Mediation Of Foundation Ontology Based Knowledge Sources

Najam Anjum
Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University
Loughborough, LE11 3TU
n.a.anjum@lboro.ac.uk

Dr. J. A. Harding (j.a.harding@lboro.ac.uk), Dr. R. I. Young (r.i.young@lboro.ac.uk)
and Prof. K. Case (k.case@lboro.ac.uk)
Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University
Loughborough, LE11 3TU

Abstract:
Ontologies are helpful in giving interoperable structures to sources of knowledge and information. This interoperability, however, is greatly hindered by the heterogeneity of independently developed ontologies which in turn increases the requirements for mediation systems to reconcile the differences. A core concepts ontology for a certain domain contained by a foundation ontology can be used to alleviate this problem and to facilitate the reconciliation efforts. Possible differences in the use of concepts from the core concepts to model entities in domain ontologies can be prevented by binding the domain ontology developers to some rules. These rules can be particularly useful for domain ontologies requiring some kind of traceability of their concepts in the foundation ontology. The mediation system can then use this traceability to establish similarities between two ontologies. Software applications, like the one explained in this paper, can then be developed to perform the mediation task automatically and accurately.

Keywords: Interoperability, knowledge verification, ontology mediation, foundation ontologies, domain ontologies, Common Logic.

1. Introduction

Ontologies, being the explicit and formal specification of a conceptualization (Gruber, 1993), provide a good platform for building shareable and interoperable knowledge repositories. They not only provide a way to preserve knowledge but also enable one to produce pre-packaged sets of information and knowledge available for individual use or for constructing large knowledge sets by using them as building blocks (Neches et al, 1991). Since the inception of this concept in the field of computer science, different types and levels of ontologies have risen. Two of these types which are relevant here are foundation and domain ontologies. Foundation ontologies are a classification of very general concepts while domain ontologies belong to a specific and narrower field of knowledge or information. Foundation ontologies can be used as a mediation platform to find similarities among different domain ontologies developed independently and thus are very helpful in increasing interoperability in ontology based systems. The term mediation here refers to the process of reconciliation through which a software agent finds similarities in two ontologies in order to make them interoperable. To aid this mediation and interoperability between domain ontologies, a set of common core concepts belonging to a certain domain can be used to build them. This set of common concepts, held under the top level or foundation ontology, provides building blocks and a foundation for the construction of domain ontologies. The commonality existing in domain ontologies then helps mediation systems to resolve differences and find similarities.

This paper explains a novel mediation approach based on this idea. Description of an application working on these principles is also presented, the successful working of which further validates the viability of this approach. This software application is integrated with an ontology editor to perform the task of ontology mediation and knowledge verification using foundation ontologies through structural and semantic methods. The paper starts with a brief review of relevant literature and this is followed by an explanation of the mediation framework. The discourse concludes with a look into possible enhancements of the framework and future research directions.

2. Foundation Ontologies

To make knowledge bases more shareable and expandable, instead of building them from scratch, it is more appropriate to develop them out of a single agreed upon foundation or standard (Neches et al, 1991). Foundation ontologies, as their name suggests, provide the basis for this standard. They make the expansion and integration of knowledge bases easier. This is because if two system builders build their knowledge bases on a common ontology, the system will share a common structure, and it will be easier to subsequently merge and share the knowledge bases (Swartout et al, 1997). These knowledge bases may also be accompanied by their own ontologies relevant to a specific domain. Such ontologies are called domain ontologies and as they provide a set of terms for describing some domain they can be thought of as taxonomies of relevant objects within that domain (Swartout et al, 1997). In these ontologies local vocabularies instead of standardized global vocabularies are formed in a particular context (Yang & Zhang, 2007). Example of domains may include aerospace, biology, manufacturing, arts etc.

Some examples of existing foundation ontologies can be seen in the literature. The most famous of them include Standard Upper Ontology – SUO (Niles & Pease, 2001), Suggested Upper Merged Ontology – SUMO (Niles & Pease, 2001), WordNet (Deng et al, 2009), DOLCE (Gangemi et al, 2002), and Cyc Ontology (Matuszek et al, 2006). Foundation ontologies like these may help to reduce semantic heterogeneity by restricting domain ontology builders to match their own conceptualisations against a common foundation, so that all communication is done according to the constraints derived from the ontology (Schorlemmer & Kalfoglou, 2005). These constraints, in a way, serve as a means of binding domain ontology builders to an ontological commitment. Ontological commitment is the process in which interested parties agree on the use of terminologies in an ontology. It helps in defining precisely the meaning of a term (Gomez Perez et al, 2004) and thus also helps in sharing knowledge accurately with minimal misinterpretation. It is this agreement, in the form of constraints, which is the basis of the mediation approach described in this paper.

2.1. Ontology Mediation Through Foundation Ontologies

Existing domain ontologies can also use foundation ontologies to communicate with other independently developed ontologies. This is done by first aligning domain ontologies with concepts in a foundation ontology and then basing on these alignments between domain and foundation ontologies, the similarities between the domain ontologies are established as shown in figure 1. Some examples of this use of foundation or upper ontologies can be seen in the literature. For example, a tool named LOM is developed by Li (2004) which uses WordNet, SUMO and MILO to communicate between two ontologies. Aleksovski et al (2006) have used the DICE ontology as background knowledge. They use this background knowledge to match two flat unstructured lists of concepts. In a different work, an algorithm is developed by Mascardi et al (2007) which uses upper ontologies to align two heterogeneous ontologies. In their later work, Mascaradi et al (2010) experiment with

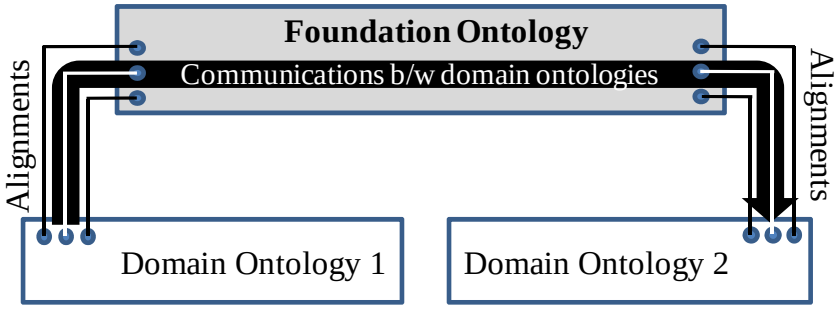

Fig. 1. Communication between domain ontologies via a foundation ontology

OpenCyc, SUMO-OWL and DOLCE and use these foundation ontologies as semantic bridges to match ontologies. In all of these cases independently developed heterogeneous ontologies are provided with a pathway to communicate with each other by using an upper ontology as a semantic bridge. These semantic bridges can be more effective and useful if they are developed during the domain ontology building phase. Ontology mediation and knowledge verification systems then can use these bridges to establish similarities between two independently developed domain ontologies. This can be done by providing a set of core concepts along with the foundation ontology for the development of domain ontologies. To ensure that these bridges are built during the domain ontology development stage, some restrictions may need to be put in for the use of these concepts. The restrictions can bind domain ontology builders to assign a proper traceability attribute, or establish a semantic bridge, to a new concept when they define elements in the domain ontology. These traceability attributes may include a relation or a simple subsumption which connects a domain concept to its counterpart in the foundation or core concept ontology. Ontology mediation tools then can be designed which specifically look for this traceability attribute in order to follow domain concepts to their foundation origin and then compare them with concepts from other ontologies connected to the same foundation concept. Through this process similarities can be discovered between two domain ontologies. The Semantic Manufacturing Interoperability Framework (SMIF) of Changoora (2010) also uses a foundation ontology and mapping models to manage reconciliation of concepts across domains. The work presented in this paper, however, gives a clearer picture and a practical way of how a verification system may work using the connections of domain ontology concepts with their counterparts in the foundation ontology. At this point it is necessary to examine the methodologies which existing ontology mediation tools follow in order to find similarities.

2.2. Conventional Similarity Discovery Techniques

Existing ontology matching and mapping tools use mainly four different types of algorithms to detect similarities between the source and target ontologies (Aleksovski et al, 2006). These tools either use:

1- the lexical matching of concepts, where concept names in an ontology are compared with those in other ontologies,
2- the lexical matching of instances, where instances of concepts are matched for similarity,
3- the structural methods, where the hierarchical structure of two ontologies are compared and
4- the semantic methods, where additional logic is used to detect similarities.

Almost all of the ontology matching and mapping tools use lexical matching methods while few of them additionally use the other two techniques to find similarities as can be seen in table 1 which lists some ontology matching tools and techniques and their similarity detection approaches. It is needless to say that the list of tools here is not exhaustive and a plethora of other tools can be found in the literature. The selection of the tools included in this table is

Table 1
Ontology matching tools, their matching techniques

S.No.	Technique	Similarity Features	Matching Technique
1	**MAFRA** (Mapping FRAmework) (Maedch et al, 2002)	Concepts, and relation names	Object Identity Establishment, Statistical Analysis of Transformations
2	**IPROMPT** (Noy & Musen, 2003)	Concept names	Lexical Matching
3	**AnchorPROMPT** (Noy & Musen, 2003)	Concept names and taxonomic structure	Hierarchical structure and nodes matching
4	**GLUE** (Doan et al, 2003)	Concept Instances	Similarity metrics (Probability of similarity of Instances)
5	**QOM** (Ehrig & Staab, 2004)	Concepts, relations and instances names	String equality, logical assertions equality
6	**ONION** (Mitra & Wiederhold, 2002)	Concept names	Context development through corpus based word relater
7	**FCA-Merge** (Stumme & Maedche, 2001)	Concept names	Context development from corpus of domain specific documents
8	**Chimaera** (McGuinness et al, 2000)	Term names, presentation names, term definitions, possible acronym and expanded forms, names that appear as suffixes of other names	Development of name and taxonomy resolution list

made on the basis of two state of the art ontology mediation tools surveys by Kalfoglou & Schorlemmer (2003) and Lourdusamy & Ganapathy (2008).
The tool AnchorPROMPT shown here needs some explanation as the technique it uses has some similarities with the work presented in this paper. AnchorPROMPT uses the structural method to find similarities. It scans the nodes of the branch of the ontological hierarchy carrying the concept to be matched. In this technique, the concept names in a certain branch are matched with an apparently similar branch in another ontology. A scoring system then determines if the selected portions of the ontology are actually conceptualizing the same thing. The similarity detection technique of the tool presented in this paper is a combination of structural and semantic methods. Like AnchorPROPMPT it follows a concept along its hierarchical lines but it does not use the technique of node matching of ontological branches.

The effectiveness of the tools working on the methodologies discussed here can, in one way, be assessed by looking at their ability to detect and resolve mismatches that exist in two heterogeneous ontologies. This analysis is presented in the next section.

2.3. Mediation Tools and Ontological Mismatches

The primary method most ontology mediation tools use to find similarities is lexical matching as discussed above. Similarity finding through this method has a very low probability of success if the mismatches existing between ontologies are of a conceptual nature. Visser et al (1997) define two main categories of mismatches that may exist in ontologies to be matched. These are conceptualization and explication mismatches. Conceptualization mismatches occur either due to a difference in the way concepts are

distinguished in an ontology or the way they are related. Explication mismatches, on the other hand, are due to differences in the way concepts are explained or defined in two ontologies (Visser et al, 1997). Some other definitions of ontological mismatches also exist in the literature but broadly they all come under either of these two mismatches defined by Visser et al (1997). Anjum et al (2010) analyze some existing ontology matching and mapping tools to determine their capability of detecting and resolving ontological mismatches. This analysis

Table 2
Analysis of Mapping Tools and Techniques from the Mismatches Point of View
(Adapted from Anjum et al (2010))

A – Automatic, U – Suggests solution to the user, M – Provides Mechanism, Mm - Mismatches

Semantic Mismatches		**MAFRA**		**PROMPT**		**Anchor-PROMPT**		**GLUE**		**QOM**		**ONION**		**FCA-Merge**		**Chimera**	
		Detection	Resolution	Detection	Resolution	Detection	Resolution	Detection	Resolution	Detection	Resolution	Detection	Resolution	Detection	Resolution	Detection	Resolution
Conceptualization Mm	Categorization Mm					M	U			M						M	
	Aggregation-level Mm					A	U			M						M	
	Concept Description Mm									M						M	
	Coverage Mm									M						M	
	Single vs Multi valued property							M		M						M	
	Unique vs Non-unique valued property							M		M						M	
	Structure Mm									M						M	
	Attribute-assignment Mm									M						M	
	Attribute-type Mm									M						M	
	Alignment conflict among disjoint relations									M						M	
Explication Mm	Concept & Term Mm									M						M	
	Concept & Definiens Mm (Homonyms)									M						M	
	Concept Mm									M						M	
	Term & Definiens Mm (Synonyms)	M		M	U	A	U	A	U	A	U	M	U	A	U	A	U
	Term Mm	M		M	U	A	U	A	U	A	U	M	U	A	U	A	U
	Definiens Mm	M		M	U	A	U	A	U	A	U	M	U	A	U	A	U

shows a trend that most of the tools are just capable of finding explication mismatches as shown in table 2. For a description of all the different types of mismatches shown in the table please see Anjum et al (2010).

The mismatches explained here occur either due to the difference in the structure in which the concepts are arranged i.e. the conceptualization mismatches, or the way they are defined i.e. explication mismatches. These potential differences can be prevented, in one way, if domain ontology builders commit their ontologies to a single common foundation. The next sections of the paper describe the mediation tool developed for foundation based domain ontologies and the framework its working is based on.

3. Project Background

Before the framework and its working is explained, it is useful to note that the research reported here is a part of a bigger project entitled 'Interoperable Manufacturing knowledge System' (IMKS). IMKS is researching the potential for improvement in interoperability when knowledge is shared through domain ontologies which are developed out of a common foundation ontology of core concepts. Two domains selected to experiment with this idea are design and manufacturing. The project features a foundation ontology of core design and manufacturing concepts and two domain ontologies developed out of these foundation concepts. A mediation system sits in the middle responsible for making sure that the knowledge shared between two domain ontologies is correctly understood. It does this by locating similar concepts in two ontologies. The ontology development in this project is being done in IODE (Integrated Ontology Development Environment) which takes input in the form of Common Logic based ontologies.

4. The Verifier – An ontology mediation tool

Keeping in view the capabilities and shortcomings of existing ontology mediation tools, the new tool needs to have the capability to make the similarity detection process more automatic and accurate when two foundation ontology based domain ontologies are matched. The components needed for testing this application include a core concept foundation ontology, two domain ontologies built by using the concepts from the core concept ontology, an ontology editing environment and a user interface which makes the use of this editing environment easier and user friendly. These four components are defined in the next sections.

4.1. Ontology Formalism and Editing Environment

Ontologies are built here using the Knowledge Frame Language (KFL). KFL is a convenience layer of syntactic sugar that sits on top of a base layer of logical syntax called ECLIF (Extended Common Logic Interchange Format) (Highfleet Inc, 2010). ECLIF belongs to a family of logical languages which are based on an ISO standard logic-based languages framework called Common Logic (ISO/IEC 24707: 2007). The selection of common logic

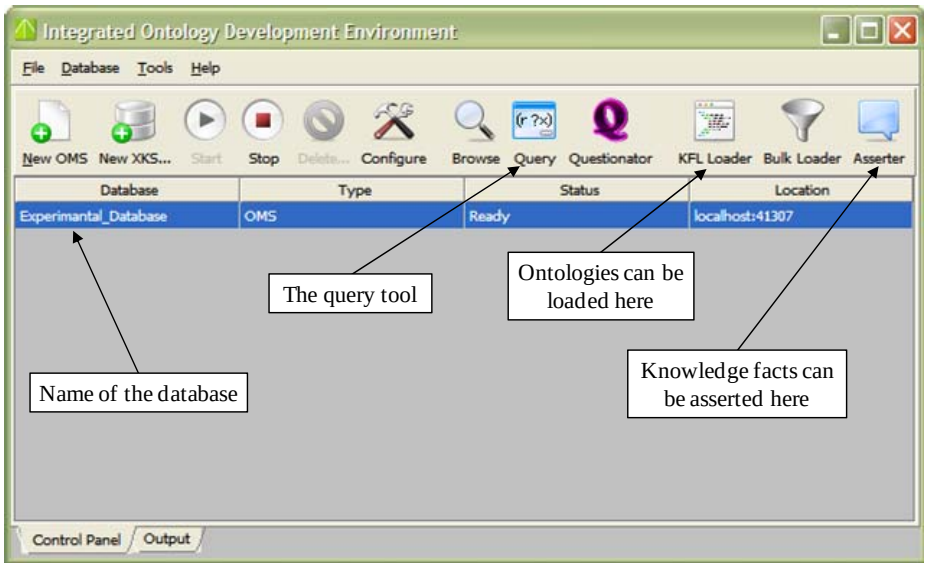

Fig. 2. Ontology editing tool IODE

instead of OWL which is the most used formalism for building and experimenting with

ontologies is due to its high expressiveness. As compared to other formalisms which are usually restricted to the creation of binary relations, CL provides the user the capability to define ternary (three places), quaternary (four places) and even quinary (five places) relations as well (Highfleet Inc, 2010). In addition to that CL also provides a highly powerful syntax for logical expressions.

The ontology editing tool used for this work is the Integrated Ontology Development Environment (IODE™) from Highfleet systems. IODE takes as input the ontologies written in the KFL syntax with prescribed file extensions. Once the ontologies are loaded the knowledge associated to them can be added, deleted and queried. Figure 2 gives a snapshot of the user interface of this tool with not all tool bar buttons shown. A detailed account of the capabilities of this tool is out of the scope of this paper. A brief explanation is, however, needed here to provide a background for the discussion in this paper and therefore this explanation follows in the next section.

4.2. KFL Ontologies

A KFL ontology typically consists of four distinct components:

a. Classes known as properties,
b. Relationships that exist between these classes,
c. Functions for defining a dimension that completely defines a fact such as a measuring unit,
d. Rules which may be used to write axioms, conditional relationships, constraints and complex facts.

A combination of these four components forms a typical KFL ontology. The inherent nature of an ontology i.e. it's hierarchical structure makes it easier for the computer systems to interpret correctly the concepts contained by it. In a KFL ontology, the essential directives used to define the above mentioned components can additionally be used by the mediation and mapping system to distinguish concepts and their attributes. For example, a class or a concept has to be defined as follows:

```
:Prop concept_1
:Inst Type
:sup Object
```

The `'sup'` directive here defines the property `'concept_1'` as a direct subsumption of the concept '`Object`'. By using this directive, through the query tool in IODE, a specific concept can be traced back through its lineage to its origin. This is particularly useful when a concept in the domain ontology has to be traced back to its origin in the foundation ontology. This attribute of an ontology is called here the '*handle*'. So for the concepts in a KFL ontology the handle is its subsumption directive. Similar handles may exist in other parts of an ontology. For example, in a KFL ontology, a relation is defined as follows:

```
:Rel relation_1
:Inst BinaryRel
:Sig concept_1 concept_2
```

Here the arguments in the '`Sig`' directive are used as the handles. These three lines say that there exists a relation named '`relation_1`' which is a binary relation i.e. existing between two concepts, and these two concepts are '`concept_1`' and '`concept_2`'. So the

signature of this relation is used as a handle to distinguish it from other relations with similar names or to match it with an identical relation with a different name in other independently developed ontologies. A similar example can be taken for 'Functions' in a KFL ontology. The way they are defined is as follows:

```
:Fun length_unit
:Inst UnaryFun
:Sig RealNumber -> length_quality
```

Here again, the signature can be used as a handle to differentiate this function named `'length_unit'` from other functions of similar names or to find a similar function with a different name in other ontologies. These three lines say that there exists a function named `'length_unit'` which is a unary function and the value contained by it is a real number and it belongs to the concept `'length quality'`. As far as the rules in the KFL ontologies are concerned, they are constructed by using concepts, relations and functions and therefore if these three components of an ontology are clearly identified and distinguished, the rules will not need any differentiation and deciphering.

Table 3
Experimental ontologies outline

Ontology	Classes	Relations	Functions	Rules
Manufacturing foundation ontology of core concepts Context: MFO	processes resources man machine material part shape_feature hole rim web cob measures width length diameter ……	:Rel has_diameter :Inst BinaryRel :Sig part length	:Fun length_mm :Inst UnaryFun :Sig RealNumber -> length	
Design domain ontology Context: DDO	disc edge bolt_hole diaphragm hub measurements dmtr	:Rel has_dmtr :Inst BinaryRel :Sig bolt_hole dmtr	:Fun length :Inst UnaryFun :Sig RealNumber -> dmtr	
Production domain ontology Context: PDO	comp_disc disc_end straight_hole webbing centre attribute dia	:Rel has_dia :Inst BinaryRel :Sig straight_hole dia	:Fun mm :Inst UnaryFun :Sig RealNumber -> dia	IC hard "dia should be greater than 20mm"

4.3. Knowledge Base Associated to a KFL Ontology

KFL ontologies are populated with instances to develop a knowledge base. This is done by writing a knowledge statement and using it to form or populate the knowledge base. This knowledge statement is called a ‘fact’ and the processes of writing and introducing it to the knowledge bases is called ‘assertion’. So facts are asserted in a KFL ontology in order to develop a knowledge base. Similar facts are also used when a change is made in an already existing knowledge statement. So if the hole size of a modelled component is to be altered, a fact is to be written and asserted. These facts can be controlled through rules existing in an ontology. So in a way the rule base of a KFL ontology decides which facts are allowed to be asserted and which are not.

4.4. Experimental Ontologies

Three very simple ontologies were developed for testing the proposed framework. A foundation ontology with a set of core manufacturing concepts. These concepts mostly consist of generic machining features, providing a foundation and building blocks to model engineering components in design and manufacturing domain ontologies. Industrial experience shows that for a similar concept different perceptions and terminologies exist in the design and production sides of a manufacturing facility. For the sake of simplicity the naming convention differences of only those features which can directly be produced through simple machining are considered here. Figure 3 shows the cross-section of an aero engine compressor disc. Different shades in the figure help to identify distinct features. The rest of the ontology building and experimentation, in this paper, uses this disc and its features as a reference. Table 3 shows the outline of the three experimental ontologies developed for this work. Figure 4 illustrates the three experimental ontologies and their connections. Some clear lexical differences can be seen in the two ontologies. For example the diameter of the disc is termed as ‘dia’ in design ontology while it’s called ‘dmtr’ in the production side. Apart from the lexical differences some conceptual differences may also occur where a combination of some design features makes one manufacturing feature. For example the combination of ‘cob’, ‘diaphragm’ and ‘rim’ of design can be just the ‘turning feature’ in production interpretation. These are just a few examples and things may go more complex when the

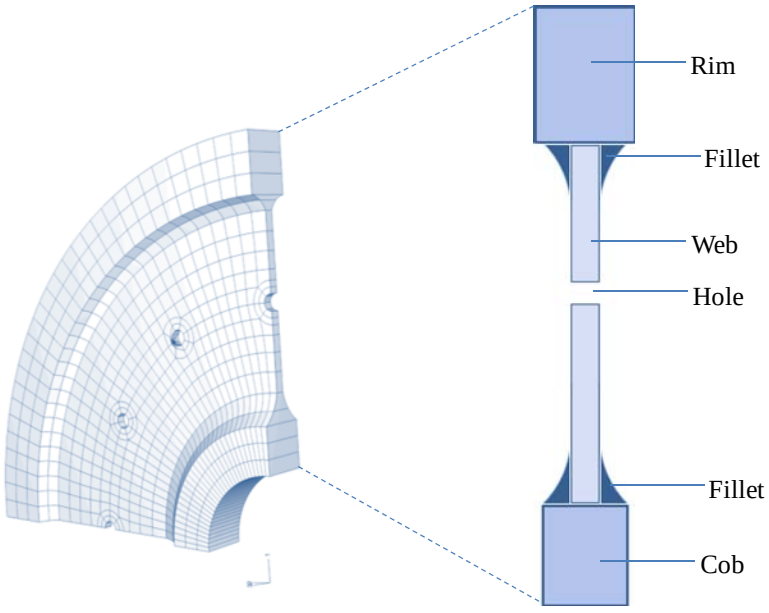

Fig. 3. Features in an aero engine compressor disc

manufacturability knowledge is associated with concepts at the instance level.

The domain ontologies are developed in such a way that they model the aero engine compressor disc shown in figure 3. One thing which differentiates a specific ontology from others is its context. The ontology column in table 1 shows the names of the ontologies and their contexts. So the contexts for the foundation, design and production ontologies are MFO, DDO and PDO respectively. When querying a fact in an ontology, these contexts need to be used as a prefix in order for the query interpreter to direct the query precisely.

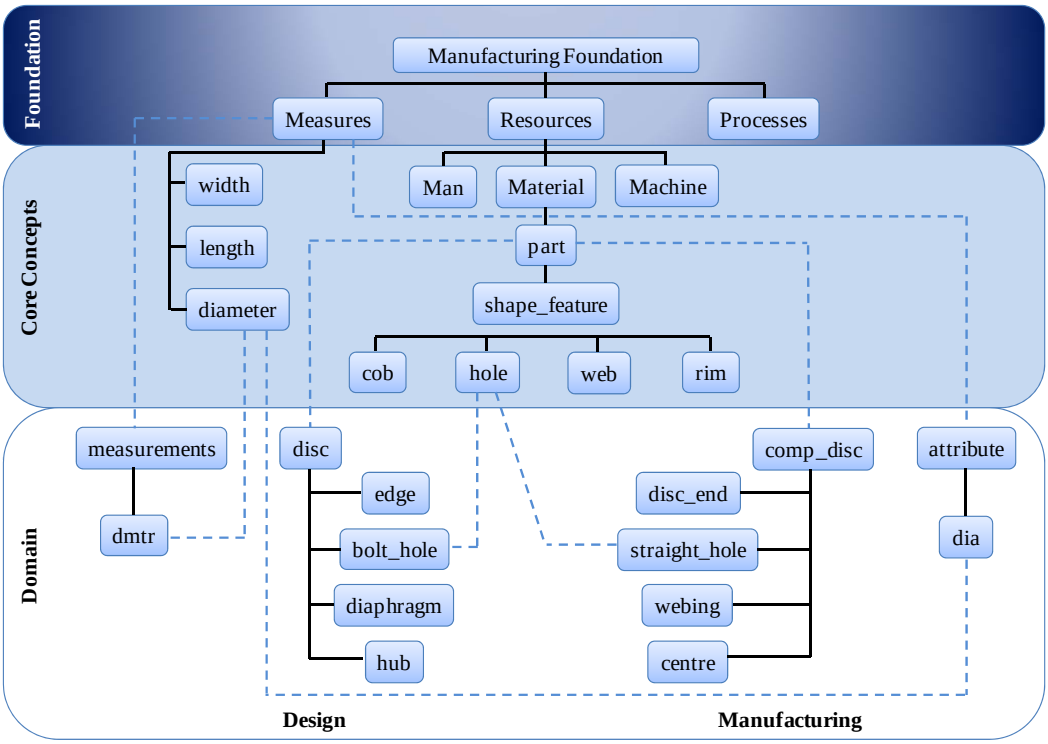

Fig. 4. Core concept and domain ontology connections

4.5. The Framework

Figure 5 illustrates the framework on which the similarity detection methodology of the Verifier is based. The software module developed to work on this framework generates and handles queries posed to IODE, which is the ontology editing environment being used in this research. The top part of the figure shows design and production domain ontologies developed out of the foundation. The lower part of the figure illustrates the functioning of the verifier. The verifier consists of three main modules. Two inheritance identifiers, one each for the source and target ontology and a concept matcher and query builder. From this point forward the terms module A, module B and module C are going to be used for the ‘source ontology inheritance identifier’, ‘concept matcher and query builder’ and the ‘target ontology inheritance identifier’ respectively, as shown in figure 5. A simplified version of the aero engine compressor disc shown in figure 3 will now be used to explain the methodology. It is assumed that this disc is independently modelled in both the design and production domain ontologies by using the feature concepts from the core concepts ontology as shown in figure 4. The acronym VMO shown in figure 5 stands for Verification Meta Ontology and it will be explained in the discussion section.

Although the disc modelled in the design and production ontologies uses totally different terms for its features, these user defined features subsume similar concepts in the core concept ontology, as indicated by dotted lines in figure 4. It is these subsumptions which are used by the verifier to match the concepts on two sides. In the scenario considered here, the manufacturability verification is needed when a change is made in a part modelled in the design ontology. For a manufacturable design, every change made in an approved design has

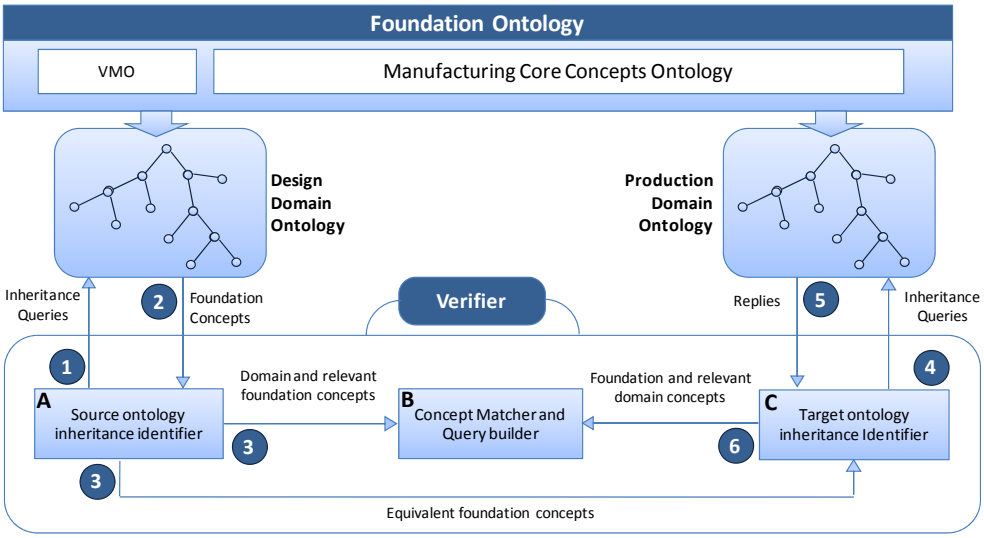

Fig. 5. Similarity Detection Methodology

to be validated by the production engineer. This can be achieved by taking into consideration the manufacturing constraints existing in the production ontology. But since, these two ontologies have been developed independently, the terminologies used to model similar components and their features may differ in the design and production sides. It is therefore necessary that changes in the model in the design ontology are also written in the production ontology language for them to be tested by the manufacturability rules existing in the production ontology and this is done as explained below. The numbers of these points correspond to the circled numbers shown in figure 5.

1- The process initiates when a fact is asserted in the design ontology in order to change a certain characteristic of the modelled component. This fact needs to be rewritten in the form of terms used in the production ontology in order to scrutinize the change by the rules existing in the production ontology. To do this, module A first identifies all the concepts used to write the fact and then generates a query to find their parent concepts in the foundation ontology.
2- Module A receives replies in the form of relevant foundation conceptualisations.
3- The replies to these queries, containing the design domain and relevant foundation concepts, are then sent to module B and module C.
4- Module C then generates a query to explore those concepts in the production ontology which possess the foundation concept inheritance identical to those found in step 3.
5- Module C then receives replies containing domain concepts having the same foundation concepts as sent by module A.
6- These domain concepts along with their related foundation concepts are then sent to module B. The foundation concepts from module C are now compared with those received from module A in step 3 and domain concepts with similar foundation

inheritance are declared as similar. At this point the fact is rewritten in the production ontology terminologies and is asserted in the production knowledge base. Now since the fact written is comprehensible by the rules written in production knowledge base these rules let the verifier and thus the designer know if that particular change is feasible or not.

4.6. The Verifier Application

Figure 6 shows a snapshot of the application interface developed to automatically execute the process explained in the previous section. This automation is achieved by making the system capable of connecting to the IODE database, writing queries using the concepts existing in the ontology, sending them up to the relevant database, interpreting the replies and asserting the facts in the knowledge base. These tasks are executed by controlling the query and fact assertion tools of IODE through the Java API. The term database is used here to refer to the collection of ontologies and associated knowledge bases existing in the IODE server. Through this application, the addition of a hole in the disc template is tested. This disc is shown in figure 3. In this scenario the designer wants to introduce a hole with diameter less than 20mm. This is done by asserting a fact which adds a new hole with name 'hole1'. This fact is written as:

```
(PDO.bolt_hole  hole1)

(PDO.hole1  has_dmtr  (length 15))
```

This statement essentially says,

"A bolt hole is added in the disc modelled in the design knowledge base, it is given the name of '`hole1`', and the `length` value assigned to its diameter (i.e. `dmtr`) is `15`".

To keep things simple at this stage, the hole location coordinates and tolerances related to the hole geometry and positioning are not included here. In the fact shown above, the prefix PDO

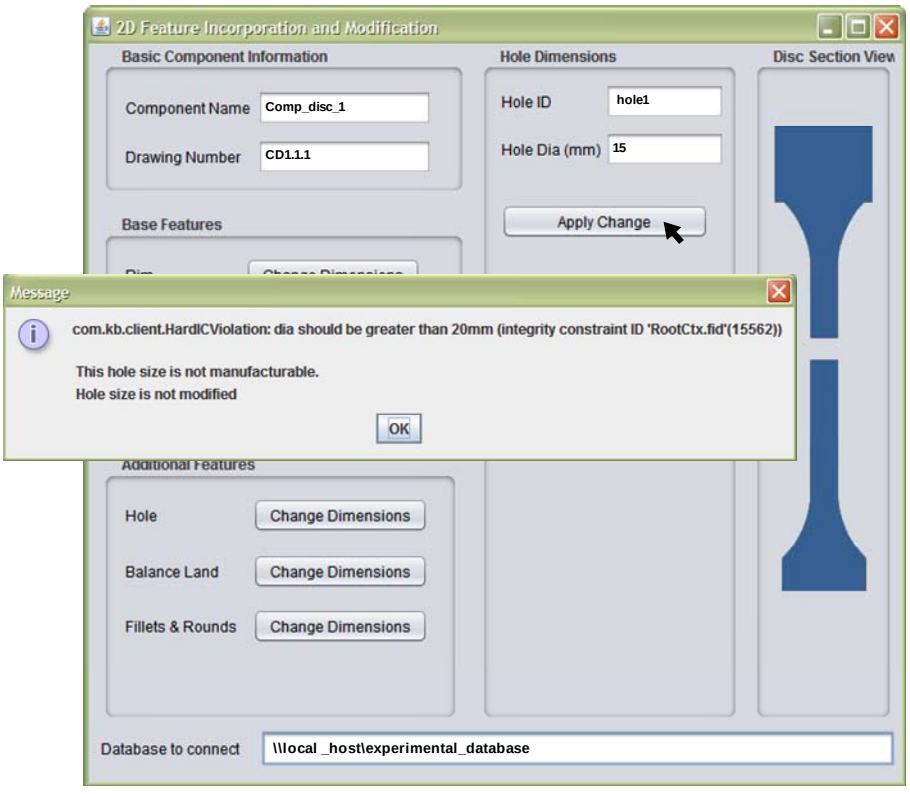

Fig. 6. Verifier user interface

shows the context of a concept and indicates here that the concept '`bolt_hole`' belongs to the design ontology and therefore the fact is aimed at the design knowledge base. The verifier application makes sure that all manufacturability checks are performed before any modification in the exiting disc template is made. This is done by checking the manufacturability rules existing in the production knowledge base. This is however only possible when a fact is asserted in the production knowledge base in its own terminologies. This is necessary for the manufacturability rules to function since they are not applicable on design ontology terminologies as can be seen from the rule statement:

```
1    (=>
2         (and
3              (straight_hole ?h)
4              (= ?x (mm ?kx))
5              (has_dia ?h ?x)
6         )
7              (gteNum ?kx 20)
8    )
9    :IC soft "dia should be greater than 20mm"
```

The keywords above with a question mark (?) are variable names and therefore they can be replaced with any name suitable. `gteNum` in line 7 states that the value assigned to the function `mm` in line 3 has to be greater than 20. The rest of the keyword all belong to the production ontology as can be seen in table 1. Now to make these rules applicable the verifier looks for the counterpart terminologies of design ontology in the production ontology. This is done by scanning all the concepts in the production ontology for their inheritance in the core concepts ontology as explained in the previous section. This process gives the verifier the following output:

`PDO.bolt_hole`	is equivalent to	`PMO.straight_hole`
`PDO.dmtr`	is equivalent to	`PMO.dia`
`PDO.has_dmtr`	is equivalent to	`PMO.has_dia`
`PDO.length`	is equivalent to	`PMO.mm`

Having obtained these results the verifier now rewrites the facts as follows:

```
(PMO.straight_hole     hole1)

(PMO.hole1    has_dia    (mm 15))
```

It can now be seen that all the terminologies used in writing this fact are from the production ontology and can be seen in the rule statement. This makes the production ontology manufacturability rule functional and it scrutinizes this fact for any integrity constraint violation. Since the rule says that "diameter should be greater than 20mm" an integrity constraint is fired and the fact is not asserted. The verifier, as a result, generates an error and the user interface at the designer's end prompts that this hole is not manufacturable by using the existing facility and therefore the template cannot be modified, as shown in figure 6. The message in the error box says that asserting a fact stating that a hole with a diameter 15mm should be added in the disc is an integrity constraint violation.

5. Discussion

Traditional mapping approaches use several algorithms to find the similarity between concepts in two different ontologies. These techniques do not take into consideration cases where common concepts from a foundation ontology are used to build domain ontologies. The technique presented here utilizes the benefits of a foundation and core concept ontology and proves that, during the ontology building process, if the domain concepts are enriched with enough traceability to their origin in the foundation ontology, the techniques of ontology mediation and knowledge verification can be made totally automatic and very accurate.

A closer look at this technique reveals that the back bone of the whole process is the connection of domain ontology concepts with their parents or origin in the core concept ontology. For this experiment, the parent-child relationship or subsumption is utilized. But to make these connections richer and more comprehensive, several different kinds of relationships can be used. For example, a relationship between the domain concepts and core concepts may exist which tells the mediation system that these two concepts are actually similar or that a certain concept is a specialization of a concept in the core concept ontology. If these options are provided in the core concepts and the verification system is designed accordingly the same accuracy and automation can be obtained while making the application of the system broader.

The capabilities and benefits of the verification system proposed here are evident from the explanation in the previous sections but this methodology is not without its weaknesses. The first and foremost is the lack of freedom for the domain ontology builders to use the concepts as they want. This is because certain restrictions, in the use of core concepts, have to be put into place in order for the verification system to interpret the concepts correctly. These restrictions may stifle the creativity of the users and may also jeopardize the correct definition of an object for which not enough concepts are available in the core concepts ontology. These problems can be tackled by making the core concepts ontology as thorough and exhaustive as possible. Secondly, a continuous maintenance and updating of this ontology is needed so that it always meets the contemporary requirements of the domain it belongs to.

When it comes to the restrictions on the use of a certain concept from the core concept ontology the rules in a KFL ontology come in handy. Similar options exist in other formalisms for ontology building, but the important issue is the way these rules are used. In the work presented here, these rules are used to make sure that the concepts from the core are used in a way which is suitable for the mediation system to function. To make things simpler and more manageable, a rule base in the form of a plug-in ontology can be developed which is in accordance with the needs of the mediation and verification system. We call this ontology the Verification Meta Ontology (VMO). This means a core concepts ontology is built separately without worrying about the way concepts are going to be used by the domain ontology builders and to meet the requirements of the verification system, a separate optionally attachable ontology is developed in the form of VMO. This optional attachability of the VMO may also relieve the domain ontology builders from unnecessary restrictions on the use of core concepts to model objects. This is because certain levels of verifiability can be introduced in the VMO for the domain ontology builders to choose from. So the more they want their ontology verifiable and the knowledge contained by it shareable, the less freedom they are granted in using the concepts in their own way and vice versa. This way, a balance between the flexibility and verifiability of the domain models would be achievable depending upon the personal preference of an ontology builder.

The VMO has to be built by the ontology mediation and knowledge verification system builders in accordance with the techniques this verification system is going to use to find similarities. It may contain a few classes but mainly a set of rules which govern the use of concepts from the core concept and foundation ontology. In this way when a VMO is loaded with the foundation and core concept ontology in an ontology editor, it cautions the domain ontology and knowledge base builders on the incorrect and untraceable use of core concepts. A detailed description of this verification meta ontology, however, in itself requires a complete paper and will therefore be covered elsewhere.

Finally, the scenario selected here is based on a case study conducted in an aero engine manufacturing company. To keep things simple, a very generic example of a hole is taken. The case study experience shows that the same feature in a component may have different titles and similar titles may get interpreted differently in design and manufacturing sides of the same factory. It is therefore absolutely essential that when knowledge between independently developed sources is shared by using computers, a mediation system verifies the interpretation across different domains. The example of design and production domains is taken here, but a similar system can be developed to mediate between design and all other domains concerning the whole product lifecycle of the designed engineering component. These domains may include assembly, maintenance, use and disposal as well as production.

6. Conclusion

It is shown in this paper that the mediation and matching of domain ontologies built by utilizing the building blocks provided by a foundation ontology of core concepts can be made automatic if domain ontology builders use these building blocks in a standard way. A Java based application working in conjunction with the ontology editing environment IODE demonstrates this by mediating and matching concepts in the design and production domain ontologies built out of a common foundation. Further research is needed to discover ways in which domain ontology builders are bound to follow the prescribed method of foundation ontology core concepts without being deprived of the freedom to create and innovate their models as they require. An optionally attachable verification meta ontology in this scenario shows promise but requires further work to reach an optimal level. Further development of this work may include the inclusion of the other lifecycle stages of engineering products to ensure an optimal design.

7. Acknowledgements

This research is funded by the Loughborough Innovative Manufacturing and Construction Research Centre (IMCRC) and is being conducted in collaboration with Rolls-Royce plc, UK.

8. References

ANJUM, N., A., HARDING, J., A., YOUNG, B. and CASE, K., 2010. Gap Analysis of Ontology Mapping Tools and Techniques, K. POPPELWELL, J. HARDING, R. POLER and R. CHALMETA, eds. In: *Enterprise Interoperability IV*, 14th and 15th April, 2010, Springer pp303.

ALEKSOVSKI, Z., KLEIN, M.C.A., TEN KATE, W., AND HARMELEN, F. VAN, 2006, “Matching Unstructured Vocabularies Using a Background Ontology”, Proc. Int. Conf. Knowledge Eng. and Knowledge Management (EKAW ’06)

ALEKSOVSKI, Z., TEN KATE, W., AND HARMELEN, F. VAN, 2006, “Exploiting the Structure of Background Knowledge Used in Ontology Matching”, in Shvaiko, P., Euzenat, J., Noy, N., Stuckenschmidt, H., Benjamins, R., and Uschold, M. eds, 2006, “Proc. Int’l Workshop Ontology Matching (OM-2006)”

CHANGOORA, N., 2010, A framework to support semantic interoperability in product design and manufacture. PhD. Thesis, Loughborough University, Loughborough, UK.

DENG, J., DONG, W., SOCHER, R., LI, L.-., LI, K. and FEI-FEI, L., 2009. ImageNet: A Large-Scale Hierarchical Image Database, *CVPR09*, 2009, .

DOAN, ,ANHAI, MADHAVAN, ,JAYANT, DOMINGOS, ,PEDRO and HALEVY, ,ALON, 2002. Learning to map between ontologies on the semantic web, *WWW '02: Proceedings of the 11th international conference on World Wide Web*, 2002, ACM pp662-673.

EHRIG, M. and STAAB, S., 2004. QOM – Quick Ontology Mapping. pp. 683-697.

GOMEZ-PEREZ, A., FERNANDEZ-LOPEZ, M. and CORCHO, O., 2004. Ontological Engineering: with Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web. Springer-Verlag.

GRUBER, ,THOMAS R., 1993. A translation approach to portable ontology specifications. *Knowl.Acquis.*, 5(2), 199-220.

HIGHFLEET INC., 2010. KFL Reference. Highfleet Inc.

ISO/IEC 24707, 2007. Information technology — Common Logic (CL): a framework for a family of logic based languages. 1st edn. ISO/IEC 24707:2007(E).

KALFOGLOU, ,YANNIS and SCHORLEMMER, ,MARCO, 2003. Ontology mapping: the state of the art. Knowl.Eng.Rev., 18(1), pp. 1-31.

LI, J., 2004, “LOM: A Lexicon-Based Ontology Mapping Tool”; Proc. of the Workshop on Performance Metrics for Intelligent Systems PerMIS ’04

LOURDUSAMY, R. and GANAPATHY, G., 2008. Feature Analysis of Ontology Mediation Tools. Journal of Computer Science, 4(6), pp. 437-446.

MAEDCHE, ,ALEXANDER, MOTIK, ,BORIS, SILVA, ,NUNO and VOLZ, ,RAPHAEL, 2002. MAFRA - A MApping FRAmework for Distributed Ontologies, *EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, 2002, Springer-Verlag pp235-250.

MASCARDI, V., ROSSO, P., AND CORDI, V., 2007, “Enhancing Communication inside Multi-Agent Systems—An Approach Based on Alignment via Upper Ontologies”, Proc. Int’l Workshop Agents, Web-Services and Ontologies: Integrated Methodologies

MASCARDI, V., LOCORO, A., AND ROSSO, P., 2010, “Automatic Ontology Matching via Upper Ontologies: A Systematic Evaluation"; IEEE Transactions on Knowledge and Data Engineering

MATUSZEK, C., CABRAL, J., WITBROCK, M. and DEOLIVEIRA, J., 2006. An Introduction to the Syntax and Content of Cyc. *AAAI Spring Symposium*, .

MCGUINNESS, D., FIKES, R., RICE, J. and WILDER, S., 2000. An Environment for Merging and Testing Large Ontologies, *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000)*, 2000, .

MITRA, P. and WIEDERHOLD, G., 2002. Resolving Terminological Heterogeneity In Ontologies.

NECHES, ,ROBERT, FIKES, ,RICHARD, FININ, ,TIM, GRUBER, ,TOM, PATIL, ,RAMESH, SENATOR, ,TED and SWARTOUT, ,WILLIAM R., 1991. Enabling technology for knowledge sharing. *AI Mag.*, 12(3), 36-56.

NILES, ,IAN and PEASE, ,ADAM, 2001. Towards a standard upper ontology, *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems*, 2001, ACM pp2-9.

NOY, N.F. and MUSEN, M.A., 2003. The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping. *International Journal of Human-Computer Studies*, 59, 2003.

SCHORLEMMER, M. and KALFOGLOU, Y., 2005. Progressive ontology alignment for meaning coordination: an information-theoretic foundation, *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 2005, ACM pp737-744.

STUMME, G. and MAEDCHE, A., 2001b. FCA-Merge: Bottom-Up Merging of Ontologies, *Proceedings of the 7th International Conference on Artificial Intelligence (IJCAI'01)*, 2001b, pp225-230.

SWARTOUT, B., RAMESH, P., KNIGHT, K. and RUSS, T., 1997. Towards Distributed Use of Large Scale Ontologies, A. FARQUHAR, M. GRUNIGER, A. GOMEZ-PEREZ, M. USHOLD and P. VAN-DER-VET, eds. In: *AAAI'97 Spring Symposium on Ontological Engineering*, 1997, pp138-148.

VISSER, P.R.S., JONES, D.M., BENCH-CAPON, T.J.M. and SHAVE, M.J.R., 1997. An analysis of ontological mismatches: Heterogeneity versus interoperability, *AAAI 1997 Spring Symposium on Ontological Engineering*, 1997, .

YANG, Q.Z. and ZHANG, Y., 2007. Semantic Interoperatbility to Support Collaborative Product Development. In: W.D. LI, S.K. ONG, A.Y.C. NEE and C. MCMOHAN, eds, Collaborative Product Design and Manufacturing Methodologies and Applications. Springer, .