schemas will be developed by independent entities [3]. Furthermore, with the widespread distributed use of ontologies, different parties inevitably develop ontologies with overlapping content [4]. These factors, although targeted at the more general problem of ontology heterogeneity, bring evidence of the existence of multiple ontologies developed to suit different functional domains and this is very likely to happen in the design and manufacture stages of the product lifecycle.

In the field of product design and manufacturing engineering, a number of efforts has been sought towards the development of new ontology-based methodologies to capture knowledge behind product geometries, assemblies and process planning among others. For example, in the AIM@SHAPE project [5] many conceptualisations have been pursued, some of which include product design and shape ontologies. Kim et al. [6] have realised an ontology to describe assembly design attuned to the requirements of their domain. On the other hand, the Process Specification Language (PSL) ontology, which explicitly and clearly defines the concepts intrinsic to manufacturing process information, has been developed [7]. The growing use of ontologies is also witnessed in manufacturing enterprises adopting formal conceptualisations for knowledge representation such as at DaimlerChrysler to support a range of design activities [8]. This brief insight provides an awareness of the extent to which heterogeneous ontologies are currently being developed and this accounts for the difficulties associated to seamless knowledge sharing. Therefore, ontology heterogeneity is the primary obstacle for interoperation of ontologies [9]. Hence it becomes of paramount significance to reconcile multiple ontologies.

This paper reveals a spectrum of semantic mismatches that can occur in design and manufacture ontologies. Next, the relevance of ontology mapping as a leap to promote ontology and semantic interoperability is elaborated. We also examine one possible mapping scenario, more specifically concerned with ontology merging through a domain ontology. From this investigation, we reinforce the verity behind semantic mismatches and finally, discussions and conclusions are provided.

2 Semantic Mismatches between Heterogeneous Ontologies

As previously seen, widespread multiple ontologies across a large number of functional domains within design and manufacture make interoperability of knowledge a difficult and perennial task. As a prerequisite to solving the ontology interoperability issue, it is first vital to understand how varied ontological concepts can be and in which ways ontology and semantic mismatches take place, which impede onto achieving seamless interoperability. Semantic mismatches can be interpreted from perspectives such as knowledge elicitation, databases and knowledge representation [1]. For the purpose of this paper, these mismatches are being considered from the knowledge representation side since it is probably the most wide-ranging direction to be taken for understanding them. Appropriate examples are given from perspectives such as design for function, design for manufacture and

manufacturing planning. Some of these examples are related to standard features on parts such as holes. Protégé 3.3 tool has also been used for the simple definition of sample classes and relations in certain cases. From the knowledge representation perspective, a comprehensive classification of semantic mismatches to explain semantic heterogeneity in systems has been proposed ([1], [10]). Two main categorisations of semantic mismatches have been identified, namely conceptualisation mismatches and explication mismatches, which are explained next.

2.1 Conceptualisation Mismatches

Conceptualisation mismatches occur as a consequence of having two or more conceptualisations of a certain domain. These conceptualisations can potentially differ in the way they are defined as ontological entities or in the way they are related within ontologies. Conceptualisation mismatches involve:

Class Mismatches. i.e. the different classes and subclasses present in ontologies.

Categorisation Mismatch. This takes place when in two ontologies the same class has been defined but the class possesses different subclasses. In the following example, both ontologies X and Y identify the concept "Hole_Feature" but in each conceptualisation, different subclasses have been defined (Fig. 1).

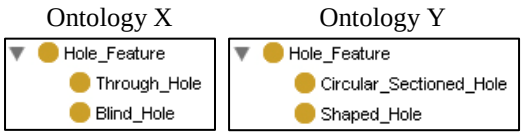

Fig. 1. Conceptual Mismatch

Aggregation-Level Mismatch. This takes place if in both ontologies the same class has been defined but the latter has varying levels of abstraction.

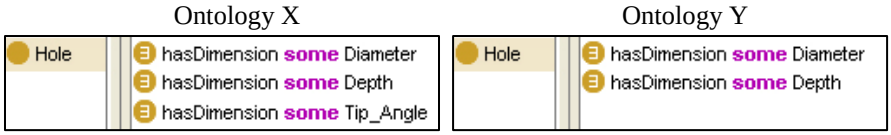

Fig. 2. Aggregation-Level Mismatch

In Fig. 2, the concept of a "Hole" is present in both ontologies X and Y. In ontology X, the concepts "Diameter", "Depth" and "Tip_Angle" are aggregated through the "hasDimension" property in order to define the class "Hole". In ontology Y, only the "Diameter" and "Depth" concepts have been aggregated through the "hasDimension" relation to define the same class "Hole". This clearly shows that the notion of "Hole" in Ontology X is broader than that in Ontology Y.

Relation Mismatches. This type of mismatch is concerned with relations or properties present in ontologies. They involve, for instance, the hierarchical relations between two classes or the assignment of attributes to classes [1].

Structure Mismatch. This is likely to happen when in two ontologies, experts have used the same set of classes but have structured the classes differently using relations/properties. Fig. 3 depicts this structure mismatch and also reveals varying domain semantics as a result of dissimilar levels of granularity. In this example, the "requiresSequence" relation illustrates a range of hole machining processes before a reaming operation can be realised. The "U" symbol refers to the union of the classes. In Ontology Y, the "hasPredecessor" relation is used to identify the necessary preconditions of having "Centre Drilling" U "Drilling" before "Reaming" can be performed. The intent from both parties is almost the same, and can surely be reconciled, but the structure mismatch present leads to potential problems.

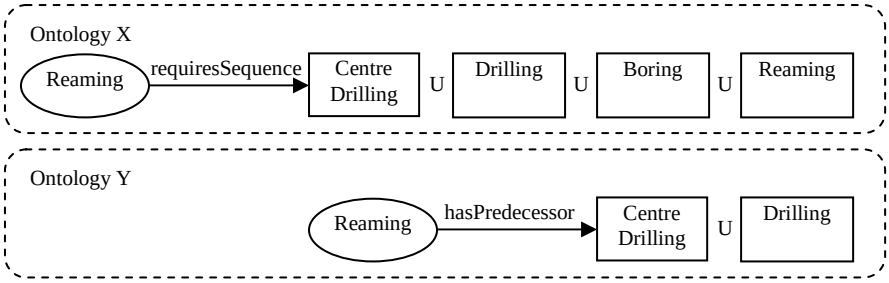

Fig. 3. Structure Mismatch

Attribute-Assignment Mismatch. This form of mismatch is found when the same relation is defined in two separate ontologies, but differ in the way the particular relation is attributed to classes in both conceptualisations. In Fig. 4, the two ontologies do not bear resemblance in the way that the "belongsTo" property has been attributed to the intended classes and/or subclasses.

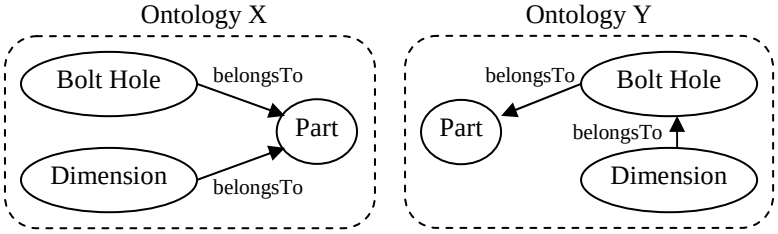

Fig. 4. Attribute-Assignment Mismatch

Attribute-Type Mismatch. This takes place when in two ontologies, the same relation has been defined, but has varying value types, which consequently affects the range of possible values for the instances in both ontologies, for example, a same relation "hasDimension" can be defined as an object property in one ontology and as a datatype property in another ontology, hence resulting in attribute-type conflicts.

2.2 Explication Mismatches

In addition to conceptualisation mismatches, explication mismatches can also occur. Three components are used in order to refer to a definition namely: a term (T) for denoting a particular concept; definiens (D) which provide the building blocks for a definition in the form of aggregated statements; and concept (C) which constitutes the underlying notion to be defined. Examples are provided in each case and expressions used are based around the Semantic Web Rule Language (SWRL) which uses a high-level abstract syntax for Horn-like rules. The → symbol denotes that elements of the expression on the left hand side (i.e. definiens) describe the expression on the right hand side (i.e. the term denoting a concept) after the arrow. The ^ symbol is used to aggregate various definiens. Namespace prefixes are used to relate expressions to their respective ontologies e.g. (X: expression) refers to an expression used in Ontology X.

Concept (C) Mismatch. This occurs when the definitions possess the same terms and definiens but vary in their intent at conceptual level. In other words, the definitions in both cases appear to be identical, when in reality different concepts are being targeted.

X: Hole(?a) ^ hasPlacementFace(?a, ?b) ^ Boss(?b) → Hole_Through_Boss(?c, true)
Y: Hole(?a) ^ hasPlacementFace(?a, ?b) ^ Boss(?b) → Hole_Through_Boss(?c, true)

The first statement says that if a hole (?a) has a placement face (?b) such that (?b) is a boss feature, then the situation of having a “Hole_Through_Boss” (?c) arises. In Ontology X, a “Hole_Through_Boss” refers to a simple hole going through a boss feature. In Ontology Y, a “Hole_Through_Boss” refers exclusively to a tapped hole through a boss feature. In both cases, identical terms and definiens have been used but the concept of a “Hole_Through_Boss” from both domains differ due to the different contexts in which the definitions of “Hole_Through_Boss” are perceived.

Concept and Definiens (CD) Mismatch. This type of mismatch happens when the same term is used by different parties to refer to different “things”, and where different concepts and definiens have been specified.

X: Part_Name(?a) ^ Description(?b) ^ Material(?c) → Part_Spec (?d, true)
Y: Part_Number(?a) ^ Quantity(?b) ^ Despatch_Date(?c) → Part_Spec(?d, true)

The first statement states that if there exist a “Part_Name” (?a), a “Description” (?b) and “Material” (?c), then a “Part_Spec”, i.e. a part specification, is present relating to the characteristics of a particular part. In Ontology Y, for a “Part_Spec” to stand true, the entities “Part_Number”, “Quantity”, and “Despatch_Date” need to be concatenated, where “Part_Spec” reflects the necessary parameters to ship the given part. In both ontologies the “Part_Spec” term is being defined, where the definition is biased to the context, hence the concept of “Part_Spec” being different in both cases, as well as through the definiens specified.

Definiens (D) Mismatch. This occurs when definitions refer to exactly the same concept but differ in the way definiens have been used as a body for the definitions. Consider the next example (Fig. 5) where in both ontologies X and Y, a counterbore hole is being considered. The "hasOverallDepth" (i.e. the total depth of the compound hole feature) has the same concept and the same term in both conceptualisations, but differs in the way definiens have been used to define it.

X: Counterbored_Hole(?a) ^ hasPrimaryDepth(?a, ?b) ^ hasSecondaryDepth(?a, ?c) ^ swrlb:add(?overallDepth, ?b, ?c) → hasOverallDepth(?a, overallDepth)
Y: Cbore_Hole(?a) ^ hasHoleDepth(?a, ?b) ^ hasCboreDepth(?a, ?c) ^ swrlb:add(?depth, ?b, ?c) → hasOverallDepth(?a, ?depth)

The expression for X simply states that the "Counterbored_Hole" has an overall depth "hasOverallDepth", which is equal to the algebraic sum (denoted by the SWRL built-in "swrlb:add") of dimensions labelled (3) and (4). In the second expression, the same concept and term "hasOverallDepth" is being defined and is equal to the algebraic sum of dimensions labelled (3) and (4). These two examples clearly depict a situation which can be classified under the definiens mismatch category, since the same concept and term is used in two separate ontologies to refer to the same "thing", but where different definiens have been used.

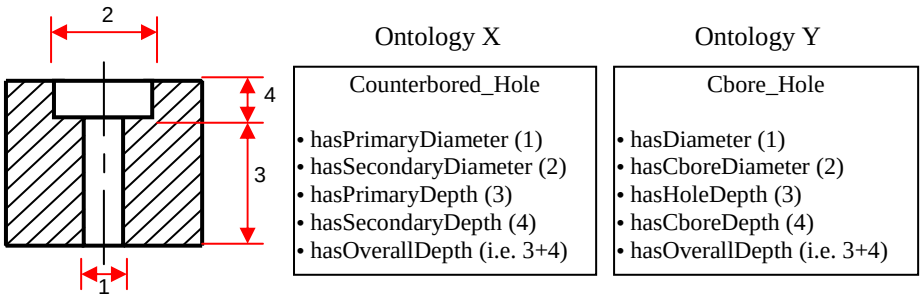

Fig. 5. Definiens Mismatch

Term (T) Mismatch. This form of explication mismatch occurs when definitions share the same concept and definiens, but employ different terms.

X: Cutting_Fluid(?a) ^ Cutting_Speed(?b) ^ Feed(?c) → Min_Process_Requirement(?d, true)
Y: Cutting_Fluid(?a) ^ Cutting_Speed(?b) ^ Feed(?c) → Sufficient_Operation_Parameter(?d, true)

In X, the necessary process parameters that make up a minimum process requirement "Min_Process_Requirement" are "Cutting_Fluid", "Cutting_Speed" and "Feed". In Y, again the necessary process parameters for having a "Sufficient_Operation_Parameter" are "Cutting_Fluid", "Cutting_Speed" and "Feed". The only discrepancy from these two statements lies in the variation in term (T).

Concept and Term (CT) mismatch. This situation arises when dissimilar concepts and terms are identified in ontologies, but where the definitions have the same definiens. In other words, the same "item" is being defined by the same body of definition (definiens) but the concepts and terms used in both cases vary.

X: Engineer(?a) ^ External_Department(?b) ^ collaboratesWith(?a, ?b) → Concurrent_Engineering(?c, true)
Y: Engineer(?a) ^ External_Department(?b) ^ collaboratesWith(?a, ?b) → Subcontracted_Engineer(?c, true)

In Ontology X, it is specified that if an "Engineer" "collaboratesWith" an "External_Department", then "Concurrent_Engineering" practice exists. On the other hand, in Y, it is said that if an "Engineer" "collaboratesWith" an "External_Department" then the "Engineer" is a "Subcontracted_Engineer". Clearly, from the two expressions depicted, the same definiens have been used to refer to an "Engineer" who "collaboratesWith" some "External_Department". However, the concepts "Concurrent_Engineering" and "Subcontracted_Engineer" do not reflect the same underlying concept, and in addition, they use different terms to refer to these.

Term and Definiens (TD) Mismatch. In this form of explication mismatch, which is the converse of the C mismatch, only the term and the definiens vary, whereas the concept stays the same in all distinct cases. The example next is taken from Kim et al. [6] who have devised an ontology to capture knowledge in assembly design. One of their rules is concerned with the definition of assembly/joining relations, and two constraints expressed using SWRL to explain assembly/joining have been identified. Quoted next are the implied constraints and the SWRL rule representing assembly/joining relations in the assembly ontology. Implied constraints are: (1) The associated form features must belong to two non-equivalent parts, (2) The associated form features must be a joining pair. The SWRL rule [6] used to cover the two constraints appear as the first expression below.

X: FormFeature(?x) ^ FormFeature(?y) ^ Part(?z) ^ Part(?a) ^ belongTo(?x, ?z) ^ belongTo(?y, ?a) ^ differentFrom(?z, ?a) ^ isJointPair(?x, ?y) → assemblyJoiningRelationship(?x, ?y)
Y: Object_Feature(?a) ^ Object(?b) ^ formsPartOf(?a, ?b) ^ Object_Feature(?d) ^ Object(?e) ^ formsPartOf(?d, ?e) ^ dissimilarTo(?b, ?d) ^ matesWith(?a, ?d) → matingAssociation(?x, ?y)

In Y, the same constraints form part of the underlying concept but almost completely different definiens and terms are used in the definition. It can be deduced from the two expressions provided that albeit the use of different definiens and terms, exactly the same concept is being referred, i.e. that of defining assembly/joining relations between form features belonging to different parts.

3 An Ontology Mapping Method for Manufacturing Features

In order to support ontology interoperability, it becomes obvious that ontology and semantic mismatches need to be overcome. Interoperability of ontologies and the approaches to solve it remain a core question, and the interoperation process cannot rely on manual input due to the complexity, size and number of ontologies being developed [11]. It is thus clear that there is a need for automatic or at least semi-automatic ways of interoperating ontologies in order to relieve the inconveniences of manually creating and maintaining ontology mappings. Three ways in which heterogeneous ontologies can be made interoperable have been recognised [9] and they are identified as: (1) building inclusion relations between ontologies, (2) building mapping relations between ontologies and (3) building a common ontology from local ontologies.

Out of those three ways to enable the interoperability of heterogeneous ontologies, the most effective method for solving ontology heterogeneity is ontology mapping [9]. Mapping provides a common layer from which several ontologies can be accessed and hence could exchange information in semantically sound manners [12]. With the intention of overcoming problems related to the interoperability of ontologies, effort has been fostered from different groups in order to improve the process of ontology mapping. Several frameworks such as ([13], [14], [15]), methods like ([4], [16], [17]) and theoretical work have been proposed and are still evolving to achieve more promising results of mapping. Fundamental to the task of interoperating ontologies, are a number of commonly adopted ontology interoperability paradigms, where ontology mapping is central to. These are shown in Fig. 6 below.

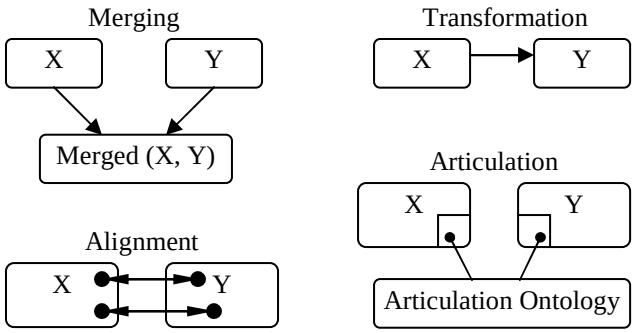

Fig. 6. Ontology Interoperability Methods Involving Mapping (based on [4]).

In this paper our approach to ontology interoperability focuses on the merging process, but the process also conceptually covers some ideas behind other methods such as articulation, namely through the development of a well-defined domain ontology. The concepts present in this domain ontology serve as a reference point for comparing concepts from external ontologies sharing a common context.

3.1 An Ontology Mapping and Merging Example

This section focuses on a simple investigation to understand the key factors behind ontology reconciliation, an essential step prior to achieving ontological and semantic interoperability. Work identified here partly builds up on our understanding of semantic interoperability requirements for manufacturing knowledge sharing [18]. The OWL Plugin of Protégé 3.3 environment has been used for ontology development and we have based our conceptualisation around the definitions and descriptions of holes occurring in design and manufacture. The four main levels involved in the approach consist of: (1) the construction of a domain ontology of design and manufacture hole features whose definitions have been formalised in OWL DL, (2) the identification and definition of two disparate hole feature ontologies which to some degree share a common context, (3) manually mapping and merging the two ontologies into the domain ontology, and (4) using the DL inference engine (FaCT++) in Protégé OWL as a basis to extract knowledge that has not been directly asserted. Fig. 7 below identifies the four-step process and relevant mechanisms, namely the user and the ontology framework, interacting with the process.

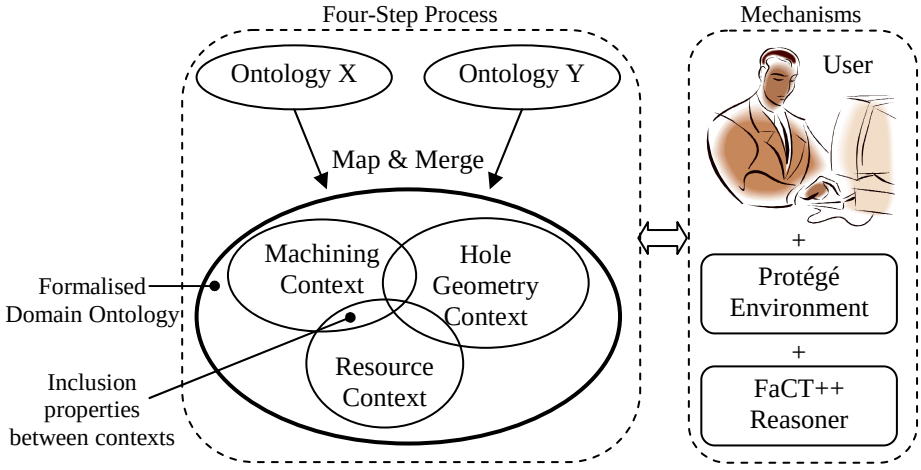

Fig. 7. Interaction of User and Ontology Framework with the Mapping Process

The formalised domain ontology acts as an ontology where external ontologies can be articulated to before mapping. Apart from defining hole concepts from a geometrical context, the domain ontology also captures other contexts such as a hole machining process context and a manufacturing resource context, both pertinent to hole machining processes. The recognition of the need to include different but interlinked contexts during ontology construction has been made [18] and the domain ontology in this example also uses inclusion properties defined to relate the different contexts together to enrich the semantics of concepts in the ontology. The formalised domain ontology on one side describes hole feature concepts from a geometrical viewpoint and clearly depicts all the “necessary” and “necessary and sufficient” conditions for the description of these concepts, e.g. a “Counterbore_Hole” from the

domain ontology has a set of "necessary" and "necessary and sufficient" conditions. A formal definition of "Counterbore_Hole" in the ontology is as shown in Fig. 8.

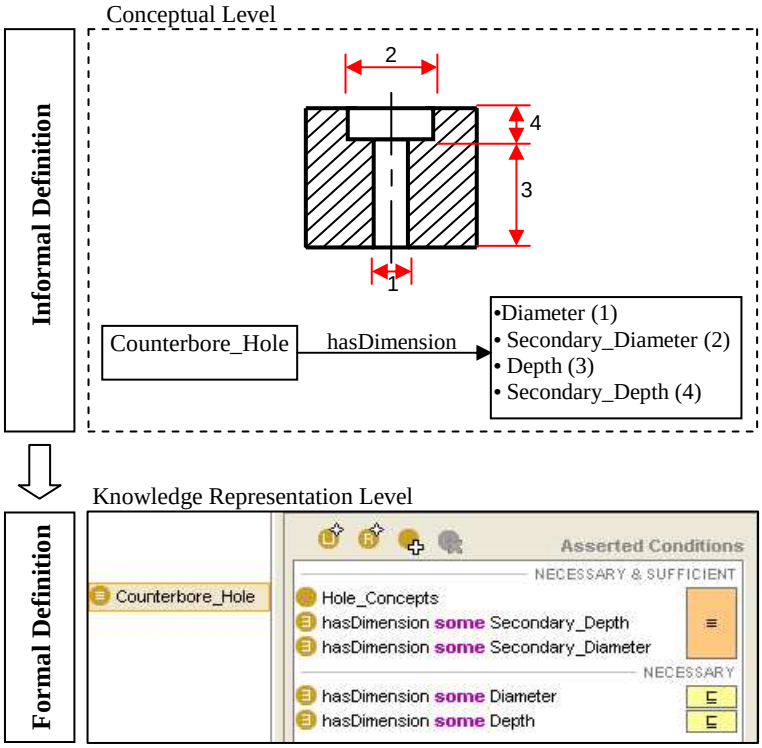

Fig. 8. From Informal to Formal Definition of a Counterbore Hole in the Domain Ontology

What is actually being implied through the various asserted conditions is that if there exists a random class that satisfies any of the "necessary and sufficient" conditions in line with those of a "Counterbore_Hole", then that random class can be inferred by the DL reasoner as being a "kind of" "Counterbore_Hole". Conversely, having a random class satisfying all the "necessary" conditions alone without satisfying "necessary and sufficient" conditions does not imply that the random class is a "kind of" "Counterbore_Hole". This type of reasoning is key behind the inference engine for the deduction of new knowledge and is used after the merging process is completed for finding commonalities between the two disparate ontologies based on the merged ontology.

3.2 The Mapping and Merging Process

As previously seen, each hole concept present in the ontology possesses a set of asserted "necessary" and "necessary and sufficient" conditions, which brings higher formality to definitions. In order to reconcile two disparate ontologies sharing a similar context to that of the domain ontology, a number of steps has to be considered

for manually being able to map concepts from ontologies X and Y to the domain ontology. First, the external ontologies have to be normalised to the representation language of the domain ontology. It is of fundamental importance that during comparison and mapping from an external ontology to the domain ontology, all entities from the external ontology find their correct match in the domain ontology, implying that the latter has to be sufficiently broad to capture large domain knowledge. Fig. 9 depicts an example where essentially the same hole concept appears in the two disparate ontologies but different terms and definiens have been used to describe the concept (see TD Mismatch). The mismatch between "C-Bore" and "Counterbored_Hole" also overlaps onto aggregation-level mismatch and structure mismatch. Reconciliation of the two concepts has to be done using the formalised definitions present in the domain ontology.

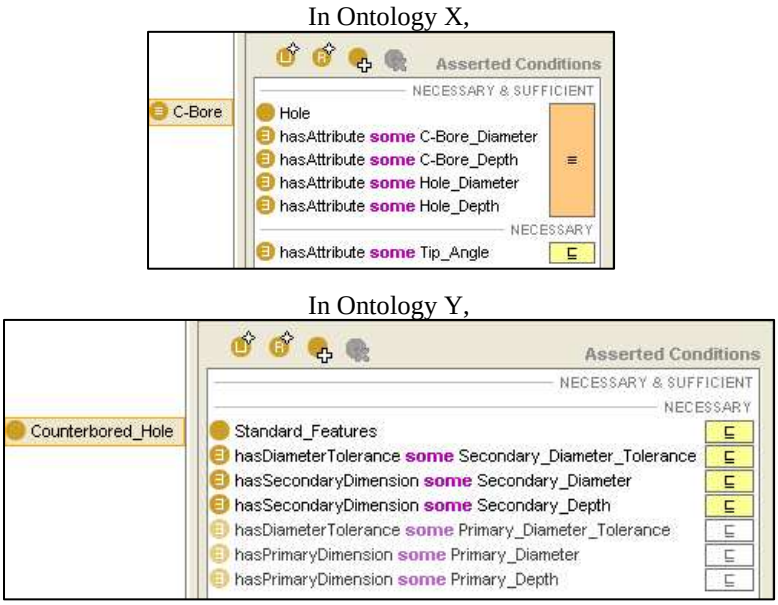

Fig. 9. Example of Classes and Properties for Mapping to the Domain Ontology

The methodology adopted for manual mapping is based around the decisions and actions input by the user to ensure consistent ontology reconciliation. In other words, the methodology depicts the knowledge that is required to carry out the mapping process and subsequent merging of concepts from ontologies X and Y to the domain ontology. The most important mapping and merging steps are identified next and take into account the examples shown in Fig. 9.

- For each class from the external ontologies to be mapped and merged, create the same class in the domain ontology in the most obvious hierarchy of concepts where that class fits, e.g. the "C-Bore" class is created as a child in the "Hole_Concepts" parent class from the domain ontology. Naming clashes can simply be resolved by prefixing concepts.

- Each external attribute is matched and replaced by a best-fit property in the domain ontology, e.g. the "hasAttribute" property of "C-Bore" is substituted by the "hasDimension" property from the domain ontology. Although "hasAttribute" as a textual statement does not directly indicate its connection to "hasDimension", it becomes feasible to suggest the link between the two properties based on the fact that in Ontology X, "hasAttribute" is used to aggregate a set of dimensional parameters implying equivalence to "hasDimension". Ontology Y, on the other hand, identifies two levels to relate dimensions to holes namely through "hasPrimaryDimension" and "hasSecondaryDimension". These properties in Ontology Y do not carry any formal semantics and are interpreted as being equivalent to the more general "hasDimension" property. Embedding more formal semantics to distinguish "hasPrimaryDimension" from "hasSecondaryDimension" would require the consideration of more expressive logic with rules.
- A "filler" class used in a restriction for describing and defining a concept is matched with the appropriate class in the ontology based on lexical and structural similarity as well as intent, e.g. "Hole_Diameter" of a "C-Bore" hole is a type of "Hole_Dimension_Parameter" but more specifically a type of "Diameter_Parameter" in the domain ontology. Hence, the class "Hole_Diameter" is recreated as a child of the "Diameter_Parameter" class. This matching and merging step has to be completed for all filler classes used in both external ontologies.
- Having created all necessary classes, synonymy among classes is specified within the domain ontology using the "hasSynonym" property which is symmetric and transitive in nature, e.g. it can be specified that "Hole_Diameter" and "Primary_Diameter" now present in the domain ontology are synonymous concepts.

3.3 Performing Inferences Based on the Merged Ontology

After mapping and merging concepts form the external ontologies to the domain ontology, the next step consists of performing an inference on the main merged ontology by using the DL reasoner. One simple inference consists of a reclassification of the taxonomy to identify subsumptions not explicitly asserted in the first instance. On running the inference engine, the individually defined hole concepts of "C-Bore" and "Counterbored_Hole", now present in the domain ontology, appear as subclasses of the formally defined "Counterbore_Hole". Furthermore, even after computing the taxonomy, "C-Bore" and "Counterbored_Hole" still appear as individual non-equivalent concepts, thereby preserving initial semantics.

Fig. 9. Inference-Based Taxonomy Classification for "Counterbore_Hole" Class

4 Discussions and Conclusions

The four-step approach used to reconcile heterogeneous ontologies with the help of a domain ontology with inclusion properties among various contexts is interesting since it explores the possibility of having an ontology mapping and merging model which is a hybrid of different techniques for ontology interoperability. Capturing the knowledge that leads to the mapping and merging process can be very useful to support the interoperability of heterogeneous ontologies in the design and manufacture domains. One of the reasons why only manual mapping and merging of the ontologies have been done at this stage is because of the necessity to develop a good understanding of the knowledge that the user has to employ during the process. Manual mapping can be a labour-intensive task [19] that requires careful analysis of ontologies in order to understand all the entities present, but it can prove to be accurate for dealing with small ontologies and can also help resolve missing knowledge behind concepts that are not well-defined. Manual mapping loses its feasibility when large ontologies need to be reconciled. For convenience in this investigation, only small ontologies consisting of a taxonomy of classes, properties and definitions based on restrictions have been considered.

In Section 3.2, a mapping methodology has been proposed, the latter being based on the thought process and decisions made during mapping and merging. It is necessary to build up and formalise this mapping knowledge so that it can effectively be applied as an algorithm to enable automatic/semi-automatic implementations, thereby saving an enormous amount of time, while making an ontology mapping system more robust and extensible. In this paper, the mapping investigation predominantly revolves around the reconciliation of classes, properties, relations and restrictions. Ontologies may also include instances carrying particular knowledge and axioms which bring semantic enrichment and at the same time restrict the interpretations of concepts in an ontology. Therefore, it is also important that ontology interoperability involves the mapping of instances and axioms as well.

On the other hand, the experiment performed shows that during the mapping process, individual entities in separate ontologies do need to find corresponding equivalent matches in the domain ontology. However, it should not be forgotten that small segments of knowledge around a given entity also play a crucial role in enabling a feasible mapping decision to be made by reducing semantic ambiguities. Observations made during the four-step approach has allowed the specification of a number of factors that need to be accounted for in the quest for ontology and semantic interoperability in the design and manufacture domains. These factors are:

- Individual external ontologies requiring mapping need to be normalised to a standard ontology representation language, which formally captures semantics of domain models.
- A mapping environment should emphasise on the identification of synonymous concepts and similarity among entities from two ontologies sharing a similar domain. Similarities can, for example, be related to lexical similarity or similarity through embedded rules.

- It is necessary to identify ontology and semantic mismatches early in the process and the user needs to be able to view and understand the nature of these mismatches before mapping can be performed. Hence, appropriate actions can be taken to overcome semantic mismatches between ontologies and improve interoperability.
- A formal algorithm has to be defined as a basis for automating a mapping process. The algorithm, therefore, needs to work hand in hand with identifying semantic mismatches while at the same time pursuing the correct actions for ontology reconciliation.
- Knowledge inference should not be limited to taxonomical classification alone. Instead, a user should be able to query the mapped ontologies in order to derive maximum constructive knowledge from the system.
- It is important to set up a framework with an appropriate user interface (UI), which facilitates user-system interaction.

The task of designing, implementing and maintaining ontology-based systems requires adequate support for ontology matching [20]. Ontology matching and reconciliation is an essential step to achieve semantic interoperability for promoting manufacturing knowledge sharing. Several frameworks, methods and tools are present in order to deal with ontology interoperability. However, these techniques do not encompass sufficient potential to resolve interoperability in design and manufacture, since the latter is an expert domain with very specific content and issues. It is intended that our future work shall address further issues in regard to ontology and semantic interoperability in product design and manufacture and shall also explore richer semantic structuring through more expressive representation formalisms such as Common Logic (CL) [21] and the Process Specification Language (PSL) [22].

Acknowledgements. The work presented herein was funded through the Research Studentship in the Wolfson School of Mechanical and Manufacturing Engineering of Loughborough University and undertaken in conjunction with the Knowledge and Information Management (KIM) Through-Life Grand Challenge Project (www.kimproject.org) funded primarily by the Engineering and Physical Research Council (EPSRC – Grant No. EP/C534220/1), the Economic and Social Research Council (ESRC – Grant No. RES-331-27-0006) and Loughborough University's Innovative Manufacturing and Construction Research Centre (IMCRC – Grant No. EP/E002323/1).

References

1. Hameed, A., Preece, A., Sleeman, D.: Ontology Reconciliation. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. International Handbooks on Information Systems, pp. 231--250. Springer, ISBN 3-54040834-7 (2004)
2. Hovy, E.: Combining and Standardising Large-Scale, Practical Ontologies for Machine Translation and Other Uses. In: 1st International Conference on Language Resources and Evaluation (LREC), pp. 535--542. Granada, Spain (1998)

3. Madhavan, J., Bernstein, P.A., Domingos, P., Halevy, A.: Representing and Reasoning about Mappings between Domain Models. In: 18th National Conference on Artificial Intelligence (AAAI'02). Edmonton, Alberta, Canada (2002)
4. Noy, N.F., Musen, M.A.: The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping. International Journal of Human-Computer Studies 59, 983--1024 (2003)
5. Advanced and Innovative Models and Tools for the development of Semantic-based systems for Handling, Acquiring and Processing knowledge Embedded in multidimensional digital objects, http://www.aimatshape.net/
6. Kim, K.-Y., Manley, D.G., Yang, H.: Ontology-based Assembly Design and Information Sharing for Collaborative Product Development. Computer-Aided Design 38, 1233--1250 (2006)
7. Schlenoff, C., Gruninger, M., Ciocoiu, M., Lee, J.: The Essence of the Process Specification Language. In: Special Issue on Modeling and Simulation of Manufacturing Systems in the Transactions of the Society for Computer Simulation International (1999)
8. Lukibanov, O.: Use of Ontologies to Support Design Activities at DaimlerChrysler. In: 8th International Protégé Conference. Madrid, Spain (2005)
9. Liping, Z., Guangyao, L., Yongquan, L., Jing, S.: Design of Ontology Mapping Framework and Improvement of Similarity Computation. Journal of Systems Engineering and Electronics 18(3), 641--645 (2007)
10. Visser, P.R.S, Jones, D.M, Bench-Capon, T.J.M, Shave, M.J.R.: An Analysis of Ontology Mismatches: Heterogeneity vs. Interoperability. In: AAAI 1997 Spring Symposium on Ontological Engineering. Stanford, USA (1997)
11. Ehrig, M., Sure, Y.: Ontology Mapping: An Integrated Approach. In: 1st European Semantic Web Symposium. LNCS , vol. 3053, pp. 76--91. Springer, Heraklion, Greece (2004)
12. Kalfoglou, Y., Schorlemmer, M.: Ontology Mapping: The State of the Art. Knowledge Engineering Review 18(1), 1--31 (2003)
13. Maedche, A., Motik, B., Silva, N., Volz, R.: A MApping FRAmework for Distributed Ontologies. In: 13th International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web. LNCS, vol. 2473, pp. 235--250. Springer, Siguenza, Spain (2002)
14. Kiryakov, A.K., Simov, K. Iv., Dimitrov, M.: OntoMap: Portal for Upper-Level Ontologies. In: 2nd International Conference on Formal Ontology in Information Systems (FOIS'01). Ogunquit, Maine, USA (2001)
15. Fernández-Breis, J.T., Martínez-Béjar, R.: A Cooperative Framework for Integrating Ontologies. International Journal of Human-Computer Studies 56, 665--720 (2002)
16. McGuinness, D., Fikes, R., Rice, J., Wilder, S.: An Environment for Merging and Testing Large Ontologies. In: 17th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000). Colorado, USA (2000)
17. Stumme, G., Maedche, A.: FCA-Merge: Bottom-up Merging of Ontologies. In: 7th International Conference on Artificial Intelligence (IJCAI'01), pp. 225--230. Seattle, Washington, USA (2001)
18. Chungoora, N., Young, R.I.M.: Semantic Interoperability Requirements for Manufacturing Knowledge Sharing. In: Mertins, K., Ruggaber, R., Popplewell, K., Xu, X. (eds.) Enterprise Interoperability III: New Challenges and Industrial Approaches. pp. 411--422. Springer-Verlag London Ltd. (2008)
19. Mitra, P., Wiederhold, G.: Resolving Terminological Heterogeneity in Ontologies. In: 15th European Conference on Artificial Intelligence: Workshop on Ontologies and Semantic Interoperability. Lyon, France (2002)
20. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer-Verlag New-York Inc., Secaucus, NJ, USA (2007)
21. Common Logic (CL), http://cl.tamu.edu/
22. Process Specification Language (PSL), http://www/mel.nist.gov/psl/