This item was submitted to Loughborough's Institutional Repository (https://dspace.lboro.ac.uk/) by the author and is made available under the following Creative Commons Licence conditions.

creative commons
COMMONS DEED

Attribution-NonCommercial-NoDerivs 2.5

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Noncommercial. You may not use this work for commercial purposes.

No Derivative Works. You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/

The configuration of design and manufacture knowledge models from a heavyweight ontological foundation

N. Chungoora[a]*, R.I.M. Young[a]

[a] *Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, City, Country*

Nitishal Chungoora

Wolfson School of Mechanical and Manufacturing Engineering,
Loughborough University,
Loughborough,
LE11 3TU, UK
Tel: +44(0)7871226566
e-mail: N.Chungoora@lboro.ac.uk

Robert Ian Marr Young

Wolfson School of Mechanical and Manufacturing Engineering,
Loughborough University,
Loughborough,
LE11 3TU, UK
Tel: +44(0)1509227662
Fax: +44(0)1509227648
e-mail: R.I.Young@lboro.ac.uk

* Corresponding author. e-mail: N.Chungoora@lboro.ac.uk

Problems related to knowledge sharing in design and manufacture, for supporting automated decision-making procedures, are associated with the inability to communicate the full meaning of concepts and their intent within and across system boundaries. To remedy these issues, it is important that the explicit structuring of semantics, i.e. meaning in computation form, is first performed and that these semantics become sharable across systems. This paper proposes a Common Logic-based ontological foundation as a basis for capturing the meaning of core feature-oriented design and manufacture concepts. This foundation serves as a semantic ground over which design and manufacture knowledge models can be configured in an integrity-driven way. The implications involved in the specification of the ontological foundation are discussed alongside the types of mechanisms that allow knowledge models to be configured. A test case scenario is then analysed in order to further support and verify the researched approach.

Keywords: design and manufacture; heavyweight ontology, common logic, semantics, process specification language, entity information

1 Introduction

The rationale behind ensuring the seamless exchange of manufacturing knowledge within and across enterprise boundaries, is dominated by the need to speed up the production of goods and services at lower cost, while ensuring higher levels of quality and customisation (Mertins et al., 2008). Specifically in Product Lifecycle Management (PLM), knowledge which is shared for collaborative product development not only resides and cuts across various product lifecycle phases, but also involves groups that may jointly function within institutional boundaries as well as across multiple organisations (Hameed et al., 2004).

Therefore, in modern PLM, design and manufacturing knowledge handled by decision support systems has to be efficiently communicated across the entire lifecycle. Interoperable knowledge, for instance, is paramount to the integration of mechanical analysis into the design process, one of the most obvious and crucial requirements, particularly during the early stages of design (Aifaoui et al., 2006). However, seamless interoperability to effectively support collaborative product development, is still not completely achievable. This lack of seamless exchange capability is costly to many globally distributed industries since significant amounts of money are spent into overcoming the related problems (Research Triangle Institute, 1999; Brunnermeier and Martin, 2002).

There exist two obvious yet problematic solutions to realising interoperable knowledge sharing. The first is linked to the adoption of an all-embracing common rigid model across systems. This approach to interoperability is, however, immensely problematic and remains a very unlikely scenario (Hameed et al., 2004), as the level of flexibility required by multiple systems would be greatly affected. The other possibility involves allowing different systems to develop and use their preferred methods, and to later worry about information exchanges. This approach provides multiple systems with their desired level of flexibility. Unfortunately, the translation mechanisms that would be needed for allowing inter-system interpretation and sharing of semantics would demand considerable effort and may not provide optimal solutions.

Another possible way to tackle this issue, which is explored in this paper, is to adopt a direction where the meaning, in computational form, associated to core feature-based design and manufacturing concepts is established. These foundation semantics use formal heavyweight ontological structures (Gómez-Pérez et al., 2004; Young et al., 2007) for defining the meaning behind entity information, such as standard features, and machining process execution knowledge. It is to be pointed out that although there has been a range of work that has focused on foundation ontologies (Pease and Niles, 2002; Masolo et al., 2003; Borgo and Leitão, 2008) the scope of these foundation ontologies has remained relatively broad. This suggests the key ongoing concern of how effective foundation ontology approaches can be tailored to support the communication requirements of manufacturing (Young et al., 2007).

The heavyweight ontological foundation explored in this work targets a different level of granularity compared to existing foundation ontologies. Such a foundation supports the argument for a common set of core feature-oriented semantics that can be reused and extended via rigorous ontological mechanisms, by multiple domains. The approach identified in this paper has been experimented in a test case scenario whose scope is based on the configuration of a knowledge model of hole features in manufacturing.

2 Semantics in design and manufacture

Figure 1 opens the issues arising in the quest for semantic interoperability, based on a design and manufacture information organisation perspective. For any given product family whose evolution follows the epicycles in product lifecycle development (Subrahmanian et al., 2005), several views of the same artifact are bound to exist when considered from the different nodes residing in the product lifecycle such as conceptual design, detailed design, manufacturing, operation, etc. In Figure 1, these multiple perspectives include "Geometric Dimensioning and Tolerancing", "Function", "Process Planning and Execution", "Machining Resource" and may consist of other views as well. Multiple perspectives of the same artifact result in multi-viewpoint models (Kugathasan and McMahon, 2001; Gunendran and Young, 2006). Multi-viewpoint models of a type of product naturally overlap with each other since they pertain to the same artifact.

[Figure 1 to be inserted about here]

The example in Figure 1 suggests that product features in design and manufacture are often defined based on view-specific semantic information structures.

The capture and representation of similar structures are necessary in order to support multi-viewpoint information sharing (Kugathasan and McMahon, 2001; Canciglieri and Young, 2003; Gunendran and Young, 2006). Furthermore, to capture the interactions between elements from different view-specific semantics, relationships need to be made across views so that the knowledge contained in one viewpoint can be interpreted in another without any loss of meaning. For example, based on the nominal dimensions and tolerances carried by the features shown in Figure 1, it could be possible to identify potential machining processes, in order to establish the inputs and outputs between the feature entities and relevant machining process execution sequences.

On the other hand, the ability to harness the appropriate semantic technologies, in order to facilitate the explicit capture of domain semantics in computational form (formalisation) and to support shared meaning across knowledge models (i.e. formal models of domains), constitutes another key aspect. Several families of knowledge representation formalisms have been developed to capture and represent semantic structures. Such formalisms include among others Frame-based languages (Wang et al., 2006), Description Logic-based languages (Baader et al., 2007) and Common Logic (ISO/IEC 24707, 2007) altogether forming a repertoire of languages with different levels of expressiveness as far as the representation of semantics is concerned (Ray, 2004). It has been acknowledged that there is an ongoing need for more mathematically rigorous approaches to ensure that the true meaning of terminology coming from different systems is identical, to permit computational comparisons of the meaning of terms (Young et al., 2007; Das et al., 2007). For this reason, this works builds on top of this understanding through the exploration of a heavyweight Common Logic ontology-based approach.

3 Ontology-based approach

The ontology-based approach applied to the specification of a heavyweight ontological foundation, while supporting the configuration of multiple design and manufacture knowledge models, is illustrated in Figure 2. The diagram firstly identifies a “Foundation Layer” which provides the capability to address the semantic structures needed for capturing the meaning of core concepts. The provision of adequate ontology-based mechanisms then allow multiple knowledge models to be constructed in the “Domain Ontology Layer”. The implications of each layer are discussed next.

[Figure 2 to be inserted about here]

3.1 Foundation layer

3.1.1 Knowledge Framework Language (KFL)

The “Foundation Layer” comprises two characteristic elements, namely a rigorous Common Logic-based ontological formalism over which a heavyweight manufacturing ontological foundation is constructed. Figure 3 provides a more detailed view of the “Foundation Layer”. From the diagram, it can be seen that the rigorous Knowledge Framework Language (KFL), a Common Logic-based formalism developed by Ontology Works Inc. (2009), imparts the syntax and first-order semantics, governing the way in which the heavyweight manufacturing ontological foundation is formalised.

[Figure 3 to be inserted about here]

3.1.2 Heavyweight manufacturing ontological foundation

The heavyweight manufacturing ontological foundation captures and expressively represents generic entity information and process semantics, together with some of the overlapping relationships that hold between entities and processes (refer to Figure 3). These semantics are applicable to a spectrum of viewpoints and domains in product design and manufacture.

The accommodation of process semantics in the "Foundation Layer" involves the formalisation of relevant concepts from the Process Specification Language ontology (PSL) (ISO 18629, 2005). Since it has been shown that PSL provides intuitions for reasoning about various forms of processes (Chen et al., 2003; Bock and Gruninger, 2005; Bock, 2006; Das et al., 2007), this implies that PSL provides a suitable choice for the capture of generic process semantics. As a result of the current limitations of PSL to relate to resource definitions and to products inputs and outputs (Young et al., 2007), the "Object" concept from PSL has been expanded to include a broader understanding of entity information semantics. It is to be noted that PSL is currently available as an ontology written in the Common Logic Interchange Format (CLIF) (Process Specification Language Website, 2009), thereby implying its straight-forward implementation expressed in KFL.

To capture generic entity information semantics, the fundamentals from the revised Core Product Model (CPM) (Fenves et al., 2004) and those from STEP 10303-224 (ISO 10303 AP224, 2006) are being exploited and adapted to the needs of the "Foundation Layer". This is because the CPM is a generic, abstract model that can be used as a starting point for capturing foundation entity information semantics. Due to the fact that the CPM exists as a conceptual model while favouring extensions in order to make the model readily expandable (Fenves et al., 2005), the latter does not, for example, focus on how specific types of features need to be semantically defined. For this particular reason, concepts from ISO 10303 AP224 have been exploited because of the slant onto wide-ranging feature definitions and also because features support the integration between design and manufacture (Abouel Nasr and Kamrani, 2006; Dartigues et al., 2007; Nassehi et al., 2007).

3.1.3 Formalisation of heavyweight semantics

Since PSL has been well-documented in literature, this section concentrates on the main types of intuitions used in the formalisation of entity information semantics in the “Foundation Layer”. Figure 4 illustrates a high-level IDEF5 (Knowledge Based Systems Inc., 1994) schematic diagram of main classes present in the taxonomy for the “Object” class. The abstract “Core_Entity” class involves the basic semantics of features and artifacts that hold features, while the abstract “Core_Property” class is present to provide more detailed semantics, primarily used towards product feature definitions and their behaviours. Other classes such as “Shape_Aspect” and “Feature” in turn have classification information, which is hidden on Figure 4.

[Figure 4 to be inserted about here]

A foundation intuition, for example, captures the idea that core entities may need to hold some function, which is an essential factor that governs the existence of design entities in the first place. By adding an axiom to capture the constraint that every core entity holds some function, it is possible to enforce an optional necessary condition, which is also carried upwards to the “Domain Ontology Layer”. Figure 5 depicts the KFL statement of this integrity constraint (IC), i.e. axiom, and the related

IDEF5 schematic indicating the binary relation “holds_function” with the classes “Core_Entity” and “Function” as arguments to the relation.

In KFL, there are four degrees of gravity relating to the violation of ICs and are identified as “weak”, “soft”, “hard” and “adamant” (in ascending order of gravity). A weak IC, when infringed, would simply indicate an irregularity which does not necessarily constitute a problem. A soft IC is stronger than a weak IC and does not prevent an instance loading process from taking place. A hard IC completely prevents a wrong action from being committed, while an adamant IC is one which indicates a necessity and is destined to be used for the KFL meta-ontology system. Thus, the logical expression in Figure 5 together with the “:IC soft” line appended to it, capture the intuition that the identification of every instance of “Core_Entity” can be accompanied by the optional statement of some instance of “Function” that is held by the “Core_Entity” instance. In the event that this IC is violated, the required message is flagged to prompt the irregularity to the knowledge engineer.

[Figure 5 to be inserted about here]

The formulation of ICs is a vital part of heavyweight ontological approaches, as they support the formal interpretation of terms and conditions within ontologies, unlike lightweight ontological approaches that assume that the meaning behind the terms is readily understood. To further illustrate the strength of heavyweight semantics, some of the formal intuitions behind the interpretation of the “Round_Hole” class, a sub-class of “Feature”, are captured in Figure 6. The relevant classes and relations that take part in the logical expressions in Figure 6 are shown in the accompanying IDEF5 schematics.

[Figure 6 to be inserted about here]

Two sub-classes of “Shape_Aspect” namely “Circular_Closed_Profile” and “Linear_Path” support the geometry and dimensional semantics of “Round_Hole”. The binary relation “holds_shape” associates the class “Round_Hole” to the required “Shape_Aspect” and the unary relations “through” and “blind” dictate the behaviour of the class “Circular_Closed_Profile” when associated to “Round_Hole”. It is to be noted that all the ICs present in Figure 6 are appended with an “:IC hard” line, which implies that if any of these conditions is violated, a transaction such as the loading of instances to a KB, would not take place until the infringed IC is corrected accordingly. This helps ensure the semantic consistency and enrichment of populated knowledge.

3.2 Domain ontology layer

The “Domain Ontology Layer” is at the second level of the ontology-based approach. Reusable foundation semantics from the “Foundation Layer” can be specialised, i.e. configured, for the development of knowledge models. In the context of this paper, a knowledge model refers to a domain ontology together with its associated KB. These domain ontologies may be regarded as being extensions to the “Foundation Layer” and have their semantic structures not only based on foundation semantics but also based on the formalisation of relevant domain-specific rules, constraints, preferences and terminologies.

3.2.1 Contexts for knowledge models

During the configuration of knowledge models from the “Foundation Layer”, it is possible to envisage the use of terms, that are the same as in the heavyweight ontological manufacturing foundation, to refer to different domain-centric intuitions. Similarly, two separately-developed knowledge models could be employing the same terms to mean different notions. At first sight this would lead to semantic conflicts. However, following the proposed ontology-based approach, knowledge models are built “within contexts”. “Contexts” are very similar to namespaces applied to the Semantic Web. Contexts for knowledge models have two main purposes namely (1) to distinguish between elements and attributes from different vocabularies with different meanings that happen to share the same name (Harold and Means, 2004) and (2) to group all related ontological entities from a single ontology together so that implementation platforms can easily identify them.

3.2.2 Ontological relationships and integrity constraints

Other mechanisms that allow specialisation to take place in the “Domain Ontology Layer” consist of two fundamental ontological relationships. The taxonomy of domain classes can be made homogeneous and logical using the principle of specialisation through subsumption (Rector, 2003). The “super-class-sub-class” ontological relation allows the specialisation of domain classes and taxonomy to take place. The other ontological relationship, which is not a subsumption relation, is the notion of “instance-of”, which makes the population of individuals possible through the instantiation of classes. These two ontological relations are key to the internal structure of any knowledge model, and are thus accounted for in all meta-model ontologies such as the Ontology Works Upper Level Ontology for KFL, the Protégé knowledge model (Noy et al., 2000) and that of Ontolingua (Gruber, 1992).

Figure 7 illustrates how the "super-class-sub-class" subsumption relation is used to specialise the "Round_Hole" and "Function" classes into the domain-specific notions of "Locating_Hole" and "Assembly_Function" respectively. Another heavyweight mechanism that allows the configuration of knowledge models to occur is through the specification of domain-defined ICs. In the previous section, one of the features of ICs as a means to embed foundation ontological axioms as prescriptions to complement semantic knowledge (Mäs et al., 2005), has been exposed. In addition to this, ICs also have a direct influence onto the semantic conformance of knowledge models. Figure 7 hence depicts two legal domain-defined ICs which could be established to provide reasoning over "Locating_Hole" and "Assembly_Function". Such ICs involve the reuse of foundation semantics such as the relation "holds_function" and can exist within knowledge models as long as they conform to and do not violate foundation ICs. The "Foundation" term used in the logic expressions identifies the "context" for the term "holds_function".

[Figure 7 to be inserted about here]

3.2.3 Discrete knowledge representation

In the "Domain Ontology Layer", instantiation is the process of asserting facts (instances) and fact sentences (statements about how instances are related) to capture concrete states of a domain ontology in the KB associated to the ontology. Instantiation is a valuable process for the representation of reusable design and manufacture domain knowledge. The successful population of instances demands the satisfaction of all the related foundation and domain-specific ICs set over their classes. Figure 8 identifies how the population of instances takes place for capturing the knowledge related to a machining process sequence for making a particular "Locating_Hole". The ontological relationship "inst" firstly relates individuals to the corresponding classes that they instantiate. Furthermore, the figure also points to the formulation of fact sentences through the reuse of foundation relations to bind instance arguments together.

"Make_Hole_X" "occurrence_of" "Make_Hole" and "Reaming_Occ" "subactivity_occurrence" "Make_Hole_X" are examples of fact sentences. The classes "Activity" and "Activity_Occurrence" as well as the relations "occurrence_of" and "subactivity_occurrence" are concepts that originate from PSL. It is to be noted that the bold dotted arrow in Figure 8 reflects the linear ordering PSL-based semantics over the subactivity occurrences of "Make_Hole_X". In addition, the relations "input" and "output" are defined foundation relations which can be reused for stating the participation semantics of entity information with manufacturing processes.

[Figure 8 to be inserted about here]

4 Implementation

The implementation of the "Foundation Layer" and the "Domain Ontology Layer" follow the Knowledge Engineering Methodology prescribed by (Noy and McGuinness, 2001). The Integrated Ontology Development Environment (IODE) V2.1.1 developed by Ontology Works Inc. (2009) has been employed as ontology implementation platform, because the latter is capable of handling expressive Common Logic-based semantic structures. In IODE, ontology files firstly undergo a "parsing" stage for identifying warnings and obvious errors such as missing parentheses in logic statements. If the parsing stage is correct, the "loading" phase is initiated where ICs are checked against the system's own ICs for any consistency violations. When the loading phase is completed, the "saving" phase is prompted so that the ontology can be saved in IODE.

4.1 Implementation of the foundation layer

Figure 9 portrays the implemented and saved "Foundation Layer". A majority of the classes present in the taxonomy for the heavyweight manufacturing ontological foundation is shown. The figure highlights the class "Round_Hole" and its informal description, together with the relations for which the class "Feature", and hence "Round_Hole", is an argument to. These relations include, for example, the unary relation "compound" for the statement of complex features that are composed of singleton features, and the binary relation "holds_feature", which is used to associate features to artifacts. Figure 9 also depicts two of the implemented ICs relevant to the class "Round_Hole".

[Figure 9 to be inserted about here]

4.2 Configuration and implementation of a knowledge model

This section documents the configuration and implementation of a test case scenario, which focuses on the specialisation of a "Machining Hole Feature Ontology A" and its KB from the "Foundation Layer". The aim of the test case is to prove the ways in which the "Foundation Layer" facilitates the specialisation of the knowledge model, such that a semantically rich and accurate representation of the ontology and the instances populated in its KB are obtained.

4.2.1 Machining hole feature ontology

The diagram in Figure 10 provides an insight into the type of part family being investigated. In this scenario, a “Housing_Part_Family” is considered and the various concepts developed within “Machining Hole Feature Ontology A” are attuned to the geometry, dimensional and machining process execution viewpoints. It is to be pointed out that Figure 10 does not represent a concrete state of the domain ontology, but in fact reflects some of configured entity information concepts, machining process concepts and informally expressed domain-defined ICs.

[Figure 10 to be inserted in about here]

An example of customised entity information semantics is present in the specification of the “Counterbore_Hole” class. The latter is specified as being a sub-class of “Feature”, of compound property, that aggregates the “Round_Hole” sub-classes “Drilled_Hole” and “Counterbore” as elements of “Counterbore_Hole”. Some of the essential axioms governing the interpretation of “Counterbore_Hole” are listed in Figure 10. In addition to entity information semantics, domain-specific process semantics are also under consideration in the “Machining Hole Feature Ontology A”. One such example appears on Figure 10, where the informal semantics for the class “Reamed_Hole_Making”, a sub-class of the foundation class “Activity”, are identified. PSL-based semantics from the PSL-Core and Outer-Core theories are used to provide the essential formal linear ordering semantics involved in the execution of “Reamed_Hole_Making”. The domain ontology also takes into account the participation semantics between “Reamed_Hole” and the machining process class “Reamed_Hole_Making” which outputs “Reamed_Hole”.

4.2.2 Implementation of the domain ontology

Figure 11 illustrates the procedure for implementing the “Machining Hole Feature Ontology A”. Within the ontology development environment, the saved “Foundation Layer” is first cloned for retaining all foundation semantic structures. The cloned “Foundation Layer” is renamed to “Machining Hole Feature Ontology A” and the corresponding domain ontology file is parsed, loaded and saved to the new Object Management System (OMS). During this process, a few warnings and errors were flagged. These occurred during the parsing phase of the ontology file. Figure 11 shows (1) warnings as a result of confusing variables declared in some domain-defined ICs and (2) an error which occurred due to an incorrect reuse of the foundation unary relation “base”, which forms part of the semantics of features of compound property. These warnings and errors have prompted the necessary rectifications prior to a successful loading and saving of the domain ontology.

[Figure 11 to be inserted about here]

Figure 12 provides a browsed view of the saved domain ontology. The top portion of the diagram illustrates domain-defined process semantics. The taxonomy for the foundation class “Activity” contains the specialised machining process classes. The “Reamed_Hole_Making” class, for example, carries informal information in the form of remarks, but is also rigorously defined via the specification of domain-specific ICs. The bottom part of the diagram then identifies domain-defined entity information concepts. The highlighted “Counterbore_Hole” class also carries informal remarks as well as formal ICs. All the different types of features making up the “Housing_Part_Family” are also shown, alongside the configuration of the foundation class “Length_Measure” used to associate dimensional parameters to these types of features.

[Figure 12 to be inserted about here]

4.2.3 Populating the KB for the domain ontology

Figure 13 identifies part of a concrete state of the domain ontology captured through the specification of facts and fact sentences. The diagram exemplifies, in an informal way, samples of instances defined for representing discrete knowledge. In the figure, “Reamed_Hole_A” carries specific dimensional and tolerance parameters. These parameters also apply to “Reamed_Hole_B”, although the two are different instances as their placements in space vary. The compound feature “Counterbore_Hole_A” aggregates two instances namely “Drilled_Hole_E” and “Counterbore_A”, which in turn carry their own semantics.

Furthermore, specific branches of the occurrence tree are constrained so as capture the relevant machining process sequences. The activity occurrence “occA_Hole_Centre_Drilling” constitutes the initial occurrence in the tree. In Figure 13, it is also possible to view two complex activity occurrences namely “occ_Make_Reamed_Holes_AB” and “occ_Make_Counterbore_Hole_A”. The instances “Reamed_Hole_A” and “Reamed_Hole_B” become outputs of the occurrence “occ_Make_Reamed_Holes_AB” after the latter has been executed. Similarly, the instance “Counterbore_Hole_A” becomes an output of the occurrence “occ_Make_Counterbore_Hole_A”. During the commitment of instances to the KB for “Machining Hole Feature Ontology B”, a complete product representation of the artifact shown in Figure 13 has been captured as well as the related process knowledge.

[Figure 13 to be inserted about here]

The instance file containing the facts and fact sentences used for the scenario in Figure 13 are loaded in the KB of the domain ontology using the "Asserter" tool as shown in Figure 14. In the first attempt to load and save the instance file, two hard IC violations and eight soft IC violations have been reported in the "Assertion Log". The source of the infringements appear at the end of each listed violated IC (not shown in Figure 14 for clarity). As a result of the hard IC violations, for example, "Every counterbore hole involves a drilled hole and a counterbore which are elements of the counterbore hole", the first commitment transaction of instances to the KB is rejected.

[Figure 14 to be inserted about here]

On consulting the instance file, it was discovered, for example, that missing information was present in the specification of the "Counterbore_Hole_A" instance (see Figure 13) where "Drilled_Hole_E" and "Counterbore_A" were not aggregated under the compound feature "Counterbore_Hole_A". Note also from Figure 14 the soft IC violation "Every core entity holds some function", which is present because core entities from the machining viewpoint do not carry semantics about their functions, as this is more relevant to the functional design viewpoint. Five other soft ICs, which involve process-based semantics, have also been reported in the "Assertion Log" such as "If an occurrence of drilling is allowed, then an occurrence of reaming immediately after it may be allowed". The consequence of soft ICs is not detrimental to the integrity of instances being populated under "Machining Hole Feature Ontology A". However, the occurrence of hard ICs in the instance file prevents the commitment transaction from taking place.

Facts with hard IC violations are rectified accordingly, reloaded and checked for IC violations again, and saved in the KB. Figure 15 shows the “Counterbore_Hole_A” instance that has been successfully created and can be browsed from the “Instances” tab for the class “Counterbore_Hole”. The diagram also identifies successfully created instances of the class “Activity_Occurrence”. The complex activity occurrence instance “occ_Make_Reamed_Holes_AB” has been highlighted. Moreover, the subactivity occurrences of the complex occurrence can also be viewed, suggesting the complete and accurate representation of discrete knowledge articulated through ICs.

5 Discussions

The test case scenario has demonstrated how the integrity-driven configuration of a knowledge model can be achieved using foundation semantics structures, as well as the ontological mechanisms that allow knowledge model configuration to take place. It is evident from the approach that the specialisation of knowledge models is flexible enough to address the conceptual preferences of domains, thereby implying that the “Foundation Layer” supports customisable domain extensions, as long as they comply to the fundamental intuitions established in the “Foundation Layer”.

The types of ICs explored in this paper have also shown that the population of instances in knowledge models can be carefully articulated. This provides an ideal direction to enable the meaningful capture and enrichment of manufacturing knowledge for reuse. In the test case, it is seen that although soft ICs do not have an impact on semantic integrity, yet when flagged, these raise the awareness of the knowledge engineer about the possible options available to assert additional semantics if needed. In other words, the action of ensuring that soft IC violations are corrected is not obligatory but may help to introduce additional semantics.

Furthermore, the proposed approach could be extended to support industrial applications. Domain ontologies that derive from the "Foundation Layer" could be interfaced with Computer Aided Engineering (CAE) applications, for example, a Computer Aided Design (CAD) environment could be linked to a domain ontology that represents the semantics in solid modelling. The KB related to the domain ontology would be used as a repository for storing, accessing, updating and creating parts information. In addition, rigorous heavyweight semantics from PSL could be exploited towards monitoring shop-floor activities such as automated machining and assembly sequences. This is another area where the applied importance of ICs would be witnessed. These ICs would ensure that correct and complete information is captured and adequate procedures carried out.

Following the explored ontology-based approach, it is feasible to suggest that if multiple knowledge models were configured from the same set of core foundation semantics, then, they would all share an overlapping definitional basis which would support knowledge model interoperability. This would help facilitate the ability to evaluate correspondences between knowledge models that have been constructed from the "Foundation Layer". However, to be able to achieve such a capability, the identified ontology-based approach would require extensions to accommodate further mechanisms to permit the reconciliation of the content across knowledge models. Figure 16 thus illustrates our current view on a framework to support semantic interoperability in product design and manufacture.

[Figure 16 to be inserted about here]

In addition to the “Foundation Layer” and “Domain Ontology Layer”, two extra dimensions can be depicted namely (1) a “Semantic Reconciliation Layer” which supports the deployment of ontology mapping concepts and reusable semantic alignments in the form of mapping concepts and (2) an “Interoperability Evaluation Layer” which provides necessary querying mechanisms to retrieve and interpret semantic alignments between knowledge models that undergo the reconciliation process. Experimental work has already been conducted regarding the “Semantic Reconciliation Layer” and the “Interoperability Evaluation Layer” and is to be documented in a manuscript which is currently under preparation.

6 Conclusions

The Common Logic-based slant adopted in this work contributes to the motivation for new heavyweight ontological formalisms applied to the field of design and manufacture, as opposed to the frequently-exploited Semantic Web approaches, which to some extent are limited in terms of logical expressiveness. This is because Common Logic is a First Order Logic language for knowledge interchange that provides a core semantic framework for logic together with the basis for a set of syntactic forms (dialects) all sharing a common semantics (Delugach, 2005). This work hence fulfils the task of illustrating the potentials for more expressive formalisms to allow the integrity-driven configuration of design and manufacture knowledge models.

However, it is clear from the breadth of the scope of this work, that it would be highly desirable to explore an extended heavyweight ontological foundation of design and manufacture. Extensions would require capturing a set of product lifecycle concepts, together with more complicated feature ranges such as pockets, splines and complex closed profiles. Moreover, to refine the definition of design and manufacturing features, an engaging facet would be the formalisation of relevant semantic relationships between part families and features. Such extensions would inevitably imply the need for improving the understanding behind the different nuances of conceptualisations within ontological foundations for design and manufacture.

Acknowledgements

The research developed in this paper has resulted from a number of strands of work which have been supported by different funding agencies. In particular we wish to thank (1) the EPSRC, who are funding a majority of our work on "Interoperable Manufacturing Knowledge Systems" (IMKS) under project 253 of the Loughborough University Innovative Manufacturing and Construction Research Centre, and (2) the Wolfson School of Mechanical and Manufacturing Engineering of Loughborough University for funding research studentships.

References

Abouel Nasr, E.S. and Kamrani, A.K., 2006. A new methodology for extracting manufacturing features from CAD system *Computers and Industrial Engineering*, 51, 389-415.

Aifaoui, N., Deneux, D. and Soenen, R., 2006. Feature-based interoperability between design and analysis processes. *Journal of Intelligent Manufacturing*, 17, 13-27.

Baader, F., Horrocks, I. and Sattler, U., 2007. Description logics. *In:* van Harmelen, F., Lifschitz, V. and Porter, B., eds. *Handbook of Knowledge Representation*. Elsevier.

Bock, C. and Gruninger, M., 2005. PSL: a semantic domain for flow models. *Software and Systems Modelling Journal*, 4, 209-231.

Bock, C., 2006. Interprocess communication in the process specification language. NISTIR 7348, NIST, Gaithersburg, MD, USA.

Borgo, S. and Leitão, P., 2008. Foundations for a core ontology of manufacturing. *In: Ontologies: a handbook of principles, concepts and applications in information systems*. Springer.

Brunnermeier, S.B. and Martin, S.A., 2002. Interoperability costs in US automotive supply chain. *Supply Chain Management: An International Journal*, 7(2), 71-82.

Canciglieri, O.J. and Young, R.I.M., 2003. Information sharing in multi-viewpoint injection moulding design and manufacturing. *International Journal of Production Research*, 41(7), 1565-1586.

Cheng, J., Gruninger, M., Sriram, R.D and Law, K.H, 2003. Process specification language for project scheduling information exchange. *International Journal of IT in Architecture, Engineering and Construction*, 1(4), 307-328.

Dartigues, C., Ghodous, P., Gruninger, M., Pallez, D. and Sriram, R., 2007. CAD/CAP integration using feature ontology. *Concurrent Engineering Research and Applications*, 15(2), 237-249.

Das, B., Cutting-Decelle, A.F., Young, R.I.M., Case, K., Rahimifard, S., Anumba, C.J. and Bouchlaghem, N., 2007. Towards the understanding of the requirements of a communication language to support process interoperation in cross-disciplinary supply chains. *International Journal of Computer Integrated Manufacturing*, 20(4), 396-410.

Delugach, H.S., 2005. Common Logic in support of metadata and ontologies. PowerPoint Presentation. *Open Forum 2005 on Metadata Registries*.

Fenves, S.J., Foufou, F., Bock, C., Sudarsan, R., Bouillon, N. and Sriram, R.D., 2004. CPM2: a revised core product model for representing design information. NISTIR 7185, NIST, Gaithersburg, MD, USA.

Fenves, S.J., Foufou, S., Bock, C. and Sriram, R.D., 2005. CPM: a Core Product Model for product data. *Journal of Computing and Information Science in Engineering*, 5, 238-246.

Gómez-Pérez, A., Fernández-López, M. and Corcho, O., 2004). Ontological engineering: with examples from the areas of knowledge management, e-commerce and the semantic web. London, UK: Springer-Verlag London Ltd.

Gruber, T.R., 1992. Ontolingua: a mechanism to support portable ontologies. Knowledge Systems, AI Laboratory (KSL-91-66).

Gunendran, A.G. and Young, R.I.M., 2006. An information and knowledge framework for multi-perspective design and manufacture. *International Journal of Computer Integrated Manufacturing*, 19(4), 326-338.

Hameed, A., Preece, A. and Sleeman, D., 2004. Ontology Reconciliation. *In:* Staab, S. and Studer, R., eds. *Handbook on ontologies*. Springer, 231-250.

Harold, E.R. and Means, W.S., 2004. *XML in a Nutshell.* 3rd Ed. Sebastopol, CA, USA: O'Reilly Media Inc.

ISO 18629, 2005. Industrial Automation Systems and Integration – Process Specification Language (PSL).

ISO 10303-224, 2006. STEP – Mechanical product definition for process planning using machining features.

ISO/IEC 24707, 2007. Information Technology – Common Logic (CL): a framework for a family of logic-based languages.

Knowledge Based Systems Inc., 1994. Information Integration for Concurrent Engineering (IICE): IDEF5 method report. Texas, USA. Available from: http://www.idef.com/pdf/Idef5.pdf
[Accessed November 2009]

Kugathasan, P. and McMahon, C., 2001. Multiple viewpoint models for automotive body-in-white design. *International Journal of Production Research*, 39(8), 1698-1705.

Mäs, S., Wang, F. and Reinhardt, W., 2005. Using ontologies for integrity constraint definition. *Proceedings of the 4th International Symposium on Spatial Date Quality*, Beijing, China.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N. and Oltramari, A., 2003. WonderWeb deliverable D18 – ontology library. Laboratory for Applied Ontology, ISTC-CNR, Trento, Italy.

Mertins, K., Ruggaber, R., Popplewell, K. and Xu, X., 2008. Preface. *In:* Mertins, K. et al., eds. *Enterprise interoperability III: new challenges and industrial approaches*. London, UK: Springer-Verlag London Ltd, v-vi.

Nassehi, A., Liu, R. and Newman, S.T., 200. A new software platform to support feature-based process planning for interoperable STEP-NC manufacture. *International Journal of Computer Integrated Manufacturing*, 20(7), 669-683.

Noy, N.F., Fergerson, R.W. and Musen, M.A., 2000. The knowledge model of Protégé-2000: combining interoperability and flexibility. *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modelling and Management*, LNCS, 1937, 17-32.

Noy, N.F. and McGuinness, D.L., 2001. Ontology development 101: a guide to creating your first ontology. Knowledge Systems Laboratory. Available from: http://www.ksl.stanford.edu/KSL_Abstracts/KSL-01-05.html
[Accessed November 2009]

Ontology Works Inc. Available from: http://www.ontologyworks.com [Accessed November 2009]

Pease, R.A. and Niles, I., 2002. IEEE Standard Upper Ontology: a progress report. *The Knowledge Engineering Review*, 17(1), 65-70.

Process Specification Language (PSL). Available from: http://www.mel.nist.gov [Accessed November 2009]

Ray, S.R, 2004. NIST’s semantic approach to standards and interoperability. PowerPoint Presentation. Available from: http://ontolog.cim3.net/file/resource/presentation/NIST_Semantics--SteveRay_20040212a.ppt
[Accessed November 2009]

Rector, A.L., 2003. Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. *Proceedings of the 2nd International Conference on Knowledge Capture*, Florida, USA, 121-128.

Research Triangle Institute, 1999. *Interoperability cost analysis of the US automotive supply chain*. 99-1 Planning Report Prepared for the National Institute of Standards and Technology. Available from: http://www.nist.gov/director/prog-ofc/report99-1.pdf
[Accessed November 2009]

Subrahmanian, E., Rachuri, S., Fenves, S.J., Foufou, S. and Sriram, R.D., 2005. Challenges in supporting product design and manufacturing in a networked economy: a PLM perspective. *Proceedings of the International Conference on Product Lifecycle Management*, 495-506.

Wang, H.H., Noy, N., Rector, A., Musen, M., Redmond, T., Rubin, D., Tu, S., Tudorache, T., Drummond, N., Horridge, M. and Seidenberg, J., 2006). Frames and OWL side by side. *In: 9th International Protégé Conference*, Stanford, California.

Young, R.I.M., Gunendran, A.G., Cutting-Decelle, A.F. and Gruninger, M., 2007. Manufacturing knowledge sharing in PLM: a progression towards the use of heavy weight ontologies. *International Journal of Production Research*, 45(7), 1505-1519.

Figure Captions

Figure 1. Viewpoints and domains in design and manufacture

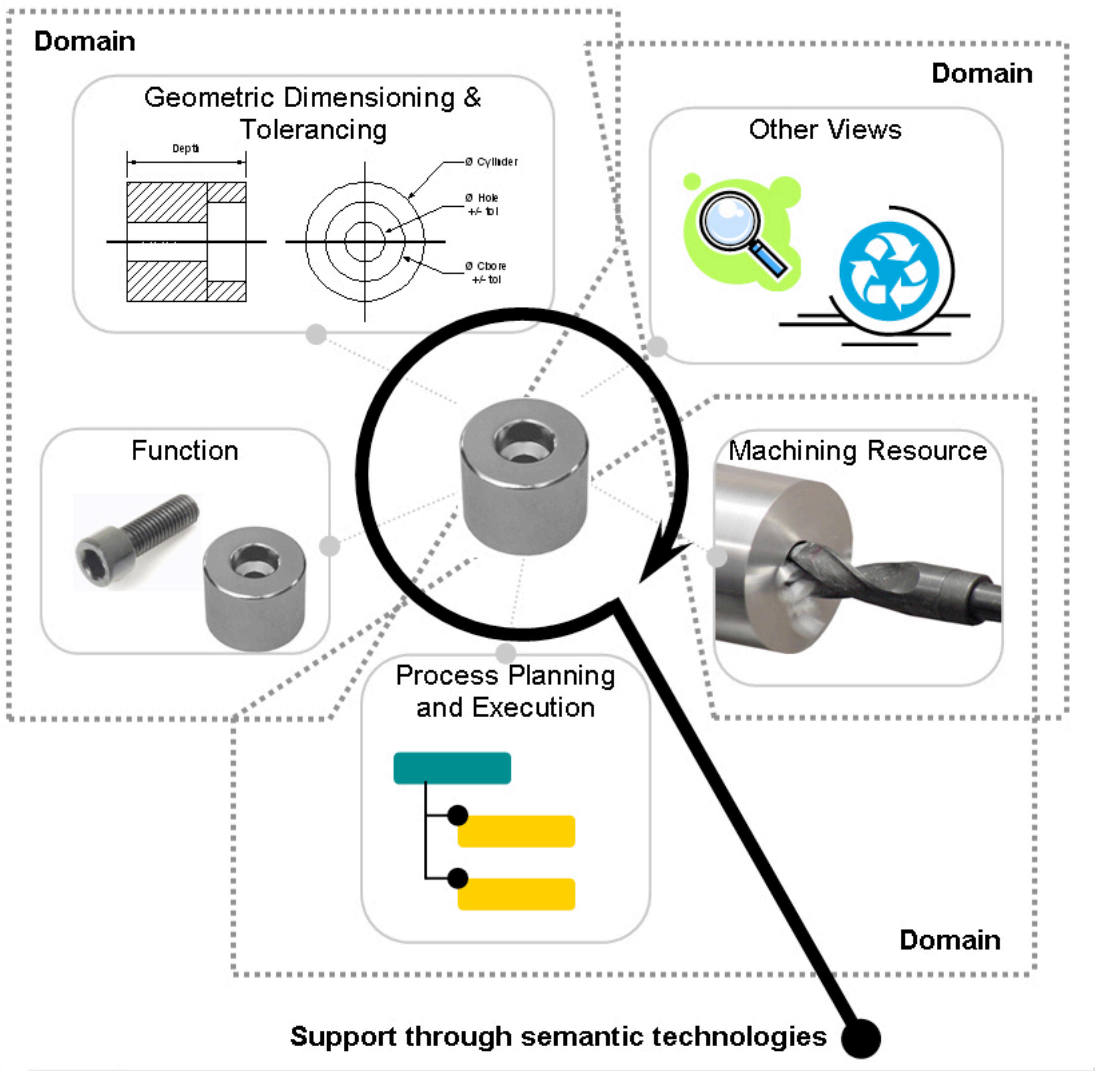

Domain
Geometric Dimensioning & Tolerancing
Depth
Ø Cylinder
Ø Hole +/- tol
Ø Cbore +/- tol
Domain
Other Views
Function
Machining Resource
Process Planning and Execution
Domain
Support through semantic technologies

Figure 2. Proposed ontology-based approach

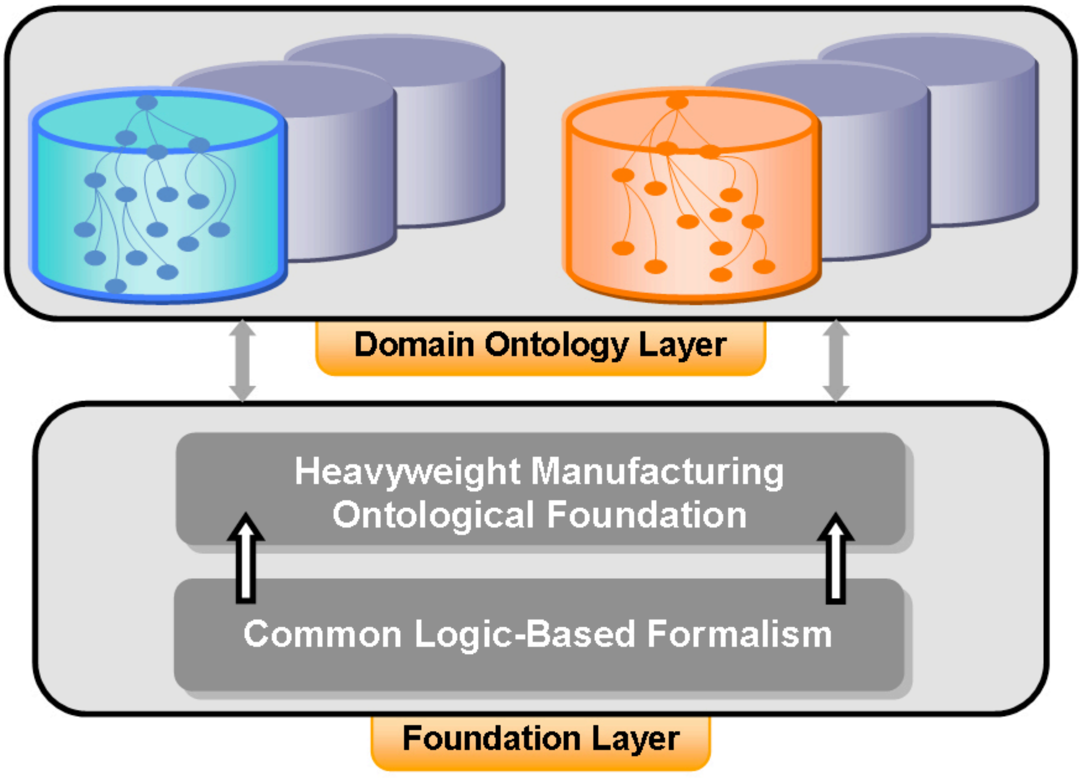

Domain Ontology Layer
Heavyweight Manufacturing Ontological Foundation
Common Logic-Based Formalism
Foundation Layer

Figure 3. Foundation layer

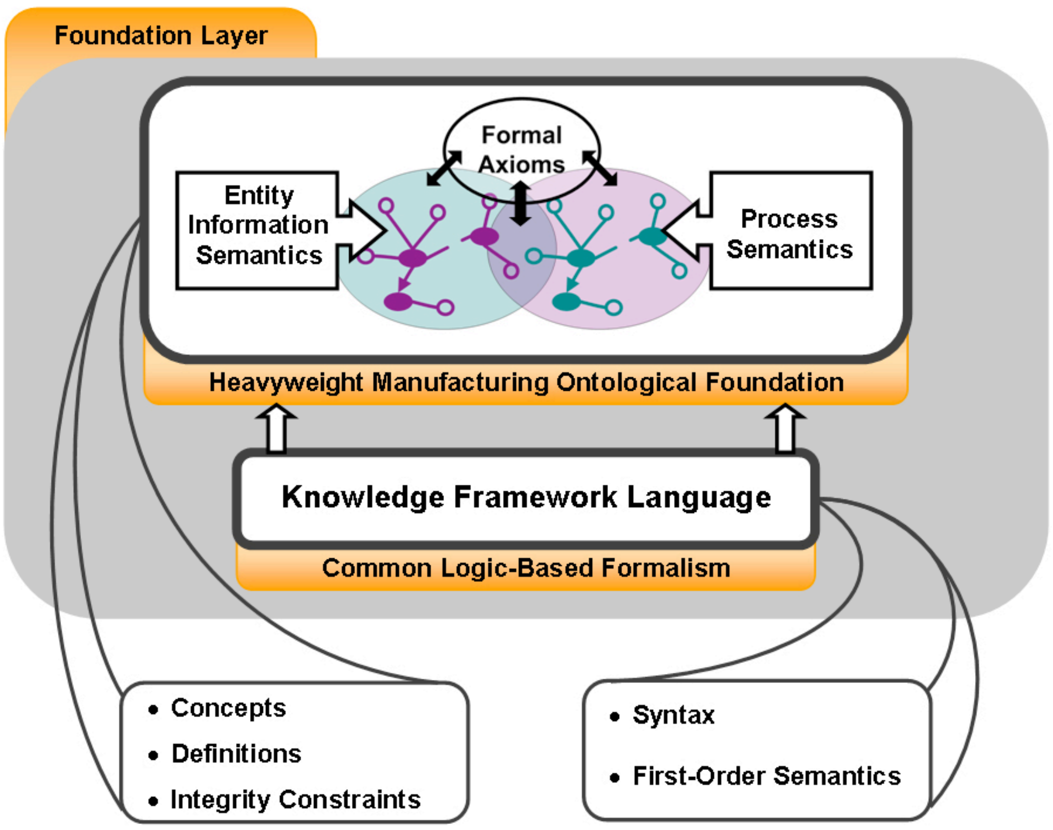

Foundation Layer
Formal Axioms
Entity Information Semantics
Process Semantics
Heavyweight Manufacturing Ontological Foundation
Knowledge Framework Language
Common Logic-Based Formalism
• Concepts
• Definitions
• Integrity Constraints
• Syntax
• First-Order Semantics

Figure 4. Taxonomy of main classes for the “Object” class

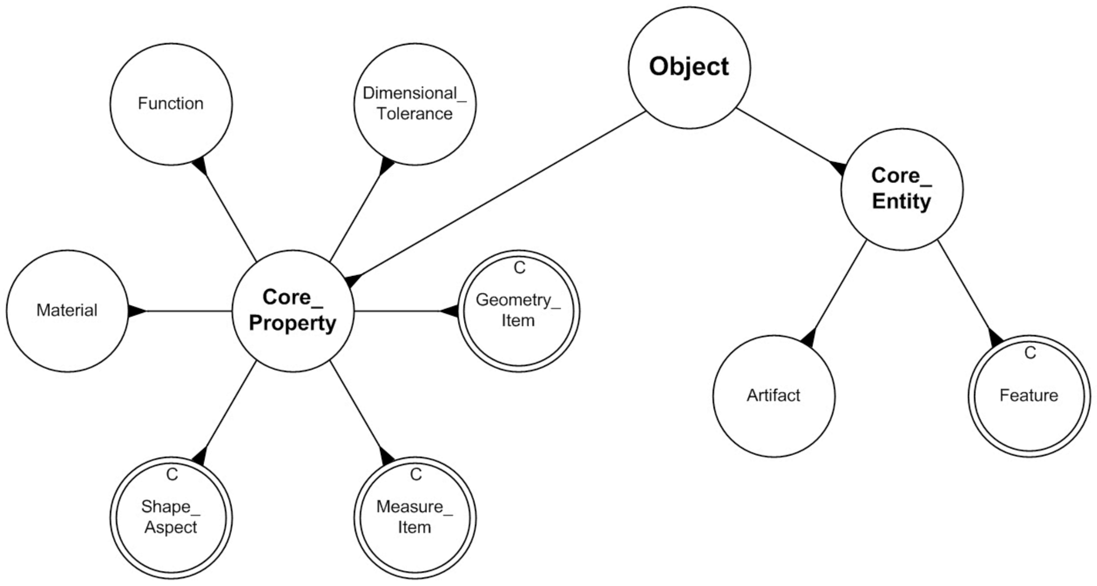
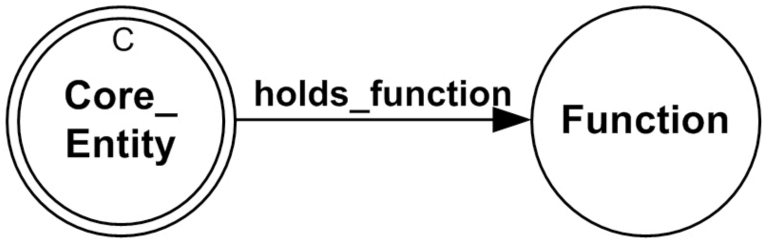


Figure 5. A soft integrity constraint



```
(=> (Core_Entity ?coreEnt)
     (exists (?func)
          (and (Function ?func)
               (holds_function ?coreEnt ?func))))
:IC soft "Every core entity holds some function"
```

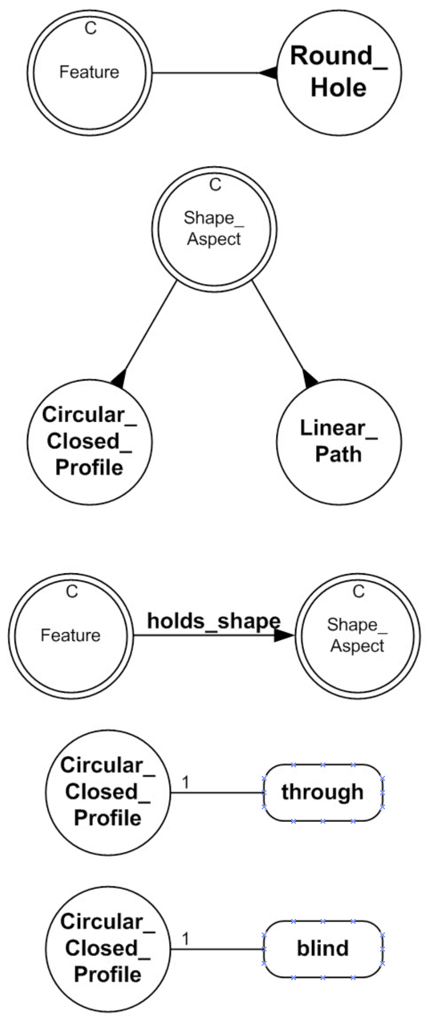
Figure 6. Hard integrity constraints over the class “Round_Hole”

```
(=> (Round_Hole ?hole)
     (exists (?ccp1 ?ccp2)
          (and (Circular_Closed_Profile ?ccp1)
                  (Circular_Closed_Profile ?ccp2)
                  (/= ?ccp1 ?ccp2)
                  (holds_shape ?hole ?ccp1)
                  (holds_shape ?hole ?ccp2))))
:IC hard "Every round hole feature can hold
exactly two circular closed profiles"
```

```
(=> (Round_Hole ?hole)
     (exists (?lin)
          (and (Linear_Path ?lin)
                  (holds_shape ?hole ?lin))))
:IC hard "Every round hole feature can hold
exactly one linear path"
```

```
(=> (and (Round_Hole ?hole)
             (Circular_Closed_Profile ?ccp1)
             (holds_shape ?hole ?ccp1)
             (blind ?ccp1))
      (not (exists (?ccp2)
          (and (Circular_Closed_Profile ?ccp2)
                  (holds_shape ?hole ?ccp2)
                  (through ?ccp2)))))
:IC hard "Every round hole that holds a blind
circular closed profile cannot have a through
circular closed profile"
```

```
(=> (and (Round_Hole ?hole)
             (Circular_Closed_Profile ?ccp1)
             (holds_shape ?hole ?ccp1)
             (through ?ccp1))
      (not (exists (?ccp2)
          (and (Circular_Closed_Profile ?ccp2)
                  (holds_shape ?hole ?ccp2)
                  (blind ?ccp2)))))
:IC hard "Every round hole that holds a through
circular closed profile cannot have a blind
circular closed profile"
```



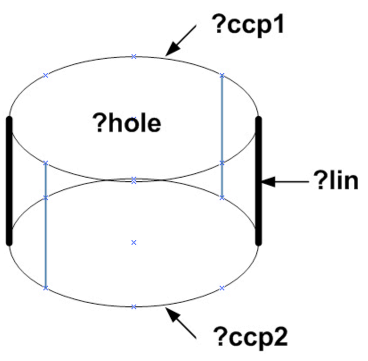
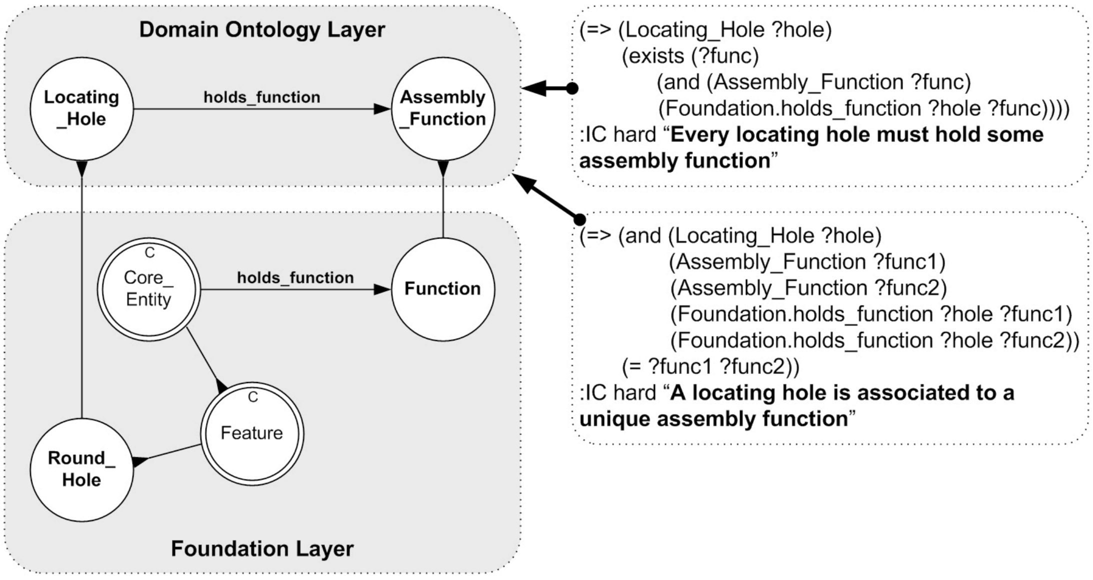


Figure 7. The configuration of concepts in domain ontologies



```
(=> (Locating_Hole ?hole)
      (exists (?func)
            (and (Assembly_Function ?func)
            (Foundation.holds_function ?hole ?func))))
:IC hard "Every locating hole must hold some
assembly function"
```

```
(=> (and (Locating_Hole ?hole)
            (Assembly_Function ?func1)
            (Assembly_Function ?func2)
            (Foundation.holds_function ?hole ?func1)
            (Foundation.holds_function ?hole ?func2))
      (= ?func1 ?func2))
:IC hard "A locating hole is associated to a
unique assembly function"
```

Figure 8. Discrete knowledge representation through instantiation

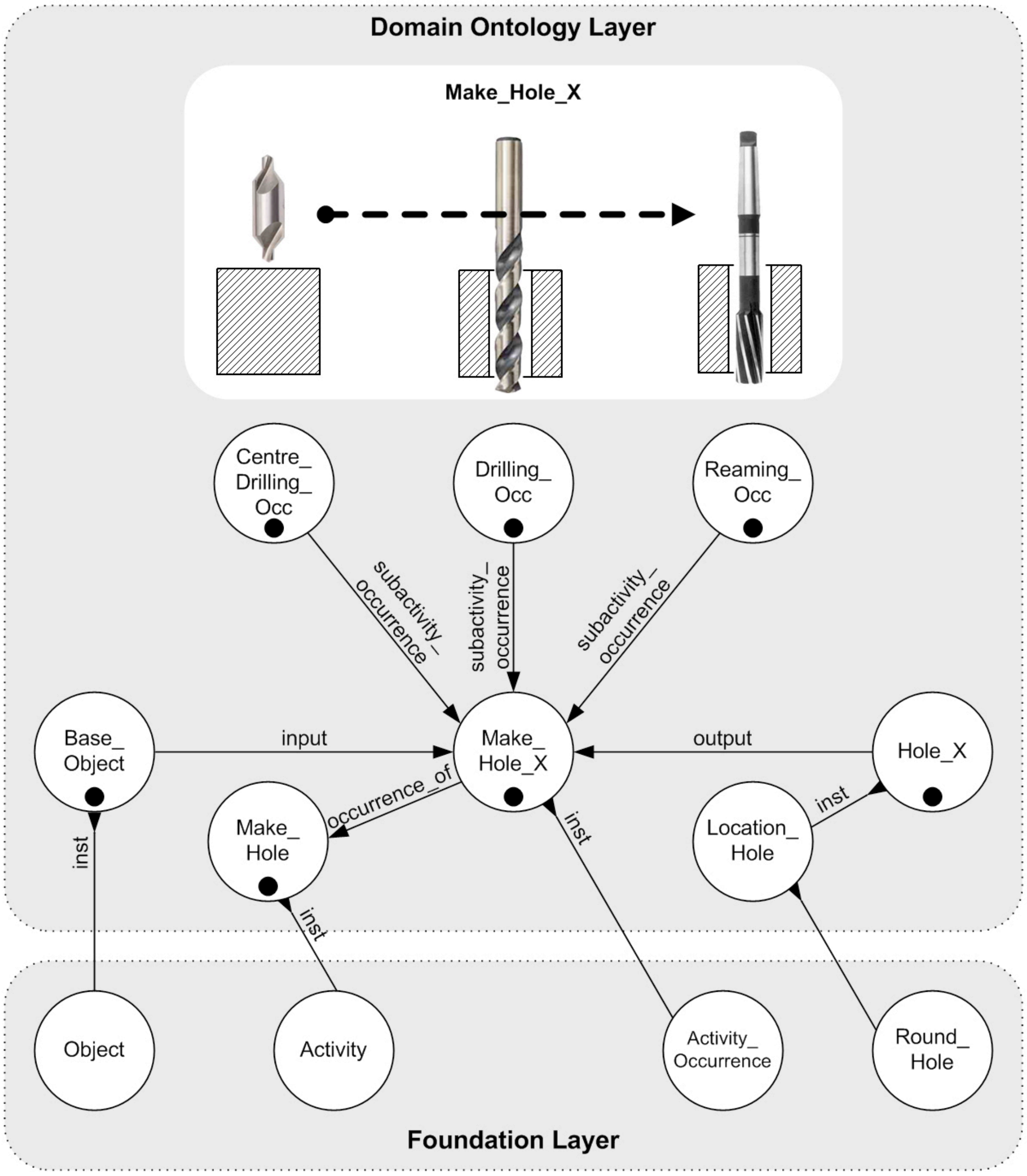
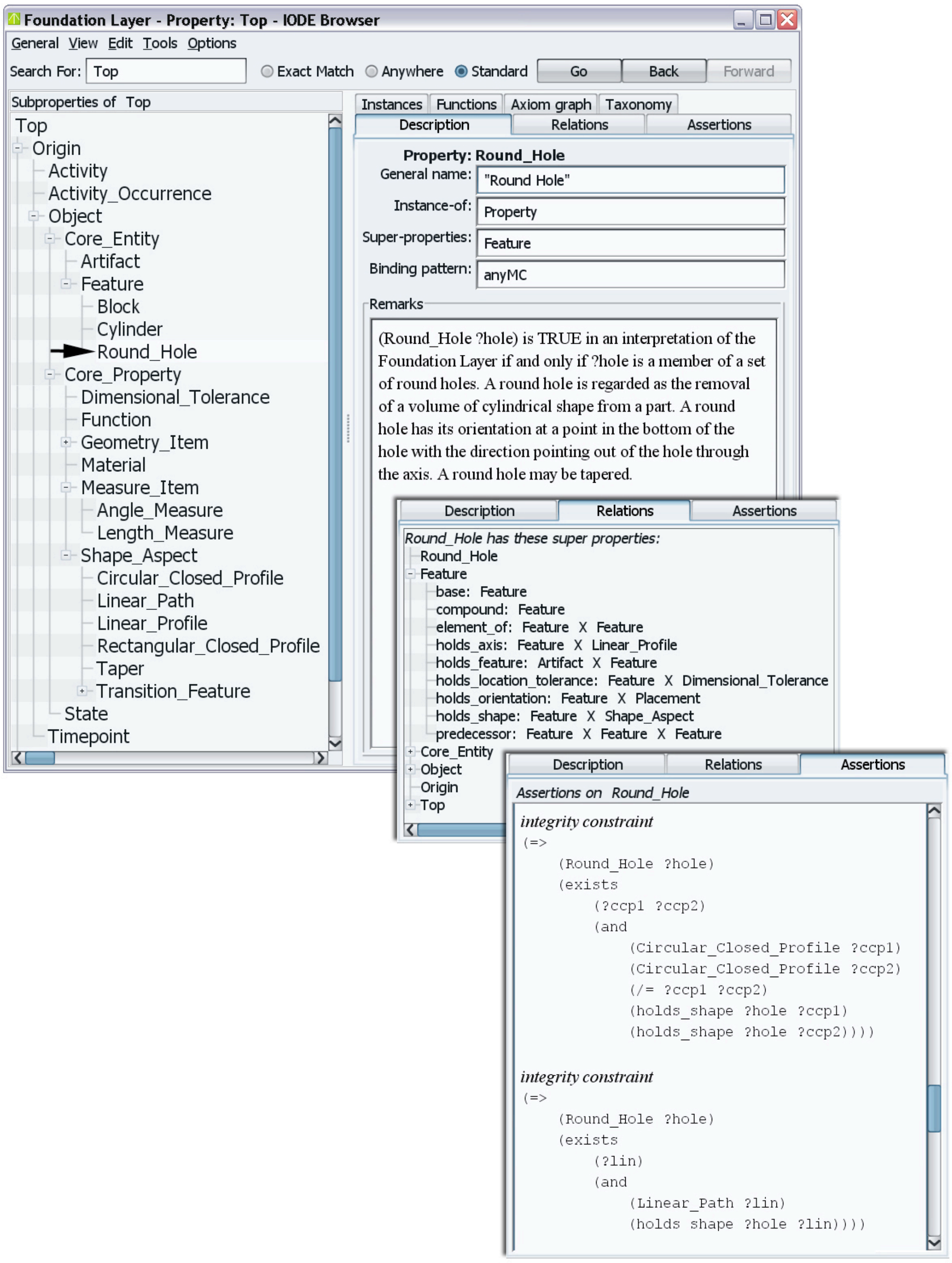


Figure 9. Implemented “Foundation Layer”


Foundation Layer - Property: Top - IODE Browser

General View Edit Tools Options

Search For: Top ○ Exact Match ○ Anywhere ● Standard Go Back Forward

Subproperties of Top

```
Top
 Origin
   Activity
   Activity_Occurrence
   Object
     Core_Entity
       Artifact
       Feature
         Block
         Cylinder
         Round_Hole
     Core_Property
       Dimensional_Tolerance
       Function
       Geometry_Item
       Material
       Measure_Item
         Angle_Measure
         Length_Measure
       Shape_Aspect
         Circular_Closed_Profile
         Linear_Path
         Linear_Profile
         Rectangular_Closed_Profile
         Taper
         Transition_Feature
     State
   Timepoint
```

Instances | Functions | Axiom graph | Taxonomy

Description | Relations | Assertions

Property: Round_Hole

General name:	"Round Hole"
Instance-of:	Property
Super-properties:	Feature
Binding pattern:	anyMC

Remarks

(Round_Hole ?hole) is TRUE in an interpretation of the Foundation Layer if and only if ?hole is a member of a set of round holes. A round hole is regarded as the removal of a volume of cylindrical shape from a part. A round hole has its orientation at a point in the bottom of the hole with the direction pointing out of the hole through the axis. A round hole may be tapered.

Description | Relations | Assertions

Round_Hole has these super properties:

```
Round_Hole
Feature
  base: Feature
  compound: Feature
  element_of: Feature X Feature
  holds_axis: Feature X Linear_Profile
  holds_feature: Artifact X Feature
  holds_location_tolerance: Feature X Dimensional_Tolerance
  holds_orientation: Feature X Placement
  holds_shape: Feature X Shape_Aspect
  predecessor: Feature X Feature X Feature
Core_Entity
Object
Origin
Top
```

Description | Relations | Assertions

Assertions on Round_Hole

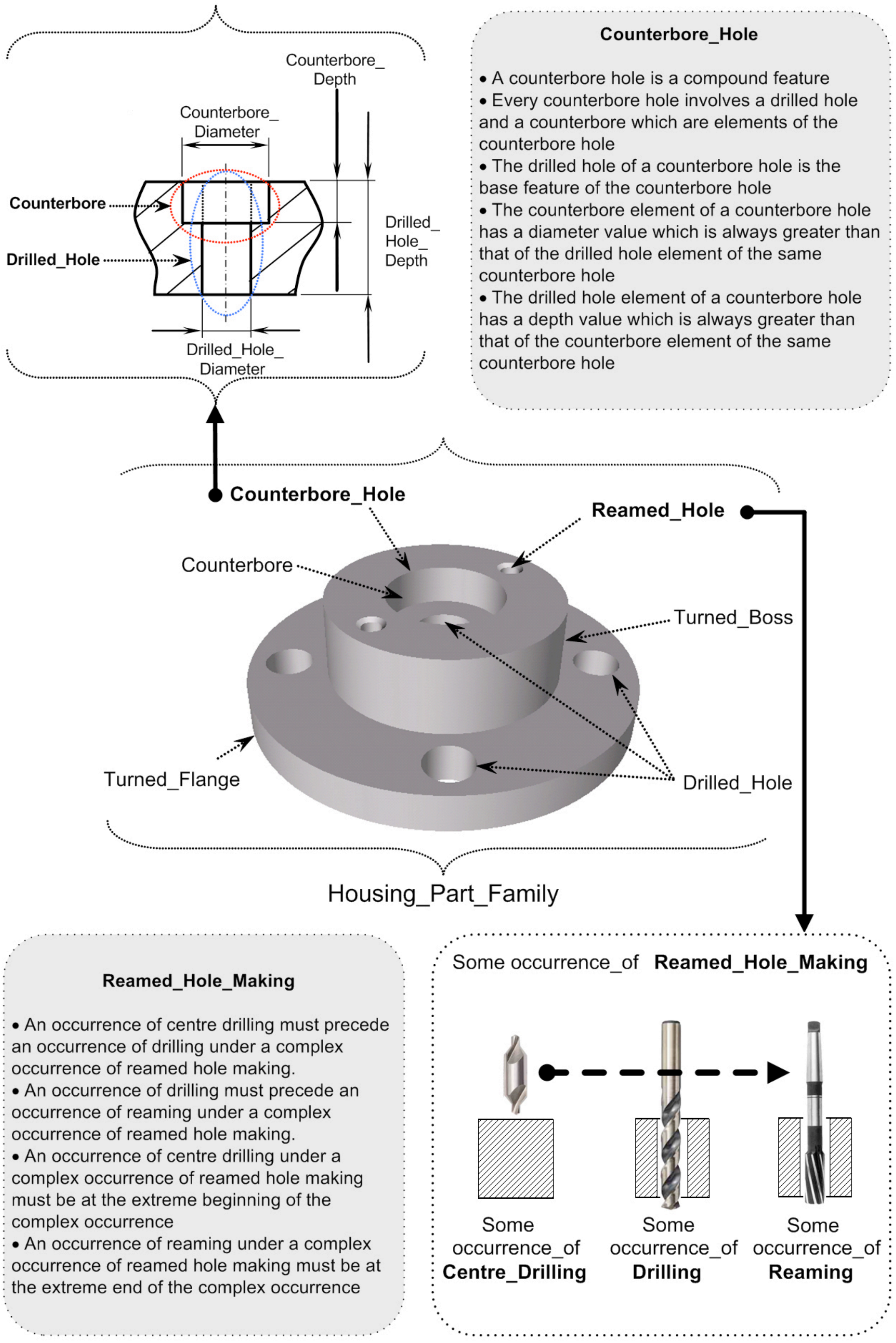
integrity constraint

```
(=>
    (Round_Hole ?hole)
    (exists
        (?ccp1 ?ccp2)
        (and
            (Circular_Closed_Profile ?ccp1)
            (Circular_Closed_Profile ?ccp2)
            (/= ?ccp1 ?ccp2)
            (holds_shape ?hole ?ccp1)
            (holds_shape ?hole ?ccp2))))
```

integrity constraint

```
(=>
    (Round_Hole ?hole)
    (exists
        (?lin)
        (and
            (Linear_Path ?lin)
            (holds shape ?hole ?lin))))
```

Figure 10. Example of explored domains-specific semantics



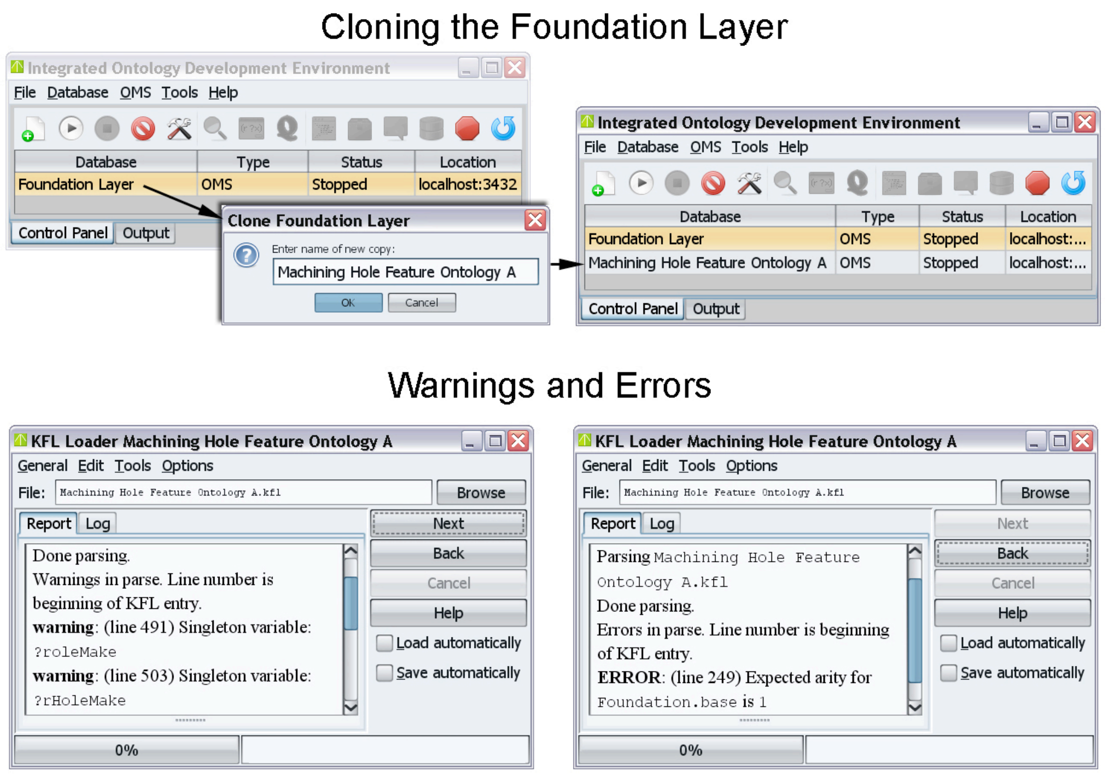
Counterbore_Hole

- A counterbore hole is a compound feature
- Every counterbore hole involves a drilled hole and a counterbore which are elements of the counterbore hole
- The drilled hole of a counterbore hole is the base feature of the counterbore hole
- The counterbore element of a counterbore hole has a diameter value which is always greater than that of the drilled hole element of the same counterbore hole
- The drilled hole element of a counterbore hole has a depth value which is always greater than that of the counterbore element of the same counterbore hole

Reamed_Hole_Making

- An occurrence of centre drilling must precede an occurrence of drilling under a complex occurrence of reamed hole making.
- An occurrence of drilling must precede an occurrence of reaming under a complex occurrence of reamed hole making.
- An occurrence of centre drilling under a complex occurrence of reamed hole making must be at the extreme beginning of the complex occurrence
- An occurrence of reaming under a complex occurrence of reamed hole making must be at the extreme end of the complex occurrence

Figure 11. Initial stage for implementing “Machining Hole Feature Ontology A”


Cloning the Foundation Layer
Integrated Ontology Development Environment
File Database OMS Tools Help
Control Panel
Output
Clone Foundation Layer
Enter name of new copy:
Machining Hole Feature Ontology A
OK
Cancel
Warnings and Errors
KFL Loader Machining Hole Feature Ontology A
General Edit Tools Options
File: Machining Hole Feature Ontology A.kfl
Browse
Report
Log
Next
Back
Cancel
Help
Load automatically
Save automatically
0%

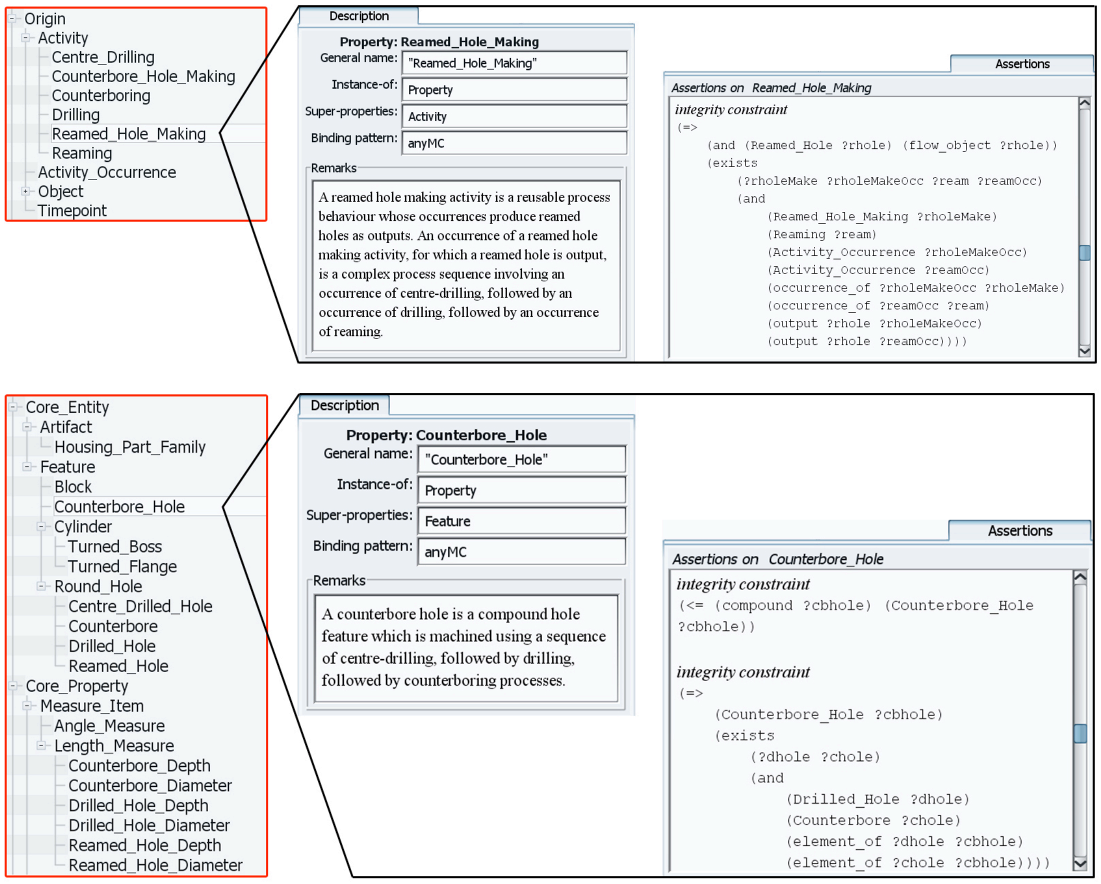
Database	Type	Status	Location
Foundation Layer	OMS	Stopped	localhost:3432

Database	Type	Status	Location
Foundation Layer	OMS	Stopped	localhost:...
Machining Hole Feature Ontology A	OMS	Stopped	localhost:...

Done parsing.
Warnings in parse. Line number is beginning of KFL entry.
warning: (line 491) Singleton variable: `?roleMake`
warning: (line 503) Singleton variable: `?rHoleMake`

Parsing `Machining Hole Feature Ontology A.kfl`
Done parsing.
Errors in parse. Line number is beginning of KFL entry.
ERROR: (line 249) Expected arity for `Foundation.base` is 1

Figure 12. Implemented “Machining Hole Feature Ontology A”



Origin
- Activity
  - Centre_Drilling
  - Counterbore_Hole_Making
  - Counterboring
  - Drilling
  - Reamed_Hole_Making
  - Reaming
- Activity_Occurrence
- Object
- Timepoint

Description

Property: Reamed_Hole_Making

General name:	"Reamed_Hole_Making"
Instance-of:	Property
Super-properties:	Activity
Binding pattern:	anyMC

Remarks

A reamed hole making activity is a reusable process behaviour whose occurrences produce reamed holes as outputs. An occurrence of a reamed hole making activity, for which a reamed hole is output, is a complex process sequence involving an occurrence of centre-drilling, followed by an occurrence of drilling, followed by an occurrence of reaming.

Assertions

Assertions on Reamed_Hole_Making

integrity constraint

```
(=>
    (and (Reamed_Hole ?rhole) (flow_object ?rhole))
    (exists
        (?rholeMake ?rholeMakeOcc ?ream ?reamOcc)
        (and
            (Reamed_Hole_Making ?rholeMake)
            (Reaming ?ream)
            (Activity_Occurrence ?rholeMakeOcc)
            (Activity_Occurrence ?reamOcc)
            (occurrence_of ?rholeMakeOcc ?rholeMake)
            (occurrence_of ?reamOcc ?ream)
            (output ?rhole ?rholeMakeOcc)
            (output ?rhole ?reamOcc))))
```

Core_Entity
- Artifact
  - Housing_Part_Family
- Feature
  - Block
  - Counterbore_Hole
  - Cylinder
    - Turned_Boss
    - Turned_Flange
  - Round_Hole
    - Centre_Drilled_Hole
    - Counterbore
    - Drilled_Hole
    - Reamed_Hole

Core_Property
- Measure_Item
  - Angle_Measure
  - Length_Measure
    - Counterbore_Depth
    - Counterbore_Diameter
    - Drilled_Hole_Depth
    - Drilled_Hole_Diameter
    - Reamed_Hole_Depth
    - Reamed_Hole_Diameter

Description

Property: Counterbore_Hole

General name:	"Counterbore_Hole"
Instance-of:	Property
Super-properties:	Feature
Binding pattern:	anyMC

Remarks

A counterbore hole is a compound hole feature which is machined using a sequence of centre-drilling, followed by drilling, followed by counterboring processes.

Assertions

Assertions on Counterbore_Hole

integrity constraint

```
(<= (compound ?cbhole) (Counterbore_Hole
?cbhole))
```

integrity constraint

```
(=>
    (Counterbore_Hole ?cbhole)
    (exists
        (?dhole ?chole)
        (and
            (Drilled_Hole ?dhole)
            (Counterbore ?chole)
            (element_of ?dhole ?cbhole)
            (element_of ?chole ?cbhole))))
```

Figure 13. Representing discrete knowledge in the KB

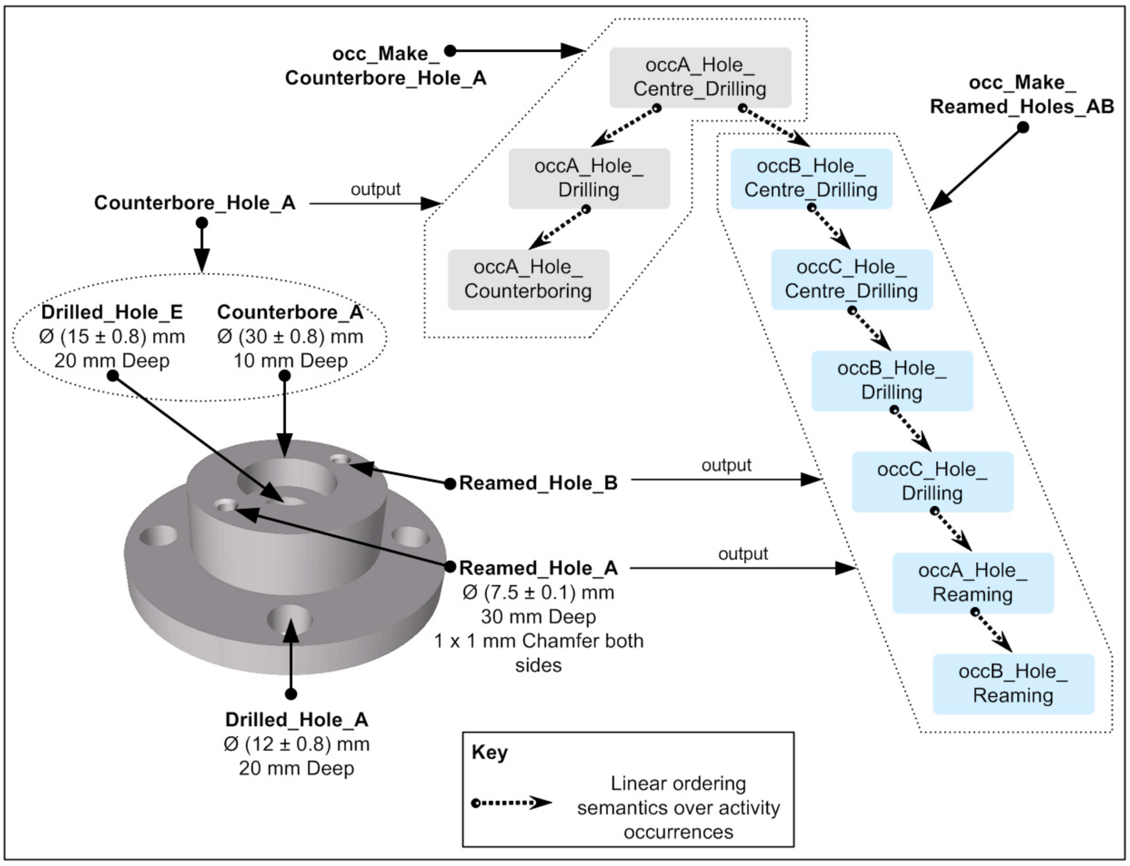
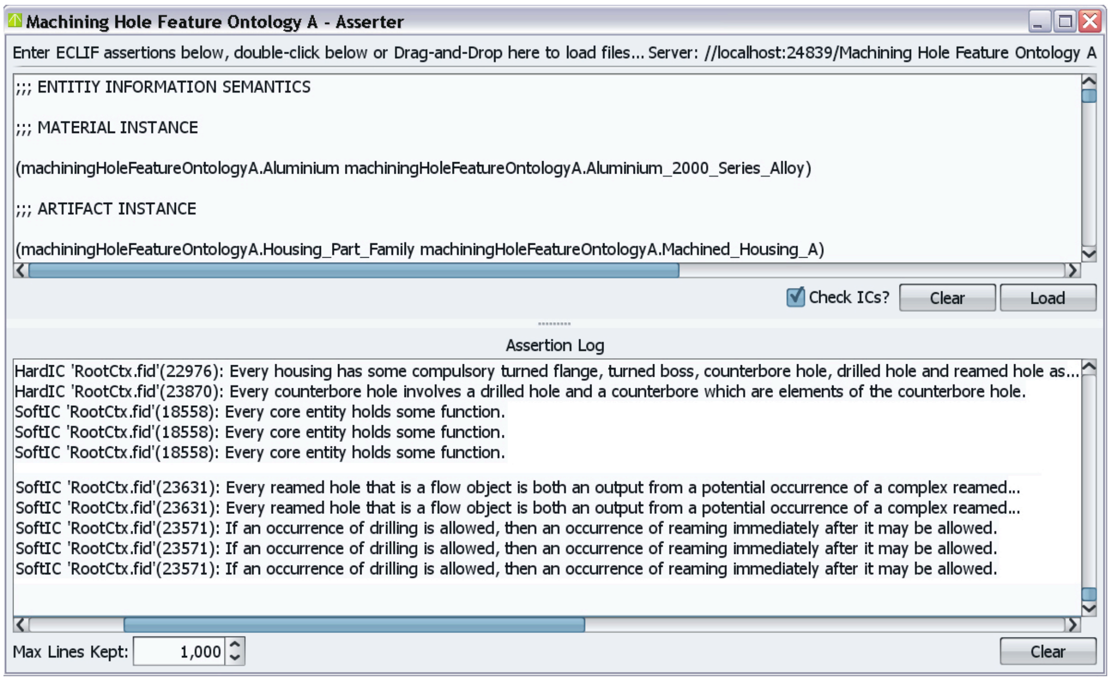


Figure 14. IC violations reported during the commitment transaction to the KB

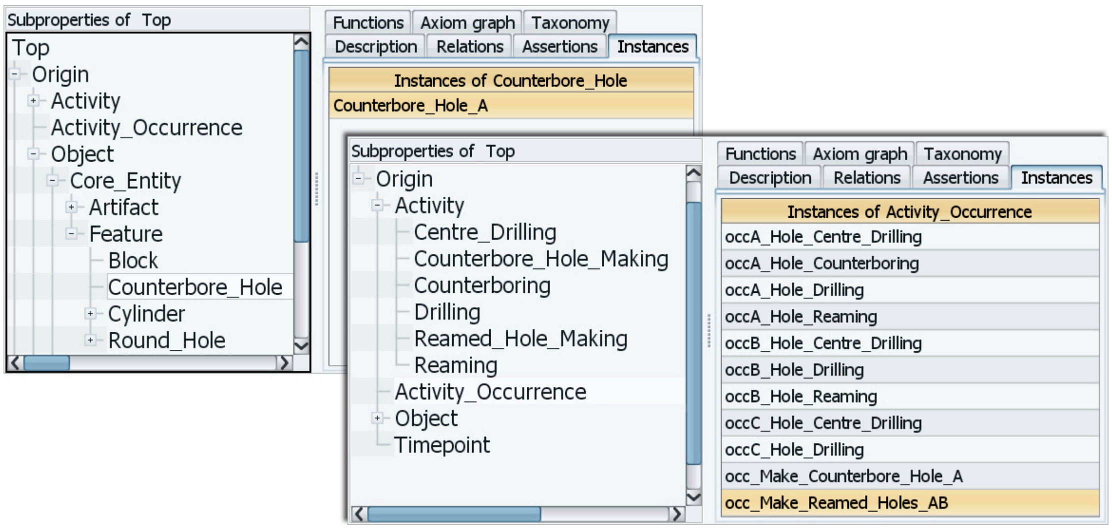


Machining Hole Feature Ontology A - Asserter
Enter ECLIF assertions below, double-click below or Drag-and-Drop here to load files... Server: //localhost:24839/Machining Hole Feature Ontology A
Check ICs?
Clear
Load
Assertion Log
Max Lines Kept: 1,000
Clear

```
;;; ENTITIY INFORMATION SEMANTICS

;;; MATERIAL INSTANCE

(machiningHoleFeatureOntologyA.Aluminium machiningHoleFeatureOntologyA.Aluminium_2000_Series_Alloy)

;;; ARTIFACT INSTANCE

(machiningHoleFeatureOntologyA.Housing_Part_Family machiningHoleFeatureOntologyA.Machined_Housing_A)
```

```
HardIC 'RootCtx.fid'(22976): Every housing has some compulsory turned flange, turned boss, counterbore hole, drilled hole and reamed hole as...
HardIC 'RootCtx.fid'(23870): Every counterbore hole involves a drilled hole and a counterbore which are elements of the counterbore hole.
SoftIC 'RootCtx.fid'(18558): Every core entity holds some function.
SoftIC 'RootCtx.fid'(18558): Every core entity holds some function.
SoftIC 'RootCtx.fid'(18558): Every core entity holds some function.

SoftIC 'RootCtx.fid'(23631): Every reamed hole that is a flow object is both an output from a potential occurrence of a complex reamed...
SoftIC 'RootCtx.fid'(23631): Every reamed hole that is a flow object is both an output from a potential occurrence of a complex reamed...
SoftIC 'RootCtx.fid'(23571): If an occurrence of drilling is allowed, then an occurrence of reaming immediately after it may be allowed.
SoftIC 'RootCtx.fid'(23571): If an occurrence of drilling is allowed, then an occurrence of reaming immediately after it may be allowed.
SoftIC 'RootCtx.fid'(23571): If an occurrence of drilling is allowed, then an occurrence of reaming immediately after it may be allowed.
```

Figure 15. Example of successfully created instances



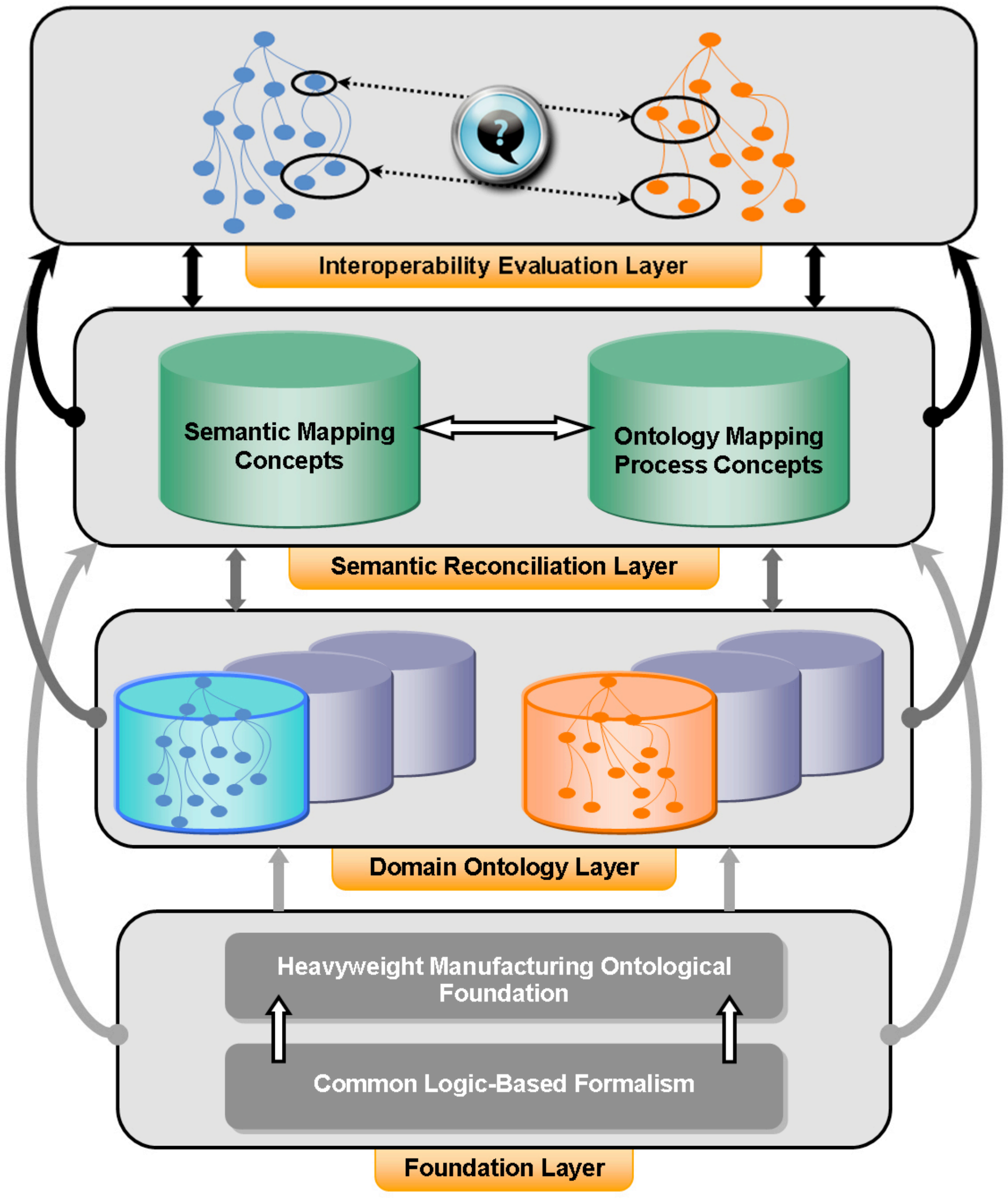
Top
- Origin
  - Activity
  - Activity_Occurrence
  - Object
    - Core_Entity
      - Artifact
      - Feature
        - Block
        - Counterbore_Hole
        - Cylinder
        - Round_Hole

Instances of Counterbore_Hole
Counterbore_Hole_A

- Origin
  - Activity
    - Centre_Drilling
    - Counterbore_Hole_Making
    - Counterboring
    - Drilling
    - Reamed_Hole_Making
    - Reaming
  - Activity_Occurrence
  - Object
  - Timepoint

Instances of Activity_Occurrence
occA_Hole_Centre_Drilling
occA_Hole_Counterboring
occA_Hole_Drilling
occA_Hole_Reaming
occB_Hole_Centre_Drilling
occB_Hole_Drilling
occB_Hole_Reaming
occC_Hole_Centre_Drilling
occC_Hole_Drilling
occ_Make_Counterbore_Hole_A
occ_Make_Reamed_Holes_AB

Figure 16. Semantic Manufacturing Interoperability Framework (SMIF)


Interoperability Evaluation Layer
Semantic Mapping Concepts
Ontology Mapping Process Concepts
Semantic Reconciliation Layer
Domain Ontology Layer
Heavyweight Manufacturing Ontological Foundation
Common Logic-Based Formalism
Foundation Layer