A Framework to Support Semantic Interoperability in Product Design and Manufacture

By

Nitishal Chungoora

Under the Supervision of

Dr. R. I. M. Young

A Doctoral Thesis

Submitted in partial fulfilment of the requirements

for the award of

Doctor of Philosophy of Loughborough University

January 2010

CERTIFICATE OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this thesis, that the original work is my own except as specified in acknowledgments or in footnotes, and that neither the thesis nor the original work contained therein has been submitted to this or any other institution for a degree.

…… (Signed)

Nitishal Chungoora

12th January 2010 (Date)

Abstract

It has been recognised that the ability to communicate the meaning of concepts and their intent within and across system boundaries, for supporting key decisions in product design and manufacture, is impaired by the semantic interoperability issues that are presently encountered. This work contributes to the field of semantic interoperability in product design and manufacture. An attribution is made to the understanding and application of relevant concepts coming from the computer science world, notably ontology-based approaches, to help resolve semantic interoperability problems.

A novel ontological approach, identified as the Semantic Manufacturing Interoperability Framework (SMIF), has been proposed following an exploration of the important requirements to be satisfied. The framework, built on top of a Common Logic-based ontological formalism, consists of a manufacturing foundation to capture the semantics of core feature-based design and manufacture concepts, over which the specialisation of domain models can take place. Furthermore, the framework supports the mechanisms for allowing the reconciliation of semantics, thereby improving the knowledge sharing capability between heterogeneous domains that need to interoperate and have been based on the same manufacturing foundation.

This work also analyses a number of test case scenarios, where the framework has been deployed for fostering knowledge representation and reconciliation of models involving products with standard hole features and their related machining process sequences. The test cases have shown that the Semantic Manufacturing Interoperability Framework (SMIF) provides effective support towards achieving semantic interoperability in product design and manufacture. Proposed extensions to the framework are additionally identified so as to provide a view on imminent future work.

Keywords: Ontology, Semantics, Interoperability, Common Logic, Knowledge Representation, Knowledge Sharing, Design and Manufacture.

Acknowledgements

This work has been supported through a research studentship funded by the Wolfson School of Mechanical and Manufacturing Engineering of Loughborough University.

First of all, I seize this opportunity to express my utmost gratitude to my supervisor, Dr. Bob Young. Bob is a very knowledgeable person who has occupied a key position throughout my three years of research. Under his supervision, I have benefited immensely from his positive coaching skills, continuous encouragement and critical but constructive comments. Without his supervision, I would not have been able to achieve the overall progress in this work. I also wish to thank Prof. Anne-Françoise Cutting-Decelle, Prof. Osiris Canciglieri, Prof. Keith Case and Dr. Jenny Harding for their advice and views regarding my work during the various meetings we had.

I wish to dedicate this work to my girlfriend, Luisa, who has constantly provided me with her moral support and motivation throughout my research and beyond…so baby, “quería decirte que no existen palabras para expresarte mi gratitud”. I also dedicate this work to my parents and brother, Veemal, who have always greatly supported and inspired me. My kind appreciation also goes to my friends Bara and Binoy for their encouragement and ideas, and to Uncle Robin and family.

I would like to thank George, from the research group, for helping me on several occasions. Thanks also to other people from the research group notably Claire, Zahid and Najam for listening to my long presentations. My gratefulness is also directed to Emily Wrobel, from Ontology Works Inc., for every single help concerning IODE and logic programming. Finally, I wish to extend my gratitude to the “Ministerio de Asuntos Exteriores y de Cooperación” of Spain for giving me the chance to undertake a one-month intensive course in Barcelona under the funding of the “Beca MAEC-AECID”.

Abbreviations

ADACOR	ADAptive holonic COntrol aRchitecture
AP	Application Protocol
API	Application Programming Interface
BFO	Basic Formal Ontology
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAPP	Computer Aided Process Planning
CIM	Computer Independent Model
CIMOSA	Computer Integrated Manufacturing Open System Architecture
CL	Common Logic
CLIF	Common Logic Interchange Format
CPM	Core Product Model
DIFF	Domain Independent Form Feature
DL	Description Logic
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
eCOIN	extended COntext INterchange
FCA	Formal Concept Analysis
FOL	First Order Logic
IC	Integrity Constraint
ICT	Information and Communications Technology
IDEF	Icam DEFinition
IODE	Integrated Ontology Development Environment
KB	Knowledge Base
KBE	Knowledge Based Engineering
KFL	Knowledge Framework Language
KIF	Knowledge Interchange Format
GD & T	Geometric Dimensioning and Tolerancing
GPU	Graphical Processing Unit
MAFRA	ontology MApping FRAmework
MANDATE	MANufacturing management DATa interchangE

MDA	Model Driven Architecture
MDI	Model Driven Interoperability
MSE	Manufacturing System Engineering
OKBC	Open Knowledge Base Connectivity
OMS	Object Management System
OWL	Web Ontology Language
PAL	Protégé Axiom Language
PDM	Product Data Management
PDM	Platform Description Model
PERA	Purdue Enterprise Reference Architecture
PFEM	Product Family Evolution Model
PIM	Platform Independent Model
PLIB	Parts LIBrary
PLM	Product Lifecycle Management
PSL	Process Specification Language
PSM	Platform Specific Model
PSRL	Product Semantic Representation Language
RDF	Resource Description Framework
RDF(S)	Resource Description Framework Schema
RM-ODP	Reference Model of Open Distributed Processing
SCL	Simple Common Logic
SMES	Saarbrucken Message Extraction System
SMIF	Semantic Manufacturing Interoperability Framework
STEP	STandard for the Exchange of Product model data
SUO	Standard Upper Ontology
SWRL	Semantic Web Rule Language
TOGAF	The Open Group Architecture Framework
UI	User Interface
ULO	Upper Level Ontology
UML	Unified Modelling Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XML	eXtensible Markup Language

Table of Contents

1 Introduction

1.1 Research Context

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). In order to achieve such capabilities, manufacturing enterprises weave their Information and Communications Technology (ICT) infrastructures to their established knowledge management strategies and practices. This is particularly important so as to maximise the benefits of reusable knowledge residing in several business processes.

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). Figure 1-1 illustrates this knowledge sharing scenario. It is shown that interoperable product design and manufacturing knowledge is required for (1) allowing seamless knowledge exchanges between multiple intra-system domains **(A)** and (2) permitting the reliable sharing of knowledge across systems **(B)**. This understanding is in line with the view that interoperation has to be established by the supply of information through inter- and intra- system communication (Chen *et al*, 2008).

Therefore, in modern collaborative PLM, design and manufacturing knowledge handled by decision support systems has to be efficiently communicated across the entire lifecycle. This knowledge is developed in activities based on Design for Function, Design for Assembly and Disassembly, Design for Manufacture, Manufacturing Planning and more. 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). Seamless interoperability, in order to effectively support collaborative product development, is still not completely achievable. This lack of interoperability is costly to many globally distributed industries where significant amounts of money are spent into overcoming interoperability problems (Research Triangle Institute, 1999; Brunnermeier and Martin, 2002).

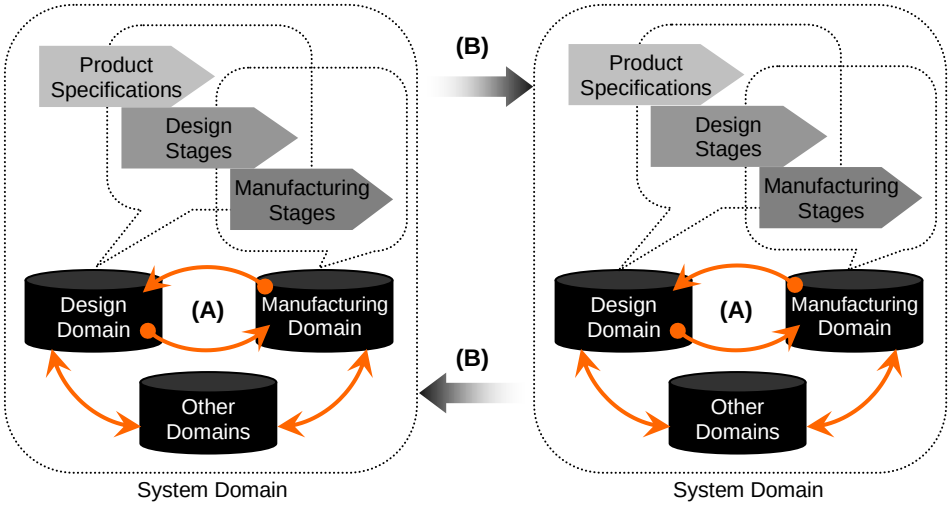


Figure 1-1 Interoperable Knowledge Sharing in Collaborative Product Development

A view on Figure 1-1 suggests that 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 as the level of flexibility required by multiple systems would be greatly impeded. The other possibility involves allowing different systems to develop and use their preferred methods, and to later worry about interoperability. 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 knowledge would demand considerable effort and may not provide optimal solutions to interoperability.

1.2 Research Hypothesis

Another possible way of realising interoperable knowledge sharing, which is under scrutiny in this research, is to adopt a direction where the meaning (i.e. semantics) associated to core design and manufacturing concepts cutting across all systems could be defined (see label **(C)** on Figure 1-2). Such core concepts may include, for example, the semantics associated to the definition of product features and manufacturing processes from several viewpoints arising in design and manufacture.

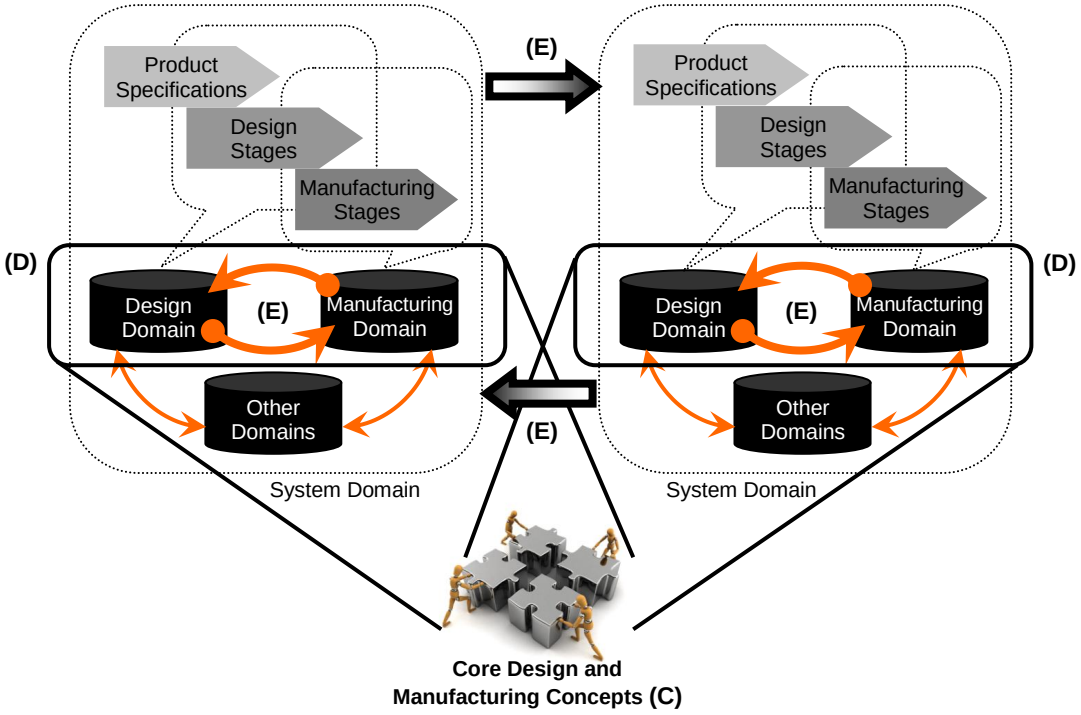


Figure 1-2 Motivation Scenario for the Research Hypothesis

These core or foundation concepts could be reused and extended, i.e. specialised, in a controlled manner by multiple design and manufacture domains across multiple system domains **(D)**. Following this approach, heterogeneous domains and system domains which use and specialise the meaning carried by the core concepts, would share a definitional basis which serves as a ground for interoperation **(E)**. In other words, the definition of mechanisms for enabling the reconciliation of intra- and inter-system semantics would raise the level of interoperability and knowledge sharing.

Linked to this understanding is the research hypothesis to be tested, which has been quoted next:

- The formal specification of a rigorously-defined set of sharable design and manufacture core concepts supports the structure for a heavyweight manufacturing ontological foundation (see Chapter 2 section 2.3.2 for a definition of heavyweight ontologies). The application of this shared foundation within and across system domains can provide a basis for the integrity-driven specialisation of design and manufacture domain models (i.e. formal ontology-based representations with their associated Knowledge Bases). The consequence of committing to this shared foundation can support the capability to evaluate and verify the correspondences between pairs of domain models that have been specialised from the foundation. These correspondences can help to identify the extent of sharable and non-sharable knowledge across the content of domain models.

The concept of ontologies is first introduced here and further explained in Chapter 2 and other chapters in the thesis. Broadly speaking, ontologies are formal models that provide a basis for sharing meaning (Young *et al*, 2007) in computational form. The concept originates from the computer science world and is showing promise in several areas of research including that of product design and manufacture.

In this work, a route towards satisfying the research hypothesis has involved the development of a novel ontology-based framework, identified as the Semantic Manufacturing Interoperability Framework (SMIF). This framework fulfils the task of (1) contributing to the understanding of combined heavyweight ontology-based approaches to support semantic interoperability in product design and manufacture, (2) consolidating knowledge behind the specification of a heavyweight manufacturing ontological foundation and the mechanisms involved in supporting the integrity-driven specialisation of domain models from the foundation, and (3) defining semantic reconciliation methods that are pertinent to both the evaluation and verification of

correspondences between domain models that have been based on the same foundation, as a means to identifying interoperable knowledge.

1.3 Research Strategy

1.3.1 Aim and Objectives

The aim of this work is to progress the understanding on ontology-based approaches to support semantic interoperability, applied to the field of product design and manufacture. This aim is to be addressed by demonstrating the feasibility of the research hypothesis. The achievement of the aim shall benefit the area of ontology-driven decision support systems in PLM. Other benefits include the ability to explicitly and formally capture design and manufacturing knowledge for reuse, which nowadays constitutes a core competence for the optimisation of collaborative product development practices. Furthermore, the realisation of the aim of this work shall benefit the support for effective knowledge sharing procedures between different agents within the product lifecycle.

With the intention of meeting the aim of this research, a number of key objectives have been identified. These cover namely:

1. The identification of key research gaps through a review of existing work on interoperable knowledge systems (see Chapter 2 section 2.8).
2. A study of the problems related to semantic interoperability in product design and manufacture, leading to the identification of key requirements to be satisfied in this research (see Chapter 3 section 3.4).
3. The proposal and exploration of a framework which meets the investigated requirements (see Chapter 4 section 4.9 for a summary of the proposed framework and chapters 5 and 6 sections 5.4 and 6.4 respectively for a summary of the exploration of the framework).
4. The development of an experimental system for implementing the framework (see Chapter 7 section 7.4 for a summary of the experimental system design).

5. The implementation of a number of test cases, as part of a complete case study, for testing the proposed framework and validating the solution (see Chapter 8 sections 8.3, 8.4, 8.5 and 8.6 for test case implementations and section 8.7 for a summary of the case study as a whole).
6. A proposition for extensions and modifications to the framework in order to support future work (see Chapter 9 sections 9.2 and 9.4 for more details).

1.3.2 Research Methodology

The research methodology adopted in this work builds on top of the previously listed objectives. Figure 1-3 depicts the flow within the research methodology. The main components of the literature review are portrayed **(F)**. Two ontology development and knowledge engineering techniques, notably IDEF5 schematics (Knowledge Based System Inc., 1994) **(G)** and the Knowledge Engineering Methodology **(H)** prescribed by Noy and McGuinness (2001), have also been applied to support the stages of proposing, exploring and experimenting the research framework. These two methods are described in Appendix A.

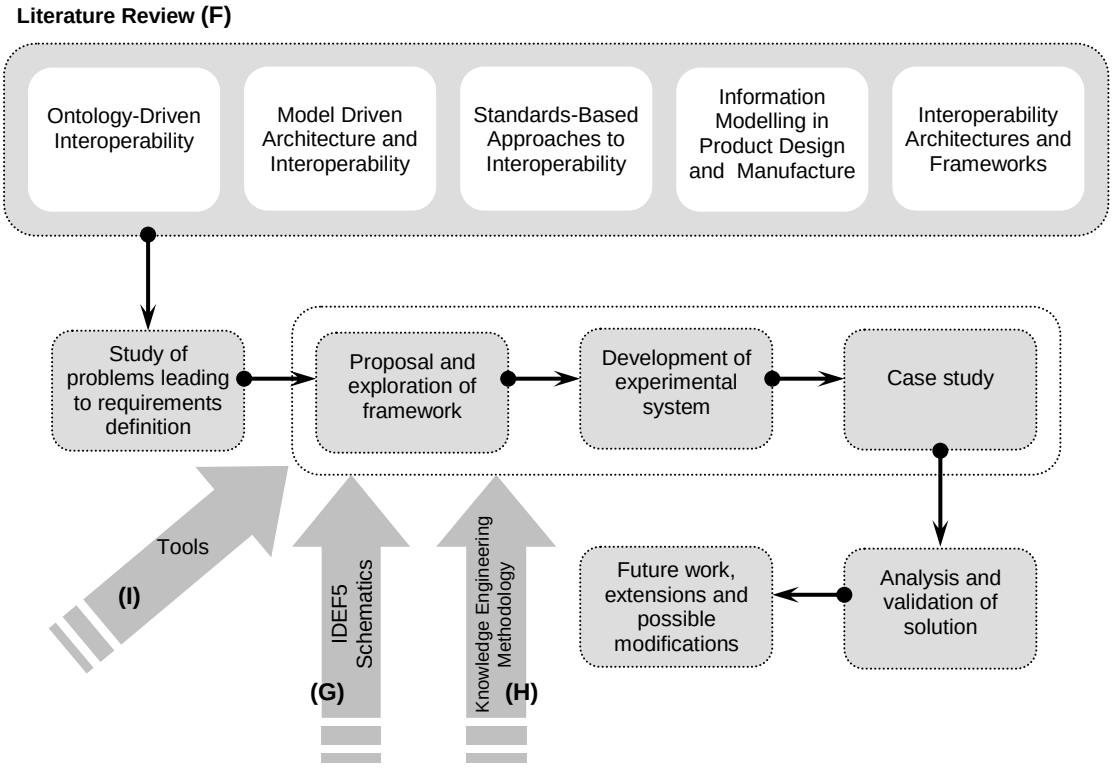


Figure 1-3 Research Methodology

It is to be noted that IDEF5 schematics are used for visually representing ontology-based content, but is not a fundamental method that complements the research methodology. This is because any other ontology visualisation methods could be employed as long as these are able to support the adequate visual representation of ontology-based content. Hence, IDEF5 schematics have only been used during the exploration and major ontology development tasks documented in chapters 5 and 6 and in Appendix C. Furthermore, relevant tools are to be identified in order to support the development and experimentation of research concepts. The harnessing of adequate tools and technologies **(I)** for satisfying this purpose forms an integral part of the research methodology and is particularly important towards the development of the experimental system and case study.

1.3.3 Research Scope

Whilst the essence of the concepts, investigated in the proposed framework, can be applied to a range of situations, the scope set to the work necessarily implies that the proof of these concepts works within clear boundaries and constraints. The research scope takes into consideration the domains of design and manufacture and their interoperability within and across system domains (refer to Figure 1-1). However, because of the substantial breadth of semantic interoperability issues in design and manufacture, this research focuses specifically onto simple product representations involving hole features in design and manufacture. For example, feature-based semantic representations for round holes constitute the chief scope, although other types of features such as cylinders have also been taken into account in order to provide a context for the existence of hole features on products. In addition, the research scope also involves the implications of machining process sequences for hole feature manufacturing and the participation relations between hole features and machining process sequences.

Hole feature manufacture is problematic and sometimes costly to industries, as a result of the diverse contexts, manufacturing processes and poorly established best practice methods associated to hole features (Chungoora

and Young, 2008a). Furthermore, the hole is one of the most complex geometrical features in prismatic machining and building effective hole-machining Computer Aided Process Planning (CAPP) system is still an important issue (Yongtao and Jingying, 2006). Hence, it is clear that several concerns still exist in relationship to the research scope. Moreover, another reason behind following the tightly-confined research scope implies the ability for this work to support the testing of the research hypothesis in its entirety.

1.3.4 Thesis Structure

A comprehensive literature review is first documented in Chapter 2. This helps to identify key research gaps that need to be addressed, so as to position this work according to these ongoing niches. The research problem is then further investigated in Chapter 3 and the observations made are used to establish a set of requirements, which dictate the specifications that this research attempts to satisfy. Based on these requirements, a novel ontology-based framework, the Semantic Manufacturing Interoperability Framework (SMIF), is proposed in Chapter 4.

The preferred concepts explored within the framework are further elaborated in the subsequent chapters 5 and 6. Chapter 7 documents the experimental system design and identifies appropriate software tools for deploying the framework. In Chapter 8, a number of test cases are analysed and validated as part of a case study, for providing a proof of concept. The overall understanding is further analysed in the concluding section of the thesis, in Chapter 9, where relevant drawbacks, possible modifications and extensions to the framework are finally exposed to provide an outlook on future work.

It is to be pointed out that the appendices C, D and E of the thesis capture the full development and implementation material required for the deployment of the framework and the analysed test cases. This has been made available for any party wishing to explicitly reproduce, verify and/or extend the concepts explored in this work.

2 Enabling Interoperable Manufacturing Knowledge Systems: a State-of-the-Art Review

2.1 Introduction

This chapter presents a state-of-the-art review on a number of active research directions related to the topic of supporting interoperability in product design and manufacture. The review is aimed at exposing the current understanding behind other research achievements made to date, in order to carefully depict ongoing niches that this research targets. Section 2.2 firstly describes interoperability as a general concept. This is then focused at semantic interoperability and its influence on knowledge sharing. With this preliminary view onto interoperability, section 2.3 then explains how semantic interoperability issues have so far been tackled using ontology-based approaches.

Section 2.4 discusses the concept of Model Driven Interoperability aided through the Model Driven Architecture. This then leads to an explanation of efforts fostered from the ISO standards community (Section 2.5) to enable common grounds to be adopted to enhance integration among stakeholders. Since this research work also emphasises on the capture of interoperable manufacturing knowledge for reuse, a special slant is given to information modelling in design and manufacture (Section 2.6). Section 2.7 is dedicated to providing a view on current interoperability architectures and frameworks, oriented at the enterprise level, as well as at the more defined world of product design and manufacture. A summary is then provided in section 2.8.

2.2 Interoperability of Information and Knowledge

2.2.1 Interoperability Definitions and Concerns

The term “interoperability” is defined as the ability to share technical and business data, information and knowledge seamlessly across two or more

software tools or application systems in an error free manner with minimal manual interventions (Ray and Jones, 2003). Other definitions for the term “interoperability” have been proposed, for example, by Chen *et al* (2008) who specify that from a computer technology viewpoint, interoperability is the faculty for two heterogeneous computer systems to function jointly and give access to their resources in a reciprocal way.

These definition, when extended to the field of product design and manufacture, is analogous to the seamless exchange of product and manufacture-centric information and knowledge across multiple expert systems. A number of key problems currently exist, which prevents the achievement of total product lifecycle interoperability. One of the most obvious issues is related to handling incompatible data and information structures between different platforms that need to interoperate (Brunnermeier and Martin, 2002; Cutting-Decelle *et al*, 2002; Das *et al*, 2007).

In addition to this, Das *et al* (2007) also point out that the most common reason to account for the lack of interoperability is due to the incompatibility between the syntaxes of the languages and the semantics of the terms used by the languages of software application systems. This statement is in concordance with Pouchard *et al* (2000), who have observed that the problems of interoperability are acute for manufacturing applications as these do not necessarily share syntax and definitions of concepts (i.e. semantics). To reinforce this view, Ray and Jones (2003) emphasise that either common terms are used to mean different things or different terms are used to mean the same thing, thereby resulting in problems related to ambiguous semantics (Young *et al*, 2007). This explains interoperability issues at the semantic level, and it becomes clear that an important leap is required to investigate new ways of promoting semantic interoperability.

2.2.2 Semantic Interoperability and Knowledge Sharing

Logical semantics or formal semantics, as used in the context of this work, is defined as the investigation of the meaning, or interpretation, of expressions

in specially constructed logical systems with the aid of mathematical logic (Lyons, 1977). Following this definition of formal semantics and the definition of semantic interoperability adopted by Yang and Zhang (2006), a view on semantic interoperability as employed in this work can be formulated. This states that semantic interoperability is the ability to support multiple applications in such a way that the computational meaning of the concepts defined in these applications can be jointly interpreted and shared.

Some of the implications of semantic interoperability to enable knowledge sharing have been considered (Yang and Zhang, 2006; Chungoora and Young, 2008a; Lazenberger *et al*, 2008; Ye *et al*, 2008). The main observation reveals that a progression towards improved methods for semantic interoperability shall support the potential for more effective information and knowledge exchanges. This additionally demonstrates that there exists a gap as far as semantic interoperability and knowledge sharing are concerned. A number of approaches that help support interoperability (and semantic interoperability) are next discussed.

2.3 Ontology-Driven Interoperability

2.3.1 Ontology Definitions

Ontology engineering is recognised as a key technology to deal with the semantic interoperability problem (Yang and Zhang, 2006). Available literature on ontological engineering points to a number of definitions for describing what an ontology is. A philosophical viewpoint is a common perspective from which an ontology can be defined such as the definition portrayed by Gruber (1993), in which an ontology is said to be an explicit specification of a conceptualisation.

This view has also been adopted by Studer *et al* (1998) to propose the definition that: “An ontology is a formal, explicit specification of a shared conceptualisation”. This definition adopts a slant towards how ontologies are realised at applications level. This is because the words in the definition have

been carefully chosen, for instance, (1) the word “explicit” reflects the exactness in the concepts, constraints and interpretations present in an ontology, (2) the word “formal” implies that the ontology should be machine-readable and (3) the words “shared conceptualisation” reflect the essence that an ontology aims at capturing agreed concepts over some field of knowledge.

Another relevant definition for an ontology is that provided in ISO 18629 (2005), stating that an ontology is “a lexicon of specialised terminology along with some specification of the meaning of the terms in the lexicon”. This description has led to the emphasis that an ontology is a representation or model that provides a basis for sharing meaning (Young *et al*, 2007). Very often, an ontology is regarded as being a multi-dimensional model of some domain of interest. Figure 2-1 identifies the multi-dimensional nature of an ontology. The figure, partly based on the structural view of what an ontology consists of (Labrou, 2002; Gómez-Pérez *et al*, 2004), regroups elements from the various definitions.

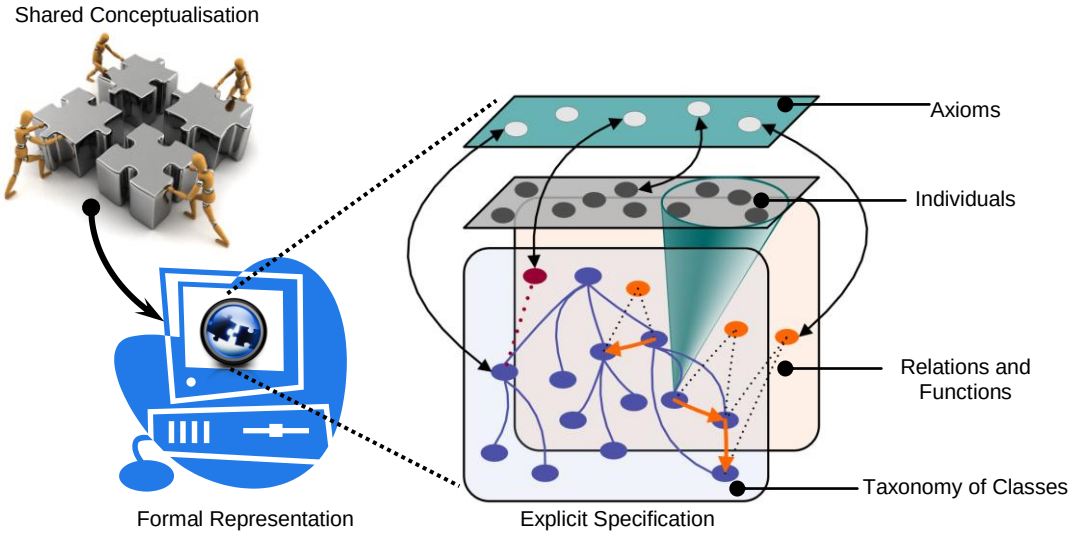


Figure 2-1 The Multi-Dimensional Nature of an Ontology

The structural view on an ontology (based on Labrou (2002) and Gómez-Pérez *et al* (2004)) indicates that the latter is typically composed of a (1) taxonomy of classes, which provides the backbone for organising concepts, (2) relations and functions which are used to build associations among concepts, (3) axioms which dictate the constraints over the ontological content and (4) individuals which are specific occurrences of classes.

2.3.2 Lightweight and Heavyweight Ontologies

An important distinction is made between ontologies in terms of the degree of expressiveness that they capture. Simple ontologies that only involve taxonomies of concepts and basic relations are referred to as lightweight ontologies (Fernández-López and Gómez-Pérez, 2002; Gómez-Pérez *et al*, 2004). Lightweight ontological approaches assume that the meaning associated to the terms of concepts within an ontology can readily be understood.

Heavyweight ontological approaches, on the other hand, on top of having the lightweight ontological structures also benefit from axioms in the form of constraints. These axioms are used to clarify the intended meaning of the terms gathered on the ontology (Gómez-Pérez *et al*, 2004). The configuration of the explicit specification captured in Figure 2-1 is that of heavyweight ontological structures. It is to be noted that in the case of a lightweight ontology, the axiom layer shown in Figure 2-1 is not be present. Additionally, Figure 3-4 in section 3.3.4.1 of Chapter 3 illustrates some common examples of lightweight and heavyweight ontological approaches.

It is clear from a semantic viewpoint, that the presence of limitations over the formal meaning of ontological content in lightweight ontologies explain their inappropriateness for inter-system interoperability (Young *et al*, 2007). For this reason, Young *et al* (2007) have also identified a need for more mathematically rigorous approaches in order to ensure that the true meaning behind the terminology coming from different systems is identical. This work, thus pursues this direction in order to reinforce and extend the understanding behind exploiting heavyweight ontological methods to drive semantic interoperability in design and manufacture.

2.3.3 Ontological Formalisms

Several ontology languages, also referred to as ontological formalisms or knowledge representation formalisms, are nowadays available for constructing ontologies. A comprehensive review of the spectrum of these languages is provided in Gómez-Pérez *et al* (2004) and in the current literature review, only the implications of the most relevant ontological formalisms are explained. Figure 2-2, partly adapted from Gómez-Pérez *et al* (2004), summarises the layout of these languages paying attention to draw a distinction between traditional ontology languages versus ontology markup languages.

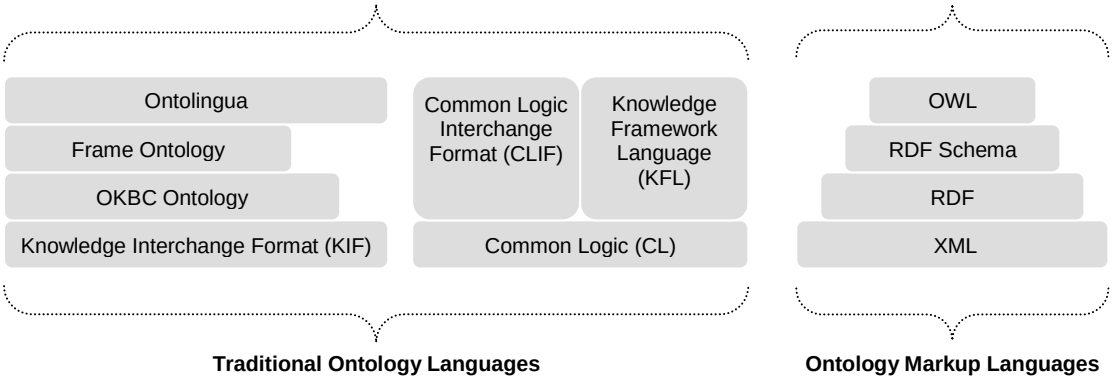


Figure 2-2 Formalisms for Building Ontologies

The main perceived difference between traditional ontology languages and ontology markup languages is that the former generally have a First Order Logic base while the latter are Description Logic-based (although Description Logic (DL) itself corresponds to the decidable fragment of First Order Logic (FOL)). Ontology markup languages help exploit the characteristics of the Semantic Web as a result of the boom of the Internet (Corcho, 2005). In traditional ontology languages, the Knowledge Interchange Format (KIF) (Genesereth and Fikes, 1992), which is FOL-based, supports the construction of the Open Knowledge Base Connectivity (OKBC) ontology (Chaudhri *et al*, 1998), Frames-based ontologies and Ontolingua (Farquhar *et al*, 1997), the latter using a combination of Frames and FOL.

More recently, with the introduction of Common Logic (CL) (ISO/IEC 24707) as a language framework for knowledge interchange, other ontological languages have been developed, for instance, (1) the Common Logic Interchange Format (CLIF), which is directly based on the CL standard itself and (2) the Knowledge Framework Language (KFL), developed by Ontology Works Inc. (Ontology Works Inc., 2009).

Ontology markup languages, as opposed to the traditional ontology languages, have their syntax supported by the eXtensible Markup Language (XML) to address flexible information structuring (Nurmilaakso *et al*, 2002). The XML capability allows the specification of the Resource Description Framework (RDF) and RDF Schema (Lassila and Swick, 1999) to support the ability to process metadata for providing interoperability between applications that exchange machine understandable information (Cingil and Dogac, 2001). The RDF and RDF Schema stack shown on Figure 2-2 then provides even further potentials, where the Web Ontology Language (OWL) has been pursued (Bechhofer *et al*, 2004), for capturing more rigorous properties required for building more meaningful DL ontologies.

One of the observations deriving from the identified ontological languages is that there exists an ongoing requirement to refine the understanding of the level of logic expressiveness (related to ontological formalisms) capable of semantically structuring the meaning of product lifecycle concepts (Young *et al*, 2009). Being a relatively new ontological direction, Common Logic-based ontological formalisms as a means to support semantic interoperability in product design and manufacture has not yet been given due attention. This work thus aims at contributing to this aspect (consult Chapter 3 and Appendix B for more details).

2.3.4 Foundation Ontologies

Ontological engineering embraces different levels of conceptualisation, from the general to the more specific. These gradations of conceptualisations include the upper or top level towards the domain level. Domain ontologies

are generally developed according to the preferences of specific fields of knowledge. Foundation ontologies also referred to as upper or top-level ontologies, on the other hand, are regarded as theories that capture the most common concepts relevant to many tasks and represent human commonsense which is hard to formalise (Kiryakov *et al*, 2001a). These theories involve the definitions of general concepts and formal axioms that govern the ways in which to interpret these theories. Foundation ontologies are also sometimes regarded as “formal” or “foundational” ontologies (Borgo and Leitão, 2007), due to their significance in supporting mutual understanding and interoperability among people and machines (Masolo *et al*, 2003).

The Basic Formal Ontology (BFO) is an example of a foundation ontology, whose core identifies the “SNAP” and “SPAN” which provide foundation theories for notions about objects and processes respectively, spanning over time (Grenon, 2003). Other established foundation ontologies include the Cyc’s Upper Ontology, developed under the Cyc project (Lenat and Guha, 1990) and the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE). The latter aims at capturing the ontological categories underlying natural language and human commonsense (Masolo *et al*, 2003). Particularly relevant to the field of manufacturing engineering is the development of the ADAptive holonic COntrol aRchitecture for distributed manufacturing systems (ADACOR) ontology (Leitão *et al*, 2005), which uses concepts from the DOLCE foundation ontology to provide a core ontology of manufacturing. A segment of the primary concepts of ADACOR are portrayed in the re-drawn UML class diagram in Figure 2-3.

Another wave of foundational ontologies involve (1) WordNet (Miller, 1995) which is an example of a top-level linguistic ontology whose purpose is to describe semantic constructs that offer a heterogeneous amount of resources, used mostly in natural language processing (Gómez-Pérez *et al*, 2004), (2) the Standard Upper Ontology (SUO) (Pease and Niles, 2002), formalised in SUO-KIF (a variant of the Knowledge Interchange Format (KIF)) which acknowledges “Object” and “Process” as physical concepts, and (3) Ontology

Works Upper Level Ontology (ULO) developed by Ontology Works Inc. (Ontology Works Inc., 2009).

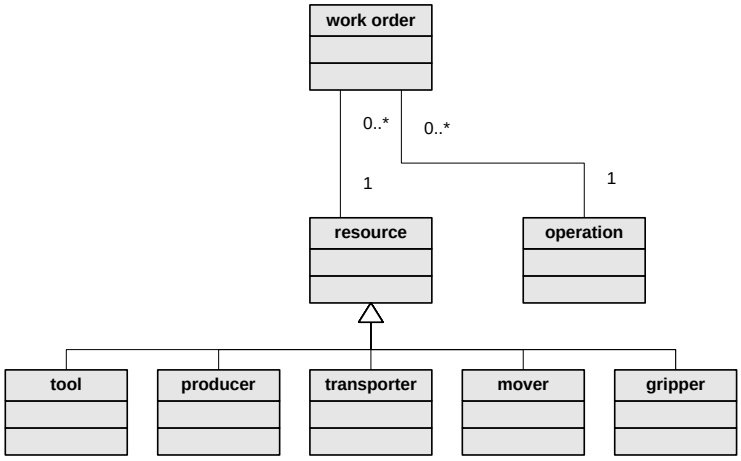


Figure 2-3 Core Manufacturing Ontology in the ADACOR Architecture (Redrawn from Borgo and Leitão (2008))

Figure 2-4 depicts the three main concepts in Ontology Works ULO taxonomy which are "Particular" **(A)**, "SystemEntity" **(B)** and "Universal" **(C)**. These concepts are defined as: (1) particulars are unique things as long as no other thing is the same as them, i.e. particulars are only identical with themselves, (2) system entities are the entities upon which the operation of Ontology Works ontological environment depends on and (3) universals are things that are allowed to have extents i.e. instances (individuals).

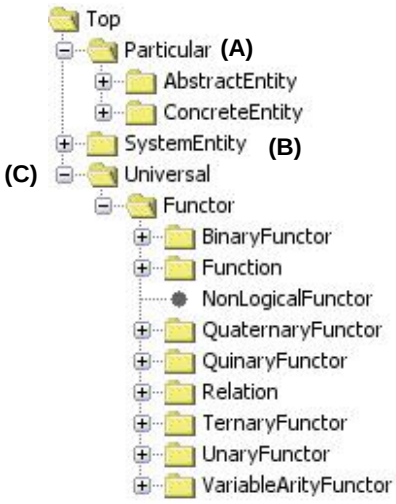


Figure 2-4 Taxonomy of Basic Concepts for Ontology Works ULO (Captured from the ontology environment of Ontology Works Inc. (2009))

In addition to the previously identified foundation ontologies, the value of the Process Specification Language (ISO 18629, 2005) as providing an effective foundation for capturing process-related meaning has also been mentioned (Young *et al*, 2007). PSL, as a foundation ontology, does not fall under the same category as the BFO, DOLCE or Ontology Works ULO. However, because the semantics captured in PSL provide a robust foundation for building explicit conceptualisations for processes of various sorts, this implies that PSL acts as a foundation ontology which supports an interlingua approach to interoperability (Gruninger and Kopena, 2005). This observation is particularly pertinent since PSL has shown benefits to a wide range of work such as (1) for project scheduling information exchange (Cheng *et al*, 2003), (2) for the support of process interoperation in cross-disciplinary supply chains (Das *et al*, 2007) and (3) for capturing the semantics of flow models and process planning knowledge (Bock and Gruninger, 2005).

There is a general view, as far as foundation ontologies are concerned, that they should provide the core semantics of endurants (objects/entities) and perdurants (processes). By understanding relevant work in the field of foundation ontologies, a major question emerges. This question reflects the ongoing concern of how effective foundation ontology approaches can be tailored to support the communication requirements of manufacturing (Young *et al*, 2007). It is clear that this is an important research direction which still deserves attention, especially to facilitate the reuse of the semantics of endurants to model product representations and those of perdurants to model manufacturing processes.

2.3.5 Ontologies in Manufacturing Engineering

A significant amount of work has been performed in the field of manufacturing engineering, where the concept of ontologies has been applied in order to solve specific problems. The area of supply chain management and enterprise engineering, for instance, has witnessed the benefits of ontological engineering (Gruninger and Fox, 1994; Chandra and Kamrani, 2003; Loss *et al*, 2005).

Other researchers have developed ontologies to aid decision support in product design and manufacture. One such example can be seen in work performed by Seo *et al* (2006) who have researched a methodology for achieving interoperable product data through the use of a layered reference ontology. Lin and Harding (2007) have defined a Manufacturing System Engineering (MSE) ontology model that has the capability to enable communication and information exchanges between inter-enterprise, multi-disciplinary engineering design teams.

On similar lines, ontologies for product representation have been pursued. One example is portrayed in the research approach taken by Patil *et al* (2005), where an ontology formalised in Description Logics (DL) has been exploited for capturing and representing the semantics of product representations. Formal concept definitions are captured using DL axioms, which to some extent have enabled the capability for semantic data interchange, i.e. semantic interoperability. Another example appears in the work performed by Costa *et al* (2007), where a refinement of the ISO 10303 AP236 standard, for supporting information exchange for the furniture industry, is proposed using a product ontology.

More competitive methods for capturing semantics while helping decision support in product design and manufacture have been researched. A combination of the Web Ontology Language (OWL) with the Semantic Web Rule Language (SWRL) has recently been employed for this purpose (Kim *et al*, 2006; Rabe and Gocev, 2008; Yang *et al*, 2008; Chang and Terpenny, 2009; Wei *et al*, 2009). SWRL rules provide a relatively powerful axiom layer that interacts with OWL-based ontologies for semantic enrichment. For example, in their work Kim *et al* (2006) have specified the constraints and inferences, that hold over the semantics of concepts arising in assembly design, using SWRL rules. Rabe and Gocev (2008), on the other hand, have illustrated that a similar principle would also work in a framework where SWRL rules help generate knowledge within manufacturing domains.

Other related research efforts (Fiorentini *et al*, 2007; Chen and Stuckenschmidt, 2008), also exploiting ontology-based approaches, have culminated in contributions with striking similarities to the ones already identified in this section. The main finding is that most of these contributions tend to concentrate on DL and sometimes OWL with SWRL. However, because DL and SWRL do not provide full coverage for more expressive First Order semantics, this shows that there is still room for improvement in terms of exploring new methods for semantic representation and interoperability. This work targets this niche for the purpose of probing deeper into this aspect.

2.3.6 Ontology Mapping

The continuing diversity of ontologies is partly related to ontologies being aligned with particular views of the world, resulting in biases and subjective features (Hameed *et al*, 2004). Ontology heterogeneity in design and manufacture also occurs as a result of interspersed knowledge at different stages of the product lifecycle. The examples of ontologies discussed in the previous section reveals this ongoing semantic heterogeneity. If these ontological models are to semantically interoperate, methods need to be devised to reconcile disparate ontologies.

The area of ontology mapping has been a key direction to tackle semantic heterogeneity issues across ontologies, with the intention of promoting semantic interoperability. Several overlapping views over categories of ontology mapping methods have been suggested (Kalfoglou and Schorlemmer, 2003; Noy and Musen, 2003; Euzenat and Shvaiko, 2007; Liping *et al*, 2007). There is almost general consensus over the types of methods that can be applied in ontology mapping. Figure 2-5, partly adapted from Noy and Musen (2003), identifies and summarises these methods.

Ontology mapping methods include (1) techniques that focus on combining (merging) two ontologies to construct a new ontology from the individual ontologies, (2) tools that compile a transformation function that transforms a given ontology into another based on the transformation rules specified (Noy

and Musen, 2003), (3) methods that concentrate on establishing a collection of binary relations between the vocabularies of two ontologies (alignment) (Kalfoglou and Schorlemmer, 2003) and (4) methodologies that enable specific portions of two ontologies to be reconciled, through the definition of mappings via an intermediate articulation ontology. It is to be noted that although some researched ontology mapping methods fit very well into this category, others occur as hybrids of the common ontology mapping methods identified in Figure 2-5.

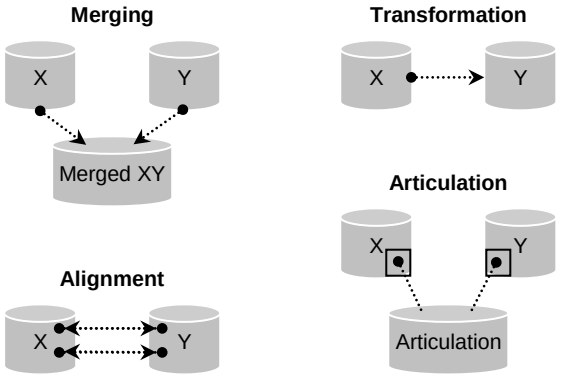


Figure 2-5 Common Methods Used for Ontology Mapping (Based on Noy and Musen (2003))

Comprehensive available literature reviews on ontology mapping and the related methods (Kalfoglou and Schorlemmer, 2003; Euzenat and Shvaiko, 2007) point to a large number of ontology mapping tools that have been either theoretically proposed or fully implemented and tested (Kent, 2000; McGuinness *et al*, 2000; Maedche and Staab, 2000; Kiryakov *et al*, 2001b; Stumme and Maedche, 2001a; Kalfoglou and Schorlemmer, 2002; Madhavan *et al*, 2002; Noy and Musen, 2003; Bach *et al*, 2004; Euzenat and Valtchev, 2004; Mitra *et al*, 2004). In the literature review exposed in this work, only the most outstanding and pertinent ontology mapping methods are documented.

The ontology MApping FRAmework (MAFRA) developed by Maedche and Staab (2000) is an ontology mapping method used for the reconciliation of distributed ontologies on the Semantic Web. MAFRA consists of five horizontal dimensions which relate to the implementation structural aspects of MAFRA and four vertical dimensions which focus on the more strategic

perspectives pertaining to the framework (see Figure 2-6). Following the MAFRA approach, the first step in ontology mapping is that of **(D)** lift and normalisation where all information to be mapped are set onto the same RDF(S) representation platform. Lexical similarities are analysed in stage **(E)** and, then, based on the similarities found between the source and target ontologies, the "Semantic Bridging" module **(F)** establishes necessary correspondences (Kalfoglou and Schorlemmer, 2003). These semantic bridges are then executed **(G)**, verified and enhanced in the final stage **(H)**.

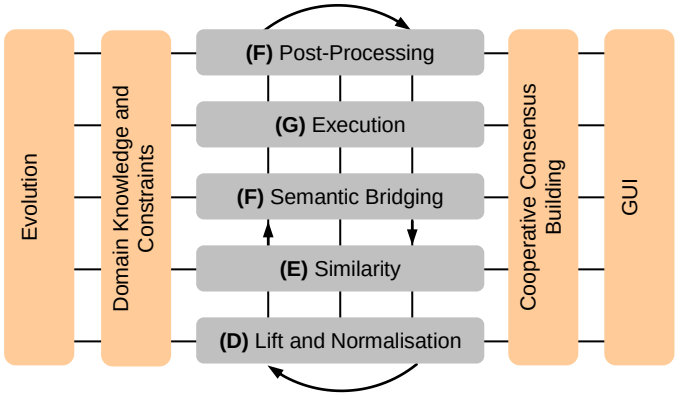


Figure 2-6 Conceptual Architecture of MAFRA (Redrawn from Maedche *et al* (2002))

The FCA-Merge (see Figure 2-7), presented by Stumme and Maedche (2001a), is another important ontology merging environment. Unlike similar ontology merging tools which tend to exclude instances during semantic reconciliation, it is said that FCA-Merge in fact extracts meaningful information from classified instances. The merging process realised in FCA-Merge comprises three vital steps. The first consists of the extraction of instances and the computation of two formal contexts where the ontologies reside. An information extraction technique known as SMES **(I)** (Saarbrucken Message Extraction System) (Neumann *et al*, 1997) is used for this purpose.

The fundamental infrastructure underneath the second phase of the mapping process is the generation of a single context and the computation of the pruned concept lattice **(J)**. This is performed using the FCA-Merge algorithm, known as "Titanic" (Stumme *et al*, 2000), which is attuned to fit the needs of the FCA-Merge environment. Both the first and the second stages are claimed

to be fully automatic processes. The third stage, which is semi-automatic, involves an interactive user interface built on top of the OntoEdit tool **(K)**. In order to support the knowledge engineer in the different steps, there is a number of queries for focusing his attention to the significant parts of the pruned concept lattice (Stumme and Maedche, 2001b).

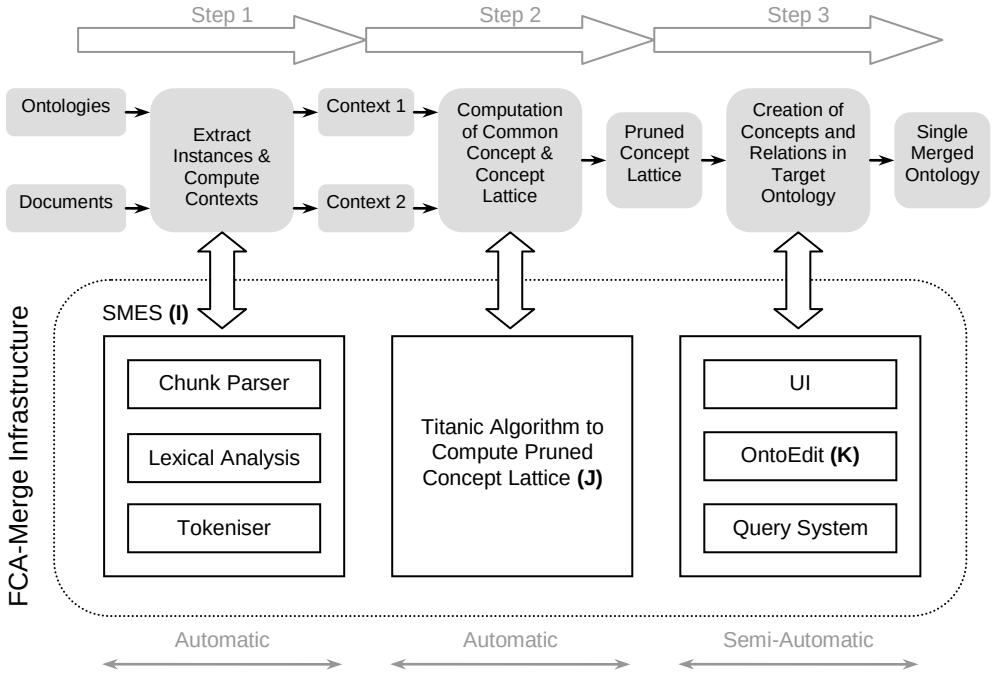


Figure 2-7 FCA-Merge Interaction Environment (Based on Stumme and Maedche (2001b))

Noy and Musen (2000) initially proposed an algorithm and tool to promote ontology merging and alignment. The authors have later exposed a complete suite of tools integrated in the "Prompt" suite (Noy and Musen, 2003), covering various functionalities for multiple-ontology management. The "Prompt" suite comprises (1) "IPrompt" for interactive ontology merging, (2) "AnchorPrompt" for graph-based mapping, (3) "PromptDiff" for ontology versioning management and (4) "PromptFactor" for factorising out semantically independent sub-ontologies.

"IPrompt", which forms part of the algorithm-driven semi-automatic ontology merging feature of "Prompt", is responsible for providing suggestions for merging, identifying inconsistencies, resolving potential problems and exposing strategies to solve these (Noy and Musen, 2003). During the

comparison of two ontologies, “IPrompt” analyses small segments of the ontology graph around each concept prior to proposing appropriate merging decisions. Overall, the “Prompt” suite remains a comprehensive semi-automatic toolkit for coping with semantic reconciliation.

Researched and validated ontology mapping tools indicate that there is currently no ontology matching technique that uses the semantics of logic-based systems that employ upper ontologies (Euzenat and Shvaiko, 2007). Moreover, it is evident, from experiments based on current ontology mapping methods, that ontology mapping has not been given due attention in design and manufacture primarily since the latter remains an expert domain with very specific content and issues (Chungoora and Young, 2008b). Hence, this work additionally addresses the relevance of semantic-based mapping methods for aiding semantic interoperability in product design and manufacture.

2.4 Model Driven Architecture and Interoperability

The Model Driven Architecture (MDA) is an initiative launched by the Model Driven Software Development (MDSD) community, and is nowadays a recommended specification from the Object Management Group (OMG) (Bourey, 2007). The MDA approach typically consists of a number of basic concepts, as defined in the MDA Guide (2003). These concepts involve three viewpoints and system models notably (1) the Computation Independent Model (CIM), (2) the Platform Independent Model (PIM) and (3) the Platform Specific Model (PSM), whose interactions consist of model transformations for converting one model to another on the same system. These basic concepts of MDA are reflected in Figure 2-8, together with the identification of model transformations between the CIM, PIM and PSM.

For a single system solution under development, the high-level requirements for the system are first set and modelled in a CIM, in order to identify the intended expectations of the system. In other words, the CIM describes the business context and business requirements for the system under consideration, corresponding to a view defined by a computation independent

viewpoint (Elvesæter *et al*, 2006). The PIM, on the other hand, defines a model at a high level of abstraction, where the model is used to describe the software solution using a technology independent view (Bourey, 2007). It is possible through transformation mechanisms to convert a single PIM into one or several PSMs as shown in Figure 2-8. A PSM corresponds to a view defined by a platform specific viewpoint and describes the realisation of software systems in the chosen set of execution platforms (Elvesæter *et al*, 2006).

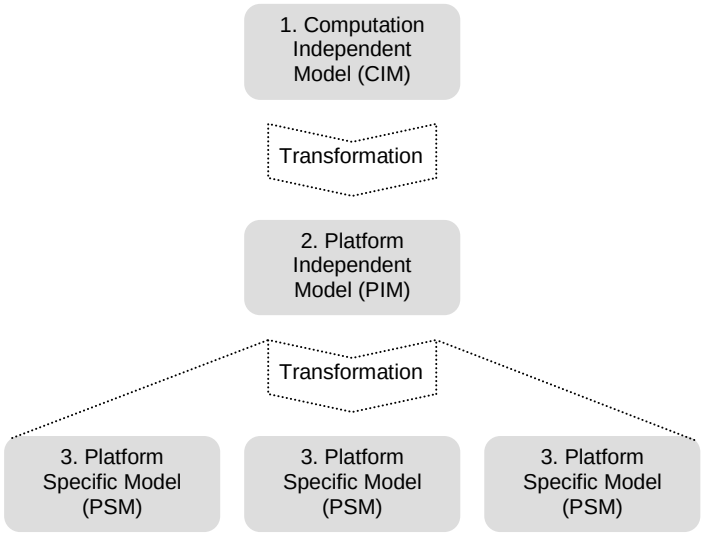


Figure 2-8 Basic Concepts in the Model Driven Architecture (MDA)

The principle of applying MDA to interoperability, referred to as Model Driven Interoperability (MDI), is an interesting direction as several researchers have utilised MDA and MDI to solve specific problems attuned to distinct fields of research (Cutting-Decelle *et al*, 2006; Elvesæter *et al*, 2006; Gnägi *et al*, 2006; Didonet del Fabro, 2008; Moalla *et al*, 2008).

Figure 2-9, which is based on the reference model identified by Bourey (2007), portrays a simplified version of the reference model used for MDI. In the reference model, two MDA approaches are shown to have been applied separately for developing two system solutions for “Enterprise E1” and “Enterprise E2”. Model transformations are present between the CIM, PIM and PSM levels within each enterprise system. The capability for interoperation between the different MDA levels across enterprise boundaries is anchored through the definition of intermediate interoperability models that

support (1) transformations and mappings between each cross-enterprise MDA level and (2) transformations between interoperability models too.

Existing work on MDI points to the fact that MDA approaches have been used for exploring solutions related to interoperability and semantics. Moalla *et al* (2008), for instance, have documented the mode in which the deployment of MDI contributes to an enhancement in product data quality across the vaccine supply chain. Other authors like Gnägi *et al* (2006) have looked at promoting semantic interoperability between Object-Oriented models through the use of MDA.

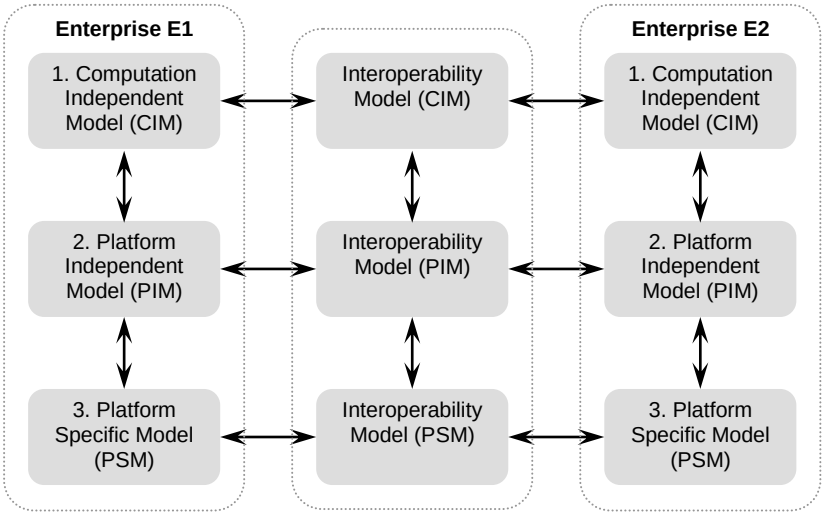


Figure 2-9 Simplified Version of the Reference Model for MDI (Based on Bourey (2007))

Bourey *et al* (2006), for example, have refined the current knowledge on models and transformations and have applied them to test cases within the INTEROP NoE project (Panetto *et al*, 2004). In these experiments, a meta-model approach is first defined for enabling transformations. Mappings, implemented in a suitable transformation language, are then established between the elements of the defined meta-models and executed to complete the transformation process. From the breadth of work performed in the field of MDA and MDI, it becomes obvious that there is an acknowledged importance relating these approaches to interoperability and semantics. Another purpose of this work, hence, is to develop novel concepts whose underlying understanding can also be positioned according to MDA and MDI.

2.5 Standards-Based Approaches to Interoperability

In addition to the previously exposed paradigms, contributions are also being pursued towards the development of international standards which would promote interoperability, for example, technical standards for product information and CAD/CAM documents realised by efforts like Product Data Management (PDM), Product Lifecycle Management (PLM) and STEP (Lin and Harding, 2007).

Particularly relevant to the field of product design and manufacture is the ISO 10303 standard, also referred to as STEP (STandard for the Exchange of Product model data). STEP is aimed at the standardisation of product data for exchange. The specifics of STEP and its implications on data management, exchange and sharing, i.e. its implications on interoperability, have long been recognised (Fowler, 1996). Furthermore, it has been demonstrated how the various STEP Application Protocols (APs), defined predominantly around the concept of “machining features”, can be harnessed to achieve an integrated manufacturing architecture (SCRA, 2006).

Figure 2-10, adapted from SCRA (2006), portrays this interoperability-enabled architecture, where some of the STEP APs are shown to relate to specific functions in design and manufacture. The total architecture enables the deployment of an integrated manufacturing environment where machining features are present at the core of the information exchange capability. Other similar efforts towards standardisation have been fostered (TC184/SC4 Website, 2009) such as (1) Parts Library (PLIB) (ISO 13584) for the representation of parts library data to support interoperability between suppliers and users, (2) manufacturing management data interchange (MANDATE) (ISO 15531) for the representation of production process data and (3) the Process Specification Language (PSL) (ISO 18629) for the semantic definition of manufacturing processes.

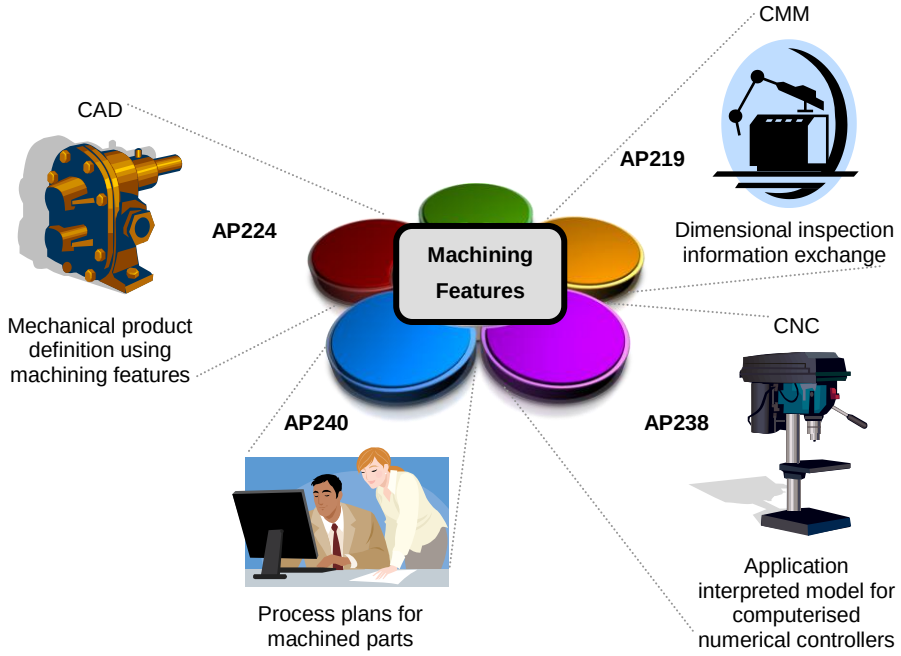


Figure 2-10 Enabling Manufacturing Integration Using STEP Application Protocols and Machining Features

Although standards-based approaches provide a viable direction to resolving interoperability issues, only few of these actually overcome the semantic interoperability challenge. This is because even concepts which are supposed to have agreed definitions within Standards, do not necessarily share the same semantics. For example, Young *et al* (2007) have shown the inconsistencies present in the informal semantics of the word “process” in ISO 19493, ISO 18629 and ISO 10303. This observation is also shared by Costa *et al* (2007), who have pinpointed the presence of obstacles related to the fuzziness in ISO 10303 AP236 definitions.

It has to be noted, however, that the concepts defined in PSL (ISO 18629) remain robust, from a semantic integrity viewpoint. This is because, PSL is aimed at capturing heavyweight semantics specifically, unlike other standards like STEP, which remains lightweight in nature and does not satisfy all the requirements for semantic interoperability (Patil *et al*, 2005). In addition to acknowledging the semantic interoperability limitations of STEP, this review also depicts a clear potential to address these issues by exploiting heavyweight ontological approaches to formalise relevant parts of ISO standards.

2.6 Information Modelling in Product Design and Manufacture

The modelling of information and knowledge structures in product design and manufacture has a direct influence on the capability to semantically interoperate. This is because, the degree of formality present in the structuring of information in a model is analogous to the semantic enrichment of the captured model. In PLM, two significant types of models have been pursued namely (1) product models (Molina *et al*, 1995; Anderl, 1997; Balogun *et al*, 2004; Sudarsan *et al*, 2005) and (2) manufacturing models (Giachetti, 1999; Zhao *et al*, 1999; Al-Ashaab *et al*, 2003; Liu and Young, 2004).

2.6.1 Product Models

A product model may be defined as an information model, which stores information related to a specific product (Molina *et al*, 1995; Anderl, 1997). Another analogous description of a product model has been provided by Balogun *et al* (2004), who specify that the model represents a complex product from the top product level to the tolerance detail of every feature characteristic.

Product models occupy a key role at the centre of the product lifecycle (Young *et al*, 2007) since they hold and share product information that are generated, used and maintained over the processes of design, manufacture, delivery, maintenance and disposal (Lee *et al*, 2006). Product models may be composed of a number of sub-models such as (1) the structure-oriented, (2) geometry-oriented, (3) feature-oriented and (4) the knowledge-oriented models, which when unified into integrated product models (Chin *et al*, 2002) enable decision support capability to be achieved.

The concept of product models continues to evolve with time. Sudarsan *et al* (2005), for example, have successfully exploited a particularly interesting product model, known as the Core Product Model (CPM) as shown in Figure 2-11. The main advantage of the CPM is that it favours product model

extensions while providing a common ground. The model proposed by Sudarsan *et al* (2005) also aims at capturing different engineering contexts that involve view-specific product considerations. The “Product Family Evolution Model” (PFEM), for instance, represents the evolution of product families and the rationale of the changes involved (Wang *et al*, 2003).

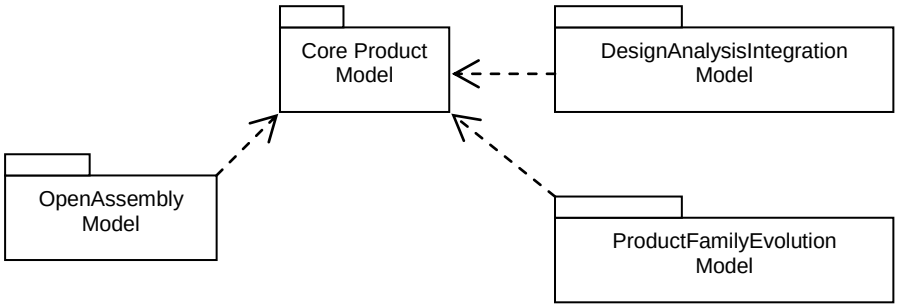


Figure 2-11 Framework Components of the Core Product Model (Redrawn from Sudarsan *et al* (2005))

2.6.2 Manufacturing Models

The concept of manufacturing models initially took root from contributions made by Al-Ashaab (1994). Manufacturing models consist of common repositories of manufacturing capability information and the knowledge and constraints over the use of manufacturing processes (Al-Ashaab, 1994; Balogun *et al*, 2004; Liu and Young, 2004). The information structures exploited for this purpose comprise of defined relationships between all manufacturing capability elements.

Similar to how product models can be decomposed into their constituent individual sub-models, manufacturing models also enfold different concepts like (1) the manufacturing resource capability model, which concentrates on representing information about functions and characteristics of manufacturing resources and their combination into manufacturing processes (Giachetti, 1999; Molina *et al*, 1995; Zhao *et al*, 1999), (2) the process plan model, used to describe the information about the process plan strategy of a manufacturing process (Feng and Song, 2003) and (3) the manufacturing cost model, used for driving the meaningful estimation of production costs incurred during design and manufacture.

In their work, for example, Feng and Song (2003) have met the aim of developing a “Manufacturing Object Model” to enable the interoperability of preliminary design with process planning. Their implementation platform utilises the Unified Modelling Language (UML) Object-Oriented (OO) approach for constructing the information structures behind the manufacturing model. Current documentation on manufacturing models (Tam *et al*, 2000; Liu, 2004; Gunendran and Young, 2006) further point to the fact that mostly an Object-Oriented slant has been given as far as information modelling of manufacturing models are concerned, i.e. exploited information structures have remained lightweight in nature.

2.6.3 Integrating Product and Manufacturing Models

Clear evidence is available which demonstrates that there is a need to integrate the product and manufacturing models. Feng and Song (2003), for instance, mention that both models have not been shown fully integrated with each other. The integration of product and manufacturing models is key towards reinforcing decision support capability and knowledge acquisition in the product development lifecycle.

The ability to capture and reuse design and manufacturing knowledge in a meaningful manner is dependent on the semantic interoperability of product and manufacturing models. Gunendran and Young (2006), for example, have documented an information and knowledge framework for capturing multi-perspective design and manufacture and have mentioned that the integration knowledge may contain several rules, equations and options to support the information integration of multiple views. However, multi-view modelling to acquire manufacturing knowledge has been developed into solutions based on the use of UML, and therefore use a lightweight ontological approach which is inappropriate for inter-system interoperability (Young *et al*, 2007). Hence, it is clear that a progression to achieve this semantic integration remains to be addressed.

2.6.4 Features and Part Families

Feature-based engineering bridges the gap between Computer Aided Design (CAD) and Knowledge Based Engineering (KBE) systems (Shah, 1995; Otto, 2001). A useful definition for a feature has been provided by Brunetti and Golob (2000), who mention that a feature is an information unit (element) representing a region of interest within a product, and is described by an aggregation of properties of a product.

Several authors have documented the importance of features of various sorts as providing valuable integration links between design and manufacture, such as the “machining features” effort from STEP. Gu (1994), for example, have recognised the significance of feature-based representation, as part of a product models for supporting integrated manufacturing. The ongoing significance of feature-based modelling is well established and has been under consideration by several researchers at different periods of time such as Young and Bell (1993) and Aifaoui *et al* (2006).

One of the recent types of feature that has emerged, with the scope of representing any geometric and non-geometric relations in an assembly, involves associative assembly design features (Ma *et al*, 2007). In their approach, Ma *et al* (2007) firstly identify the requirements for satisfying assembly features by specifying, for example, (1) the need for independent representation of feature relations and (2) the representation of relationships between features and parts for the inclusion of both geometric and non-geometric information. However, it is to be noted that a lightweight ontological approach using UML modelling has been pursued.

Feature technology follows two main paradigms namely that of (1) feature recognition and (2) design by feature. In the former, intelligent algorithms are used to extract features from existing geometry. However, a major limitation is present on this approach and relates to the effectiveness of the exploited algorithms to recognise interacting features (Martino and Giannini, 1998). In the design by feature approach, which is nowadays favoured compared to

feature recognition, a product can be modelled from a library of available features. There is, however, a drawback to this approach in that the representation of features is dependent on the context, i.e. viewpoint, being taken (Martino and Giannini, 1998). Nevertheless, where features can be understood within a part family context, there is the potential for them to provide a significant route to sharing information between lifecycle activities (Gunendran and Young, 2008), i.e. the semantics of part families can help support interoperability in product design and manufacture.

The concept of part families, in which specific parts are grouped according to their manufacturing operation requirements, is particularly relevant to group technology and cellular manufacturing systems (Ang, 1998; Chan at al, 2006; Yang and Yang, 2008). Categorisation of part families with respect to specific viewpoints arising in design and manufacture, as is the case with features, is also a fact, for example, design, manufacturing and assembly part families (Westkämper *et al*, 2000; Simpson, 2004; Jiao *et al*, 2007; Gunendran and Young, 2008).

It has been acknowledged by Li *et al* (2006), whose work is concerned with the representation and sharing of part feature information in Web-based parts library, that one of the requirements to achieve meaningful part family description is to have a comprehensive norm for capturing part family information. This, from a semantic interoperability perspective, additionally implies the importance of addressing semantic descriptions of features and part families, as well as the ability to wrap semantically-rich product and manufacturing models.

2.7 Interoperability Architectures and Frameworks

Wide-ranging interoperability architectures and frameworks have been proposed to date. A comprehensive review of some of these has been documented by Chen *et al* (2008) and this section, therefore, concentrates on a discussion of the most pertinent interoperability architectures and frameworks relevant to this work. Early efforts fostered have resulted in well-

established reference architectures such as (1) the Computer Integrated Manufacturing Open System Architecture (CIMOSA) (AMICE, 1993), (2) the Purdue Enterprise Reference Architecture (PERA) (Williams, 1994), (3) the GRAI-GIM reference model (Chen and Doumeingts, 1996) and (4) the Reference Model of Open Distributed Processing (RM-ODP) (ISO/IEC 10746, 1996).

With the evolving view on interoperability at enterprise level, a majority of interoperability architectures and frameworks are being established according to the strategic principles related to the requirements for business interoperability, considerations for appropriate technological support and the chosen architecture perspective. The Zachman Framework (The Zachman Framework Website, 2009), IDEAS interoperability framework (Chen *et al*, 2004) and The Open Group Architecture Framework (TOGAF) (TOGAF Website, 2009), for example, all identify significant multi-level prerequisites for enabling enterprise interoperability.

In the IDEAS interoperability framework, which has been developed within the ATHENA project (Ruggaber, 2006), a specific dimension is acknowledged for the implications of semantics cutting across the "business", "knowledge" and "Information and Communication Technology" (ICT) levels within single enterprises and the need for integrating, unifying and federating across enterprise boundaries. This understanding is portrayed in the simplified IDEAS interoperability framework in Figure 2-12.

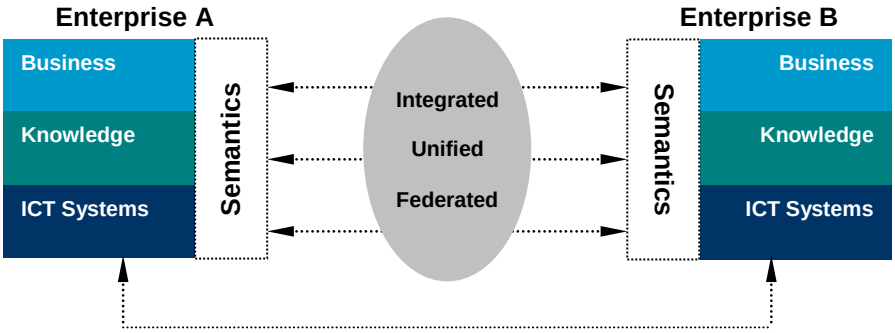


Figure 2-12 IDEAS Interoperability Framework (Redrawn from Chen *et al* (2004))

In the context of international standards, a multi-dimensional framework has been proposed for enterprise interoperability (CEN/ISO 11354, 2008). The first elaborated part of the framework entails the requirements for enabling process interoperability across manufacturing enterprises. Figure 2-13, adapted from CEN/ISO 11354 (2008) illustrates the Framework for Enterprise Interoperability. There exist three dimensions to the framework notably (1) the barriers to interoperability such as conceptual and technological, (2) relevant concerns such as business and process and (3) the approaches to interoperability such as federated and unified. In Figure 2-13, PSL has been positioned according to the Framework for Enterprise Interoperability, and it can be seen that the “conceptual”, “process” and “unified” dimensions help position PSL in the right segment of the framework matrix.

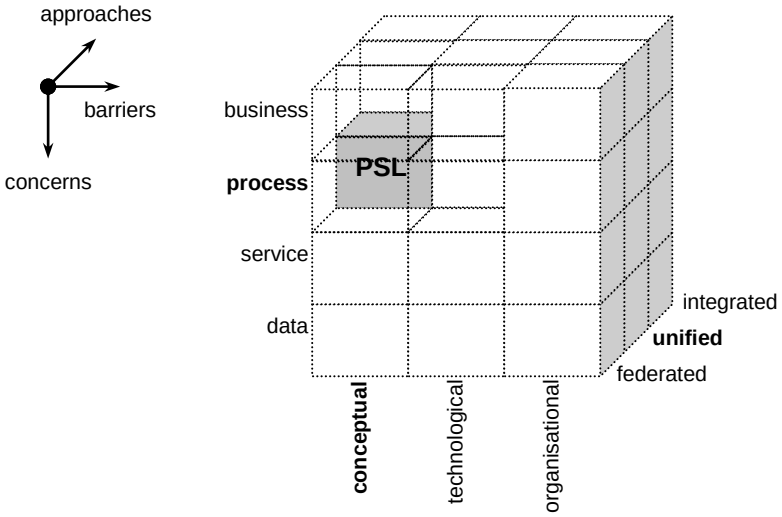


Figure 2-13 Positioning PSL in the Framework for Enterprise Interoperability (Redrawn from CEN/ISO 11354 (2008))

Other architectures, such as the semantic-mediation architecture for business-to-business interoperability (Vujasinovic *et al*, 2007), have also been researched and industrially validated. In their work, Vujasinovic *et al* (2007) have implemented their architecture within the ATHENA research project (Ruggaber, 2006). Their implementation platform primarily harnesses Semantic Web tools with XML and RDF(S) capability. Vetere and Lenzerini (2005), on the other hand, have identified four different types of models for semantic interoperability in service-oriented architectures, by following an understanding of centralised and decentralised mappings between service

schemas. However, this work has remained at a conceptual level since no test case implementation is proposed.

In current literature, very few contributions have coined the terms “semantic interoperability framework”. Amidst these contributions lies the extended COntext INterchange (eCOIN) framework (Firat *et al*, 2007), whose main purpose is to facilitate semantic reconciliation through the definition of reusable “conversion function networks” as mappings. The authors of eCOIN adopt a view that the achievement of semantic interoperability should take account of semantic heterogeneity as well as semantic reconciliation. It has been argued that the eCOIN uses a hybrid of ontology-based methods involving principles like ontology alignment through articulation axioms and ontology merging (Firat *et al*, 2007). However, the motivational scenarios that back up eCOIN remain broad in nature and have not been attuned to the world of product design and manufacture.

Specifically in the field of product design and manufacture, relatively few frameworks have been proposed in order to contribute to semantic interoperability. Patil *et al* (2005), for instance, have presented an approach to foster the semantic interoperability of product data utilising an ontology-based framework. This framework for semantic interoperability is identified in Figure 2-14.

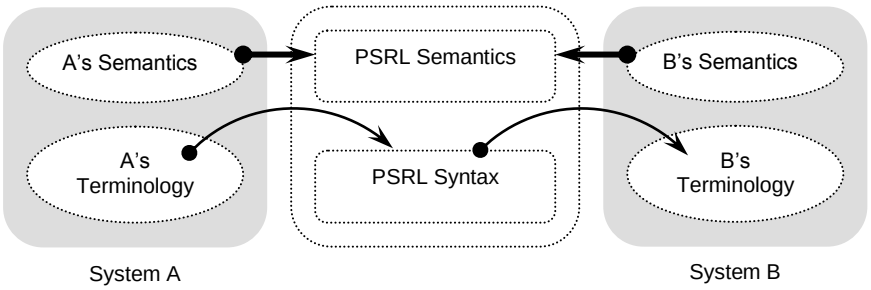


Figure 2-14 Framework for Semantic Interoperability by Patil *et al* (2005)

Following the framework diagram proposed by Patil *et al* (2005), it is possible to identify two main reconciliation mechanisms present namely (1) the mapping of the semantics from a “System A” and “System B” into an intermediate product ontology (Product Semantic Representation Language (PSRL) which is DL-based) and (2) the translation of syntax and terminology

from “System A” to syntax in PSRL, which is then translated to the syntax of the target “System B”. It is to be noted that Patil *et al* (2005) have recognised that their approach does not support low levels of abstraction in product models, such as geometric entities, as a result of their preference for the domain of Description Logics.

Gupta and Gurumoorthy (2008) have argued a feature-based framework to support semantic interoperability of product models. The concept of “Domain Independent Form Feature” (DIFF) has been proposed, over which the framework is constructed. Figure 2-15 illustrates their schematic concept which enables semantic interoperability of product models. In the figure, the DIFF model supported by an ontology, provides a basis for the representation of features, and facilitates semantic interoperability between a source and a target system.

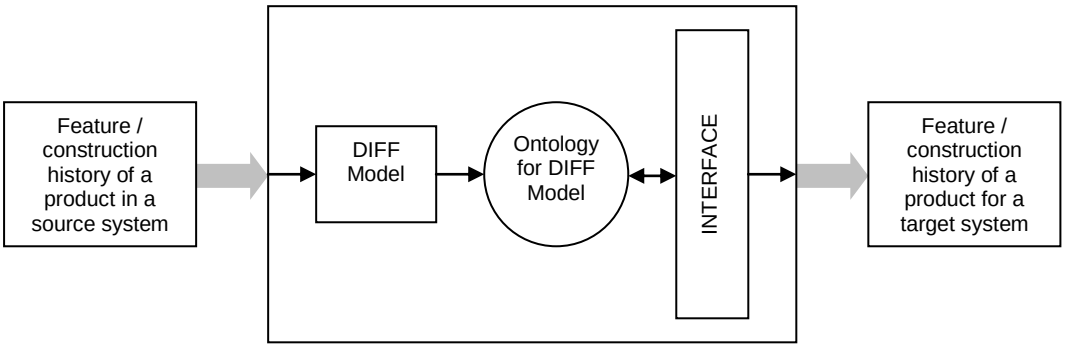


Figure 2-15 Gupta and Gurumoorthy's (2008) Approach to Semantic Interoperability of Product Model

In their approach, Gupta and Gurumoorthy (2008) have focused on the definition of features in terms of their faces solely, and have looked exclusively at semantic interoperability problems occurring due to different labels that refer to the same shape and different representations for the same shape. This implies that other significant considerations for (1) feature function in design, (2) relationships between features and manufacturing processes and (3) other forms of semantic interoperability issues remain to be addressed. Furthermore, the authors have implemented their framework utilising the Protégé (Protégé Website, 2009) ontological environment. Since Protégé does not provide full support for First Order heavyweight semantics,

this implies that opportunities still exist for improving the expressiveness of semantics in product models.

2.8 Summary

This state-of-the-art review has been conducted with a outlook onto the most pertinent areas of knowledge relevant to the problem of achieving interoperability at the semantic level, where the interoperation has to be established by the supply of information through inter- and intra- system communication (Chen *et al*, 2008). Five key areas have thus been targeted namely (1) ontology-based approaches to interoperability, (2) the Model Driven Architecture, (3) Standards-based methods, (4) the relevance of interoperable information modelling in design and manufacture and (5) current architectures and frameworks that attempt to resolve the problem of interoperability and semantics.

Ontology-based methods have attracted a lot of attention for the development of shared representations. It has been witnessed that the ability for sharing semantics across these representations is dependent on the degree of formality, or logical expressiveness, supported by ontological formalisms. However, it has to be appreciated that even in the deployment of ontology-based methods, semantic heterogeneity is unavoidable and for this reason, methods for ontology mapping are being developed for reconciling the semantics between ontologies that need to interoperate.

The Model Driven Architecture (MDA) also has a significance in shaping the future perspectives on semantic interoperability. This work recognises its influence and, therefore, the MDA approach partly serves as a basis for positioning this research in terms of the CIM, PIM and PSM levels of the architecture. Standards-based methods to interoperability are also particularly important as they corroborate the ability to employ and reinforce useful principles applied in manufacturing integration.

Information modelling in product design and manufacture has been recognised as providing valuable potential for capturing the semantic structures required in product and manufacturing models and their integration. It is seen that this integration can also be facilitated through the consideration of multiple viewpoints of product features and part families. On the other hand, it has been possible to comprehend how all the other previously-mentioned areas of knowledge are reflected in existing interoperability architectures and frameworks.

The gathered understanding from the state-of-the-art documentation has helped identify a number of niches that remain to be fulfilled. These key research gaps are listed below:

- There is a need for improved ontology-based framework solutions to support semantic interoperability and knowledge sharing in design and manufacture.
- There is the ongoing requirement to understand how to exploit effective foundation ontology approaches to meet the communication needs in manufacturing.
- There is a potential for exploiting more formal semantic-based methods for ontology matching.
- It is necessary to explore heavyweight ontological approaches to address the representation of product and process semantics.

The identification of these research gaps meets the first objective of this research (see Chapter 1 section 1.3.1). Overall, it has been shown that there is currently no existing framework that addresses, in a holistic way, the problem of semantic interoperability in product design and manufacture. This work, hence, exposes a novel attempt to support semantic interoperability in product design and manufacture by harnessing relevant capabilities from the identified areas of knowledge.

3 Requirements to Support Semantic Interoperability in Product Design and Manufacture

3.1 Introduction

This chapter elaborates a set of requirements pertinent to supporting semantic interoperability in product design and manufacture. Section 3.2 broadly illustrates the implications of pursuing semantic interoperability in collaborative design and manufacture. Section 3.3 then explores a number of semantic interoperability issues, based on hole features occurring in design and manufacture, from which the related semantic requirements are exposed. These requirements represent a valuable checklist which is closely linked to the development of the preferred concepts adopted this work. A short summary of the investigated requirements is provided in section 3.4.

3.2 Semantic Interoperability in Product Design and Manufacture

Seamless semantic interoperability is achievable when the meaning associated to captured information and knowledge in computational form can be effectively shared across systems without any loss of the meaning and intent of the information and knowledge during the exchange process. At present, unclear, implicit and ambiguous semantics lead to misunderstandings and semantic obstacles i.e. obstacles related to the definitions of business and software classes and organisation of information (Gunendran *et al*, 2007). Figure 3-1 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 3-1, 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. In a semantic interoperability-enabled environment, it is essential that the semantics of various viewpoints be captured. This is further explained in section 3.3.1.

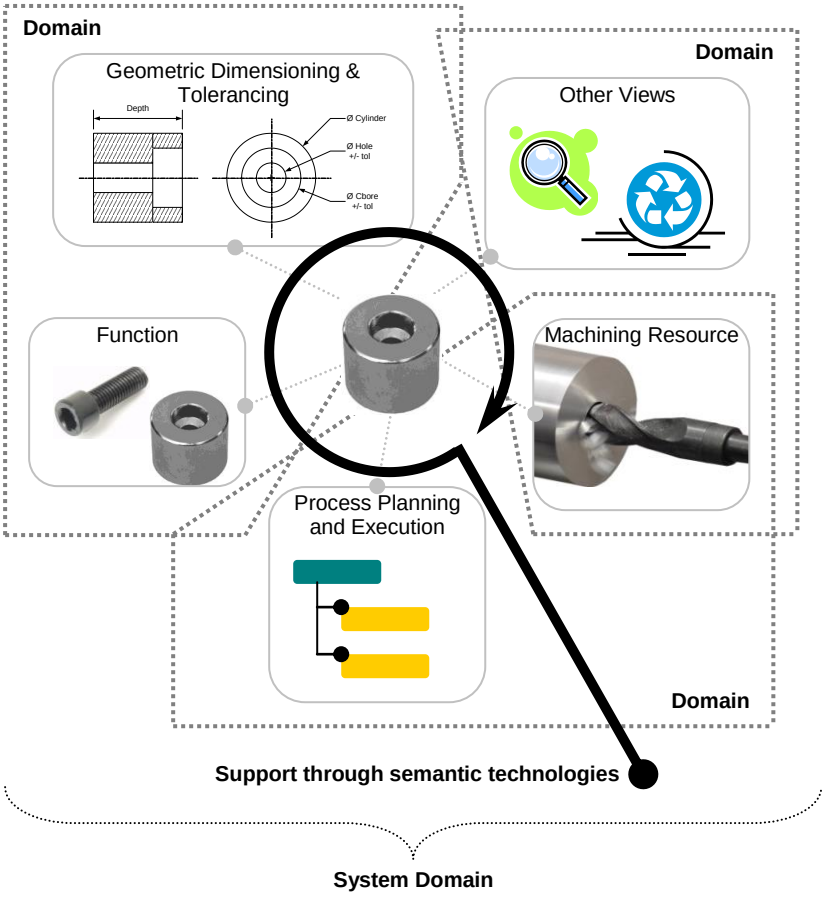


Figure 3-1 Overlapping Viewpoints and Domains in Design and Manufacture within a System Domain

In collaborative product development, intra-system domains need to establish shared interpretations over specific product viewpoints or combinations of viewpoints, as shown in Figure 3-1, in order to facilitate information exchanges. In the context of this work, a domain is regarded as a field of knowledge, based on one or multiple similar viewpoints, required to perform

the task of solving difficult real-life problems through the use of expert system procedures (Kalpakjian, 2001). A system domain, on the other hand, involves multiple interacting domains. Of particular relevance are (1) the means of driving semantic consistency and interoperability across multiple viewpoints within a single system domain and (2) the means of supporting semantic interoperability across system domains. Based on Figure 3-1, within a single system domain, the ability to semantically interoperate between view-specific domain models is dependent on the creation, derivation and extraction of semantic relationships (Ray and Jones, 2006) (see Section 3.3.2). In this work, the terms “domain model” are used to refer to a formal domain conceptualisation (ontology) and its associated Knowledge Base (KB).

In a concurrent engineering-driven arena, different system domains, that hold their own integrated product views, may need to interoperate. From a semantic interoperability standpoint, this raises a concern linked to ensuring the cross-system consistency in the meaning of overlapping concepts that cut across system domain boundaries (refer to section 3.3.3). Acquiring semantic interoperability in product design and manufacture is also dependent on available technological support. In the world of semantic interoperability, semantic technologies provide the capability to address semantic interoperability obstacles between domain models. However, fundamental concerns remain in order to identify better means of harnessing semantic technologies while overcoming the related challenges documented in Shvaiko and Euzenat (2008). Section 3.3.4 explores this is greater detail.

3.3 Semantic Interoperability Issues and Requirements

3.3.1 View-Specific Semantics in Design and Manufacture

Product lifecycle knowledge resides in multiple different but overlapping viewpoints. Approaches such as Design for Function, Design for Assembly, Design for Manufacture and Process Planning dictate the nature of the meaning and intent of concepts defined within specific viewpoints. This diversity of perspectives remains a key issue as far as ensuring semantic

integrity across viewpoints is concerned. Figure 3-2 presents two views, namely a functional view and a machining view, featuring visible semantic differences due to alternative representations of a counterbore hole.

In the functional view, a counterbore hole is considered from a product requirements angle where the purpose of the feature is to accommodate a particular bolt size specification, hence its definition as a bolt hole. In the machining view, the functionality of the counterbore hole is not immediately relevant, and the same feature is defined by a different set of semantics pertinent to the machining view. In the case of the functional view, the attribution of depth parameters to the counterbore hole is based on the bolt head position and bolt length position. In the machining view, the attribution of depth parameters involves viewing the counterbore hole as a compound feature requiring a drilling operation followed by a counterboring operation.

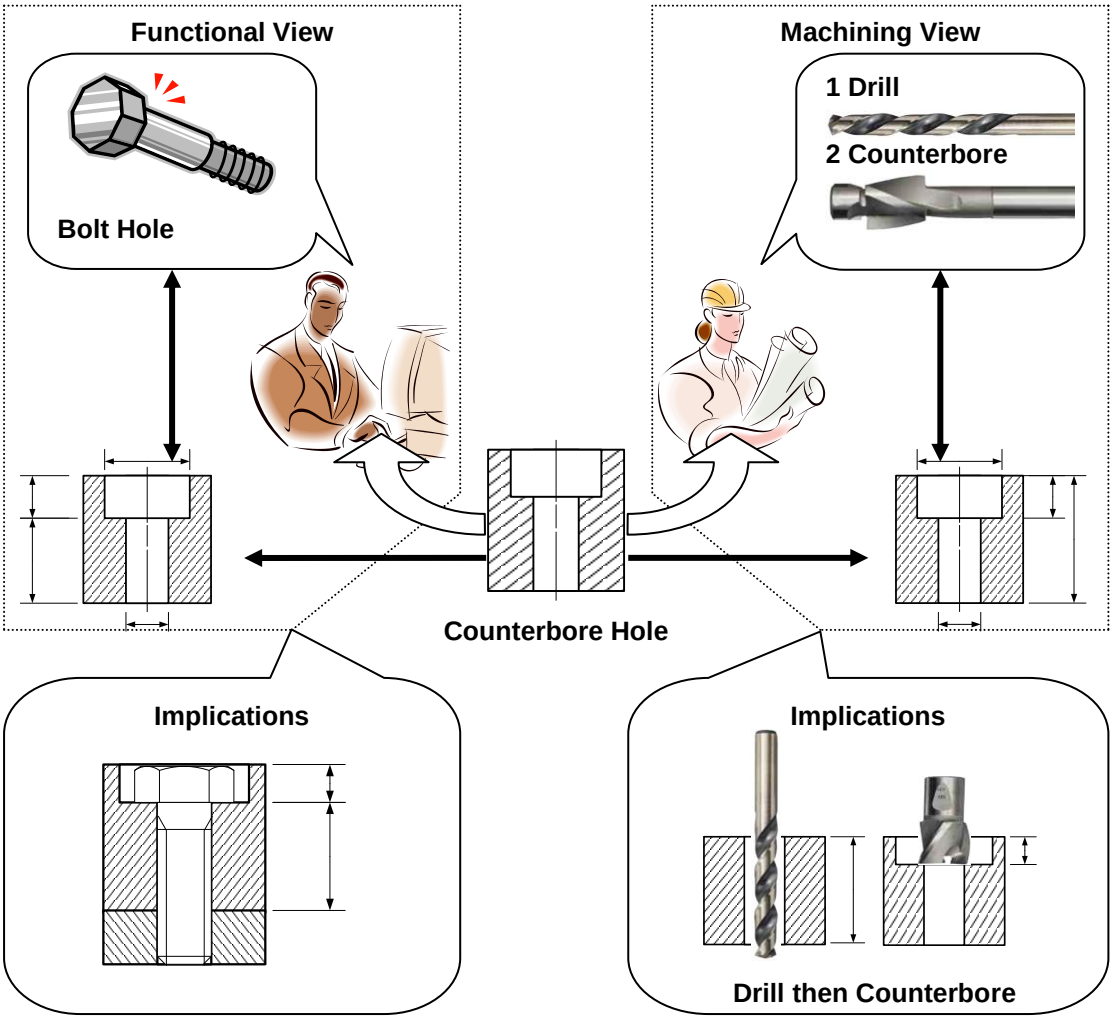


Figure 3-2 Example of a Counterbore Hole Viewed from Two Different Viewpoints

The example noticeably shows that product features may be defined using view-specific semantics which is an important encounter across information modelling in product design and manufacture (Chapter 2 section 2.6). Furthermore, it has been recognised that semantics need to be defined for contexts (viewpoints) such as functional, geometry, manufacturing, machining process and assembly (Gunendran *et al*, 2007) to support interoperability at various stages of the product lifecycle. These multi-perspective considerations are essential for sharing information (Kugathasan and McMahon, 2001; Canciglieri and Young, 2003; Gunendran and Young, 2006). Hence, it becomes evident that a progression towards the seamless exchange of design and manufacturing knowledge requires capturing the semantics of concepts from multiple product lifecycle viewpoints **[Requirement 1]**.

3.3.2 Semantic Relationships between Viewpoints

To capture the interactions between elements from different view-specific semantics, relationships need to be made across viewpoints so that the knowledge contained in one viewpoint can be interpreted in another without any loss of semantics. These relationships could be supported through the definition of ontology-based relations (Chapter 2 section 2.3.1) and via the integration of product and manufacturing model information (Chapter 2 section 2.6.3). The example which follows proposes a scenario where albeit concepts from two different viewpoints occur, there nevertheless exists a possible overlap between the two, from a semantic standpoint.

Figure 3-3 identifies a GD & T (Geometric Dimensioning and Tolerancing) viewpoint where a simple hole feature is described in terms of its nominal diameter and diameter tolerances. From a machining viewpoint, the semantics of the same hole feature take into account the machining processes required to achieve the nominal diameter and diameter tolerances. In the scenario, it can be seen that a semantic relationship between the dimensional parameter “A ± B” and a “Reaming” process, that achieves this dimensional parameter, can be used to drive knowledge of how a “Reamed

Hole" may be produced through a sequence of "Centre Drilling" followed by "Drilling" followed by "Reaming".

The example clearly demonstrates that if overlapping semantics between viewpoints can be understood, then it is possible to obtain a basis for defining semantic relationships. These relationships would apply regardless of domain boundaries developed within single system domains. Hence, there exists a need for providing semantic relationships between different but overlapping product viewpoints in order to support integrated semantic capabilities **[Requirement 2]**.

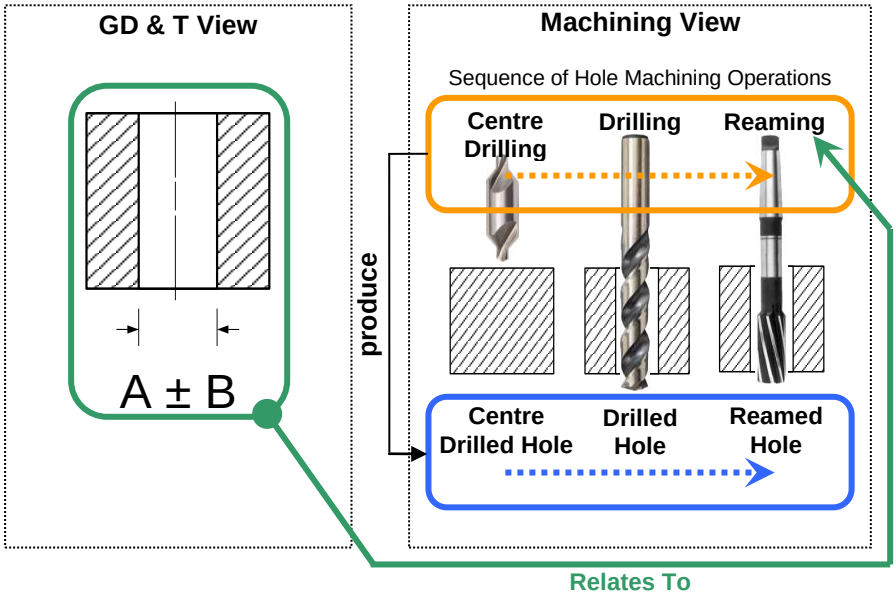


Figure 3-3 Example of a Semantic Relationship between Two Views within a Single Domain

3.3.3 Semantics of Core Concepts across System Domain Boundaries

Several shared domain conceptualisations (domain ontologies) that need to interoperate at the semantic level do not readily do so as a consequence of ontology heterogeneity. Continuing diversity of domain ontologies is partly related to the choices of knowledge representation formalism made, domain preferences and the inappropriateness of enforcing an all-embracing common ontology as a basis over which to build up information exchanges (Hameed *et*

al, 2004). This subsequently leads to multiple ontologies and schemas developed by independent entities (Madhavan *et al*, 2002).

Although multiple domain ontologies impose semantic obstacles during their interoperation, it is obvious that all system domains in the world of product design and manufacture, that treat similar families of parts, to some extent share a “virtual” set of core concepts whose meanings may apply to all system domains. This understanding partly falls into the category of (1) the product model and part family effort fostered by various researchers (Molina *et al*, 1995; Balogun *et al*, 2004; Sudarsan *et al*, 2005; Gunendran *et al*, 2007) (Chapter 2 section 2.6) and (2) foundation ontology approaches for manufacturing (Chapter 2 section 2.3.4). However, since a majority of these approaches do not include tailored semantic definitions, this indicates that there is a need for an effective basis to support the provision of a set of reusable semantically-defined core concepts, which can be exploited by multiple system domains **[Requirement 3]**.

3.3.4 Harnessing Semantic Technologies to Assist Semantic Interoperability

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 domain models constitutes another key requirement **[Requirement 4]**. Such technologies may involve, for example, heavyweight ontologies (Chapter 2 section 2.3.2) and their platforms as well as ontology mapping methods (Chapter 2 section 2.3.6). Requirement 4 can be broken down into a number of sub-requirements, the discussions of which are partly based on the challenges reviewed by Shvaiko and Euzenat (2008), and exposed in the next sub-sections.

3.3.4.1 Knowledge Representation Formalisms

Capturing and representing the semantics of domain ontologies in computational form is central to sharing across product design and manufacture. Several families of knowledge representation formalisms have been developed to capture and represent ontology-based semantics. Figure 3-4 provides some examples of existing knowledge representation formalisms. 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).

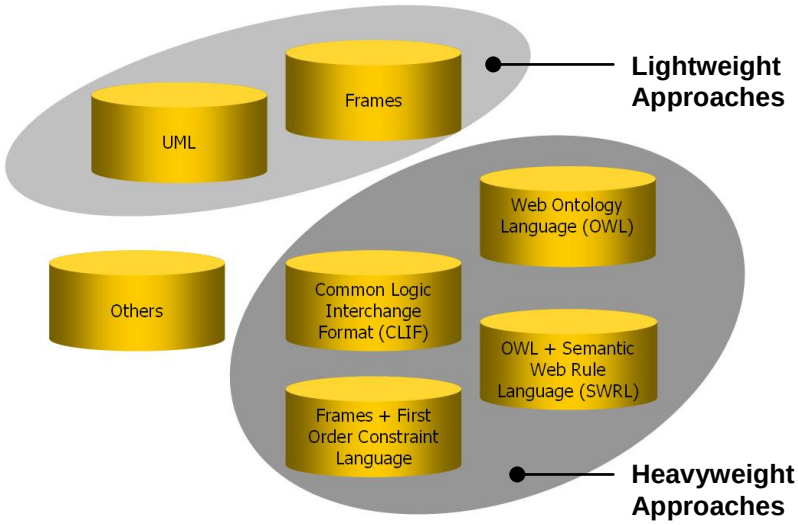


Figure 3-4 Examples of Knowledge Representation Formalisms

Figure 3-4 depicts that these formalisms for knowledge and semantic representation are either lightweight or heavyweight in nature (Gómez-Pérez *et al*, 2004). Heavyweight approaches rely on formal axioms that constrain the interpretation of concepts at computational level and are, therefore, preferred from a semantic point of view.

It has been acknowledged that there is a need for more mathematically rigorous, i.e. heavyweight, approaches (Chapter 2 section 2.3.2) to ensure that the true meaning of terminology coming from different systems is identical

to permit computational comparisons of the meaning of terms (Das *et al*, 2007; Young *et al*, 2007). Consequently, there exists an ongoing requirement to understand which family of knowledge representation formalism(s) allows the expressive capture and representation of product design and manufacture semantics **[Requirement 4a]** for the development of semantically-rich models. An experimental investigation explored in Appendix B contributes to this understanding by showing that a progression towards more expressive knowledge representation formalisms, like Common Logic (CL), is required to fully capture and represent semantic structures in product design and manufacture.

3.3.4.2 Resolution of Semantic Mismatches

Possible semantic mismatches that can exist between domain models are diverse in nature. The occurrence of these mismatches can be explained from different angles such as knowledge elicitation and knowledge representation (Hameed *et al*, 2004). When considered from the knowledge representation perspective, which provides a comprehensive way to describe semantic heterogeneity in systems, these mismatches are shown to occur at different levels of granularity (Visser *et al*, 1997; Hameed *et al*, 2004; Chungoora and Young, 2008b). Figure 3-5 exposes a classification of semantic mismatches based on the knowledge representation perspective.

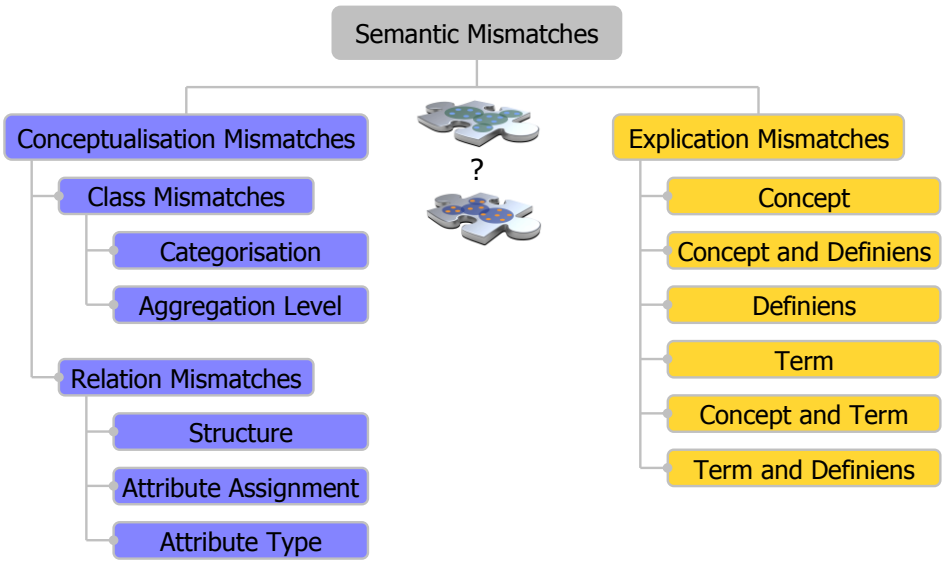


Figure 3-5 Classification of Semantic Mismatches (based on (Hameed *et al*, 2004))

In the figure it can be seen that there are two main trends to semantic mismatches namely:

- Conceptualisation mismatches which occur as a consequence of having two or more types of conceptualisations of a certain domain. Disparate conceptualisations may differ in the way their ontological concepts are defined or in the way these concepts are related to each other.
- Explication mismatches which are explained using three components of concept definitions, i.e. concepts, terms and definiens. A concept constitutes an underlying notion to be defined. A term is used to denote a particular concept and generally involves a human-assigned terminology. Definiens are other concepts which provide the building blocks of the definition of a more complex concept in the form of aggregated statements. Mismatches arising at any of the three components (i.e. concept, term and definiens) or combinations of components result in explication mismatches.

Examples of semantic mismatches, explained from the knowledge representation perspective and applied to the area of product design and manufacture, have been investigated (Chungoora and Young, 2008b). The gathered understanding leads to the identification of another requirement. With the intention of promoting semantic interoperability, there exists a prerequisite for exploring semantic technologies which can improve the identification and resolution of possible semantic mismatches between domain models **[Requirement 4b]**.

3.3.5 Concepts for Ontology Matching

A fundamental stage in the reconciliation of heterogeneous domain models involves the capability to match across ontology-based arguments (content) through the process of ontology mapping/matching (Chapter 2 section 2.3.6). Matching relationships, which can be associated across domain models, hence facilitate the process of building an agreement on concept spaces (Doerr *et al*, 2003). Figure 3-6 shows an example of how the specification of

ontology matching relationships provides a convenient way to reconcile and interoperate between concepts from two domain ontologies.

The scenario depicts that if some desirable ontology matching relationships can be specified between the semantic structures (definiens) that define two hole feature concepts “Simple_Hole” and “Plain_Hole” from “Ontology X” and “Ontology Y” respectively, then it is possible to not only understand how these semantic structures correspond, but it also raises awareness of what type of knowledge could be shared across “Ontology X” and “Ontology Y” at the hole feature definition level.

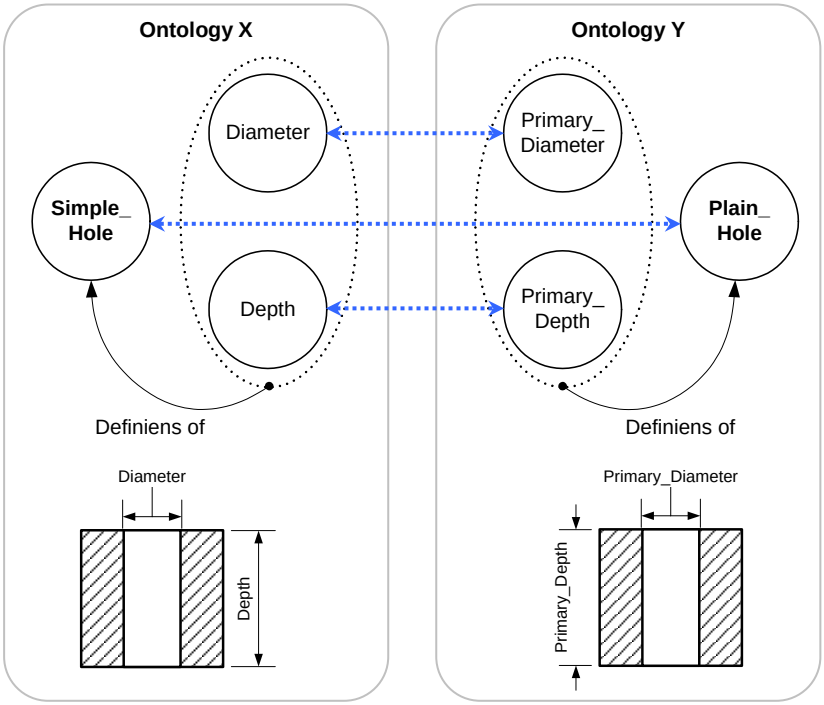


Figure 3-6 Example of Ontology Matching Relationships to Reconcile Hole Feature Concepts from Two Domain Ontologies

Several ontology mapping/matching methodologies exploit the ability to formally specify cross-ontology correspondences as a means to establishing mappings from which ontology interoperability can be achieved (Maedche and Staab, 2000; Kiryakov *et al*, 2001a; Madhavan *et al*, 2002). However, at present, ontology mapping approaches still deserve attention so as to improve the capability for more effectively matching across ontologies and verifying the integrity of mappings. Consequently, a key requirement is concerned with the

need for methods to explicitly and formally specify ontology matching relationships between domain models **[Requirement 4c]**.

3.3.6 Performance of Methods for Semantic Reconciliation

Performance is of prime importance in many dynamic applications, for example, where a user cannot wait too long for the system to respond (Shvaiko and Euzenat, 2008). Current methods for ontology matching may resolve from linear time to quadratic time, which may imply several minutes, hours or even days to complete a matching task (Shvaiko and Euzenat, 2008). Performance is also related to the level of automation of methods for the semantic reconciliation of ontologies. Several approaches have been proposed in order to reconcile heterogeneous ontologies using ontology mapping/matching, hence resulting in an extensive range of methodologies for leveraging ontological semantic interoperability (Euzenat and Shvaiko, 2007). These ontology mapping methodologies attempt to provide ways for reconciling distributed semantics either automatically or semi-automatically.

It is thus widely accepted that manual mapping is a labour-intensive task (Mitra and Wiederhold, 2002) which loses its feasibility as larger ontologies have to be reconciled. Consequently, it follows that the performance level of semantic reconciliation approaches proves to be an important asset contributing to the strength of semantic technologies for supporting semantic interoperability. For this reason, a requirement is present to support higher performance levels as far as semantic reconciliation processes are concerned **[Requirement 4d]**.

3.4 Summary of Requirements

This chapter has identified a set of requirements, whose importance is paramount to supporting semantic interoperability in design and manufacture, thereby meeting the second objective of this work (see Chapter 1 section 1.3.1). The investigation of these requirements has been based on the

aspects that occur in the organisation of manufacturing information for engineering interoperability (Gunendran *et al*, 2007).

Close considerations to these requirements are made during the proposal and development of a novel ontology-based framework, whose underlying principles are revealed in the forthcoming chapters. In other words, the investigated requirements form a checklist of development specifications for the framework. A summary of the explored requirements is provided next:

- **Requirement 1:** There is a need for a progression towards the seamless exchange of design and manufacturing knowledge through the capture of semantics coming from multiple product lifecycle viewpoints.

- **Requirement 2:** There exists a need for providing semantic relationships between different but overlapping viewpoints in order to support integrated semantic capabilities.

- **Requirement 3:** There is a need for an effective basis to support the provision of a set of reusable semantically-defined core concepts, which can be exploited by multiple system domains.

- **Requirement 4:** There is a need for harnessing the appropriate semantic technologies in order to facilitate the formal capture of domain semantics and to support shared meaning across domain models.

- **Requirement 4a:** It is essential to understand which family of knowledge representation formalism(s) allows the expressive capture and representation of product design and manufacture semantics.

- **Requirement 4b:** There exists a prerequisite for exploring semantic technologies which can improve the identification and resolution of possible semantic mismatches between domain models.

- **Requirement 4c:** There is a necessity for methods to explicitly and formally specify ontology matching relationships between domain models.

- **Requirement 4d:** A requirement is present to support higher performance levels as far as semantic reconciliation processes are concerned.

All the above-mentioned requirements have been fully taken into account for the proposal and development of the research framework (see Chapter 4). It is to be noted that these requirements have been exposed partly based on the semantic interoperability issues that derive from the research scope. However, the statement of these requirements has remained at a high level which implies that the investigated requirements are applicable to the field of product design and manufacture as a whole.

4 A Novel Framework to Support Semantic Interoperability in Product Design and Manufacture

4.1 Introduction

The purpose of this chapter is to expose the author's concept for a novel ontology-based framework which helps support semantic interoperability in product design and manufacture. As an attempt to resolve the semantic issues that prevent the achievement of semantic interoperability, the concept proposes a four-layered approach: The Semantic Manufacturing Interoperability Framework (SMIF), which is explained in further detail in section 4.2. The first element of the framework, identified as the Foundation Layer, exploits a heavyweight ontological underpinning and is explained in section 4.3. This Foundation Layer provides a ladder of capability for the specialisation of domain models, which can be individually developed in the Domain Ontology Layer. Section 4.4 discusses some of the basic implications within the Domain Ontology Layer.

In section 4.5, the Semantic Reconciliation Layer is briefly explained. The latter, also partly established as a result of the Foundation Layer, involves the semantic reconciliation of cross-domain arguments coming from pairs of domain models developed in the Domain Ontology Layer. Interactions between the first three layers of the framework are key to the fourth level, the Interoperability Evaluation Layer, which is explained in section 4.6. This fourth level is where the retrieval of cross-domain correspondences and ontological knowledge sharing capability can be evaluated. System boundaries and assumptions are discussed in section 4.7. Section 4.8 aims at aligning the main framework concepts to the requirements previously explored in Chapter 3. A summary of this chapter is then provided in section 4.9.

4.2 Semantic Manufacturing Interoperability Framework (SMIF)

The Semantic Manufacturing Interoperability Framework (SMIF) exploits a four-layered ontology-driven architecture towards meeting the identified requirements for semantic interoperability across product design and manufacture. The different layers of the Semantic Interoperability Framework are illustrated diagrammatically in the Figure 4-1, where the constituent layers are namely (1) a Foundation Layer, (2) a Domain Ontology Layer, (3) a Semantic Reconciliation Layer and (4) an Interoperability Evaluation Layer.

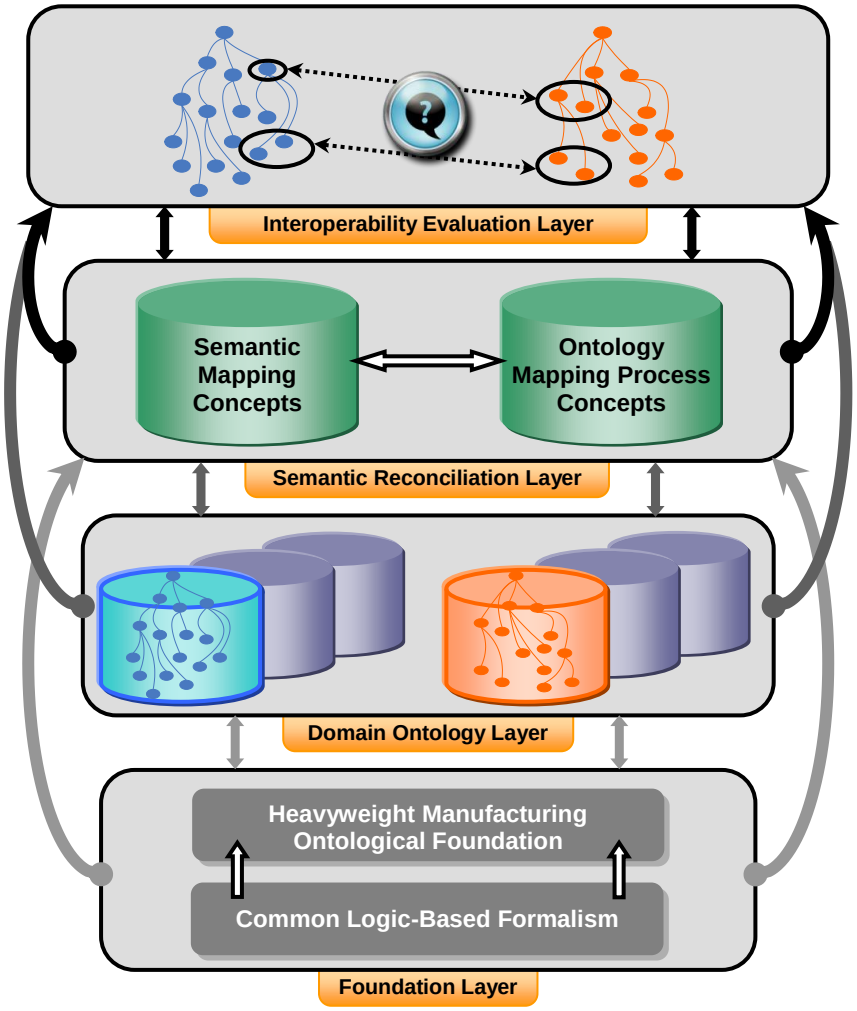


Figure 4-1 Semantic Manufacturing Interoperability Framework (SMIF)

The framework essentially draws its strength from the combined application and extension of different established methods, including ontological underpinnings such as the interlingua approach to interoperability (Gruninger and Kopena, 2005). This combined application of established methods shall be discussed in subsequent sections detailing the constituent layers of the SMIF. The novelty of the proposed concept, which is supported through the exploration of test cases, consists of three main areas namely:

- The development of a Semantic Manufacturing Interoperability Framework (SMIF) that contributes to the understanding of combined heavyweight ontology-based approaches to support semantic interoperability in product design and manufacture.
- The development of a heavyweight manufacturing ontological foundation, of core feature-based entity information and process semantics, which fosters the semantically-sound specialisation of domain models.
- A contribution to the understanding of verifiable logic-based semantic reconciliation methods as part of ontology mapping processes between pairs of domain models that have been based on the same foundation.

The proposal of the SMIF enables the research gaps summarised in section 2.8 of Chapter 2 to be addressed in the following way:

- The framework employs an ontology-based underpinning provided by the Foundation Layer and supports the capability to evaluate interoperable knowledge. The framework has been targeted to the field of product design and manufacture.
- The Foundation Layer consists of an upper ontology for the Common Logic-based formalism over which a heavyweight manufacturing ontological foundation is stacked. The Foundation Layer hence supports the understanding on the effective exploitation of foundation ontology approaches.
- The logic-based system capability supported by the Foundation Layer is conveyed to the subsequent layers of the framework. This provides the

potential for applying formal semantic-based methods during ontology matching.

- Furthermore, the framework uses a heavyweight ontological approach in order to benefit in the explicit and expressive representation of product and process semantics pertinent to design and manufacture.

4.3 Foundation Layer

The Foundation Layer is at the first level of the Semantic Manufacturing Interoperability Framework (SMIF) and conveys the essential capability for the existence of subsequent layers of the framework. This first level comprises two characteristic elements, namely a rigorous Common Logic-based ontological formalism over which a heavyweight manufacturing ontological foundation is constructed. Figure 4-2 provides a more detailed view of the Foundation Layer.

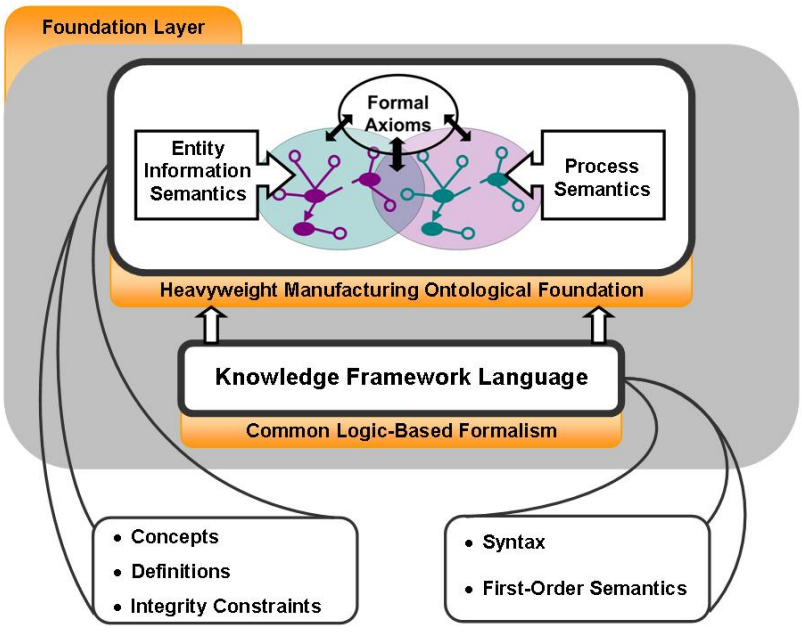


Figure 4-2 The Foundation Layer

From the diagram, it can be seen that the rigorous Knowledge Framework Language (KFL), a Common Logic-based formalism developed by Ontology Works Inc. (Ontology Works Inc., 2009), imparts the syntax and first-order semantics, governing the way in which the heavyweight manufacturing

ontological foundation is formalised at computational level. It is to be pointed out that Common Logic-based knowledge representation formalisms like KFL applied to the research problem under investigation is unprecedented and, therefore, constitutes a new aspect which is brought forward in this work (also see Appendix B for a justification of the chosen ontological formalism).

4.3.1 Heavyweight Manufacturing Ontological Foundation

The heavyweight manufacturing ontological foundation captures and expressively represents generic feature-based entity information and process semantics together with some of the existing relationships that hold between entities and processes (see Chapter 5 section 5.2 for a definition of an ontological foundation). The researched heavyweight manufacturing ontological foundation constitutes a novel effort towards the improved definition of foundation ontologies for manufacturing achieved through the development, from a low level of granularity, of process and entity information semantics.

Firstly, 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) (see Appendix C.1). Since it has been shown that PSL provides intuitions for reasoning about various forms of processes (Cheng *et al*, 2003; Bock and Gruninger, 2005; Bock, 2006; Das *et al*, 2007), this implies that the choice of PSL for the capture of generic process semantics in the Foundation Layer is relevant.

PSL has been written in the Common Logic Interchange Format (CLIF) (PSL Website, 2009). CLIF as well as KFL are both based on the ISO Common Logic standard (see Chapter 2 Figure 2-2). However, the main difference between the two is that CLIF is platform-independent whereas KFL is platform-dependent and the latter is used for implementation purposes on the appropriate ontological environment. Since both CLIF and KFL are Common Logic-based, this clearly implies that PSL expressed in CLIF can completely be expressed in KFL as well. This constitutes an important benefit which

helps to reduce the ontology development time spent during the implementation of the Foundation Layer. Hence, this work provides the first factual implementation of relevant portions of PSL on a concrete ontological platform capable of handling the required semantic expressiveness.

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 is being expanded to include a broader understanding of entity information semantics (see Figure 4-3). Thus, secondly, in order to capture these generic entity information semantics, for the meaningful description of product representations, the fundamentals from the revised Core Product Model (CPM) (Fenves *et al*, 2004) and those from ISO 10303 AP224 (ISO 10303-224, 2006) are being exploited and adapted to the framework needs. 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 are formalised in the Foundation Layer to obtain generic mechanical product representation semantics based on feature definitions. It is to be noted that selected concepts coming from the CPM as well as ISO 10303 AP224 are lightweight in nature. Hence, the progression from their lightweight representations to their corresponding heavyweight semantics is a novel aspect undertaken at this level of the framework. Figure 4-3 identifies a conceptual picture of the combined approach used in the Foundation Layer. The figure emphasises the “Object” concept from PSL which neatly maps to the “Common Core Object” from the CPM. Other CPM concepts, for example, “Feature”, “Form” and “Geometry” are integrated with ISO 10303 AP224 concepts. Appendix C.2 documents the relevant entity information semantics explored in this work.

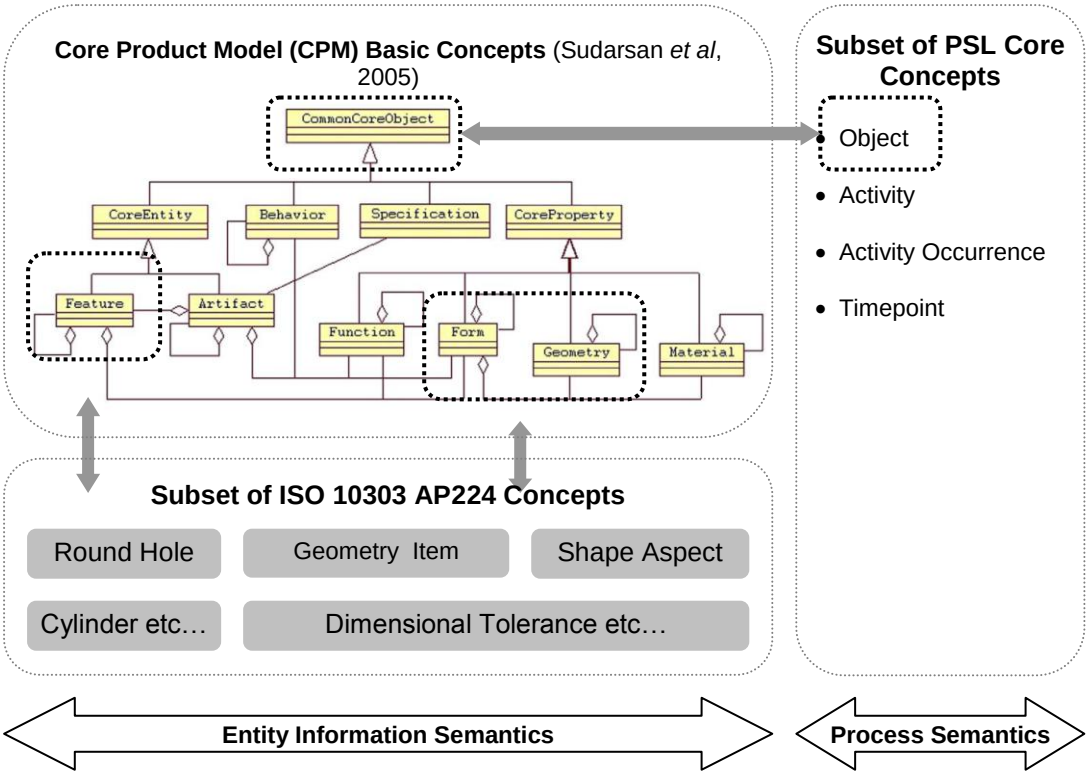


Figure 4-3 Conceptual Diagram of the Combined Approach Employed in the Heavyweight Manufacturing Ontological Foundation

4.4 Domain Ontology Layer

The Domain Ontology Layer is at the second level in the Semantic Manufacturing Interoperability Framework (SMIF). At this level, formal axiomatised semantics from the heavyweight manufacturing ontological foundation can be specialised for the development of domain-specific ontologies and the capture of domain-centric knowledge. The types of concepts explored in the Domain Ontology Layer contribute to new knowledge by consolidating the understanding behind the ontological mechanisms that ensure the integrity-driven development of domain models that are based on the same manufacturing foundation ontology.

In the Domain Ontology Layer, the purpose of a domain ontology is generally seen as providing vocabularies of the concepts within a specific domain and their relationships, of the activities taking place in that domain, and of the theories and elementary principles governing that domain (Mizoguchi *et al*, 1995; van Heijst *et al*, 1997; Gómez-Pérez *et al*, 2004). In the context of this

work, the Domain Ontology Layer is where domains develop domain models (i.e. domain ontologies and their related KBs). Domain ontologies are bound to the preferences, practices and terminologies of individual domains.

4.4.1 Part Family Semantics

The extent to which the heavyweight manufacturing ontological foundation captures entity information semantics inevitably dictates the types of products or families of parts that can be represented at the Domain Ontology Layer. Figure 4-4 shows examples of rotary type part families that can potentially be represented in the Domain Ontology Layer. Domain-specific products may involve combinations of different shapes. The complexity of foundation entity information semantics allows concepts like shape representation items, feature shape aspects, standard features (such as hole features, cylinders, blocks and compound features), transition features and dimensional tolerances to be explicitly represented.

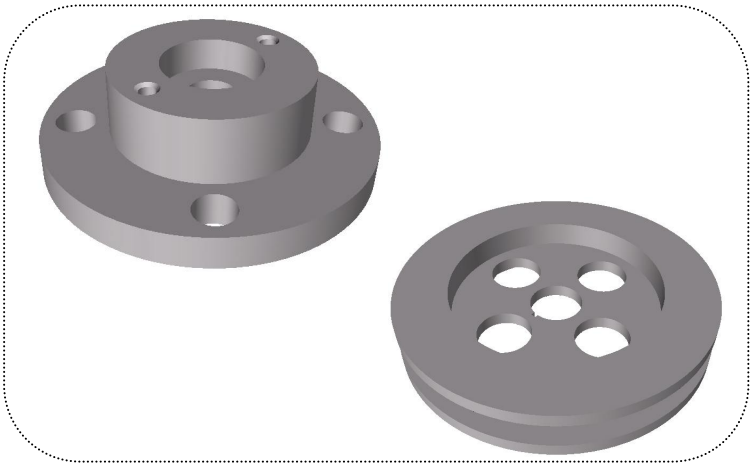


Figure 4-4 Example of Part Families which can be Represented at the Domain Ontology Layer

4.4.2 Manufacturing Process Semantics

In this work, the representation of domain-dependent manufacturing process semantics is built on the PSL concepts formalised in the heavyweight manufacturing ontological foundation. These PSL concepts entail the PSL Core and PSL Outer-Core theories (ISO 18629, 2005) (see Appendix C.1).

These two proven theories regroup a number of concepts which when used allow the expressive description of manufacturing process sequences. Figure 4-5 illustrates an example of a domain-defined machining process sequence whose semantics can readily be captured using PSL Core and Outer-Core. In the process planning sequence, it can be seen that the occurrences of processes are ordered along a timeline where “Centre Drilling” (a compulsory precondition) takes place before “Drilling” which in turn takes place before a choice of either “Reaming” or “Boring”.

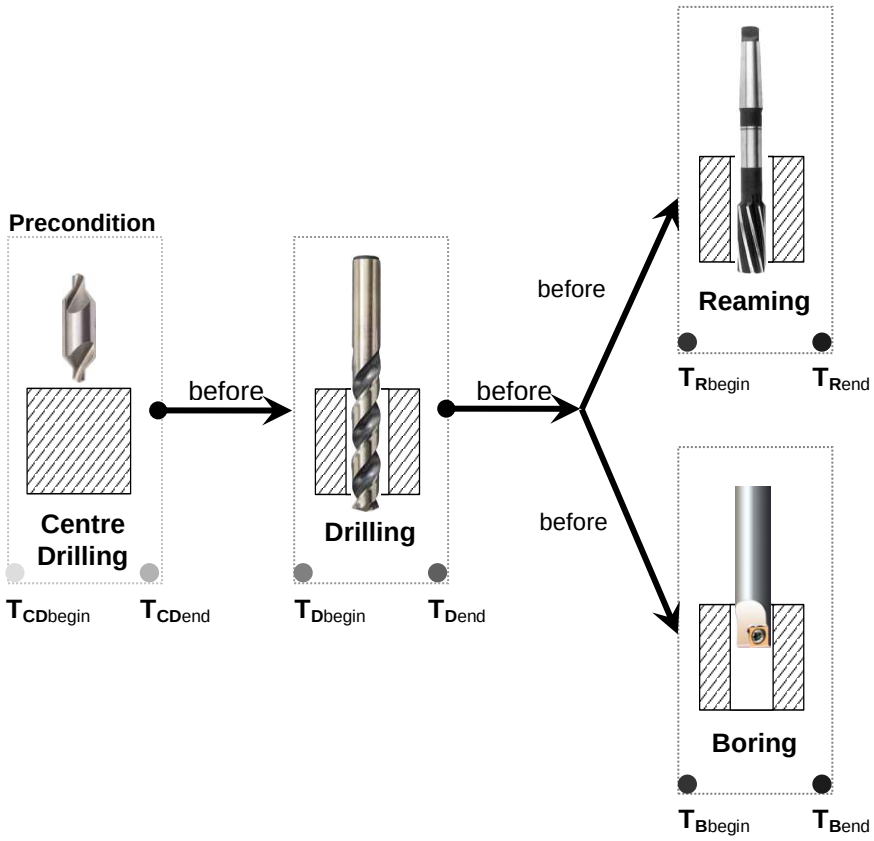


Figure 4-5 Example of a Domain-Defined Machining Process Sequence

4.5 Semantic Reconciliation Layer

The Semantic Reconciliation Layer is at the third level in the Semantic Manufacturing Interoperability Framework (SMIF). The third layer combines the definition of new semantic mapping concepts alongside ontology mapping process concepts. The primary aim of the Semantic Reconciliation Layer is to provide adequate support for the reconciliation of domain models that are developed in the Domain Ontology Layer and that need to interoperate.

The approach to semantic reconciliation pursued in the SMIF revolves around logic/rule-based ontology mapping methods. Several ontology mapping frameworks (see Chapter 2) that have been researched and validated may be regarded as utilising three broad methods for achieving ontology and semantic interoperation namely: (1) the application of heuristics and linguistic-based techniques, supported by formal algorithms, to provide measures of similarity between ontological concepts, (2) the identification and allocation of semantic relationships between ontological entities, sometimes referred to as "semantic bridges" (Maedche *et al*, 2002), and (3) combinations of both (1) and (2) for enhancing the capability of ontology mapping frameworks.

Although ontology mapping research appears to be relatively mature, yet there still exist limitations to current ontology mapping frameworks. For example, many mapping techniques do not provide complete solutions for interoperation at the structural levels of domain models, such as classes, ontological functions and instances. Moreover, some ontology matching methods are still dependent on human intervention for the verification of mappings. In many cases, mapping relations across ontologies remain basic and, therefore, do not carry sufficiently-expressive interoperable knowledge.

These limitations are being tackled through the exploration of novel verifiable Common Logic-based mapping methods in the Semantic Reconciliation Layer. At this framework level, logic-based statements can be formulated to capture the conditions behind semantic reconciliation. The capabilities of the logic-based mechanisms involved in the semantic reconciliation surpass those of other commonly exploited heavyweight approaches, such as Frames with first order constraint languages and Description Logics with rule languages (Gómez-Pérez *et al*, 2004).

4.5.1 Semantic Mapping Concepts

Semantic mapping concepts consist of ontological relations that are written in the Knowledge Framework Language (KFL). These semantic mapping concepts hold true for cross-domain arguments (e.g. classes, ontological

functions and instances), based on logical situations that arise between specialised domain models. Consider the class “Round_Hole” which is a class concept defined in the Foundation Layer (see Figure 4-6). In the IDEF5 schematic (refer to Appendix A for an overview of the IDEF5 schematic language), the “Round_Hole” class has two specialisations in a “DomainX” ontology and two specialisations in a “DomainY” ontology. In the Semantic Reconciliation Layer, an example of a semantic mapping concept could be formulated to capture the following informal intuitions:

If the class “Round_Hole” from the Foundation Layer has a number of specialised classes in the “DomainX Ontology” and also has a number of specialised classes in the “DomainY Ontology”, then a semantic mapping concept can be assigned between cross-domain sub-classes of “Round_Hole”, to understand that pairs of these sub-classes originate from the same parent class.

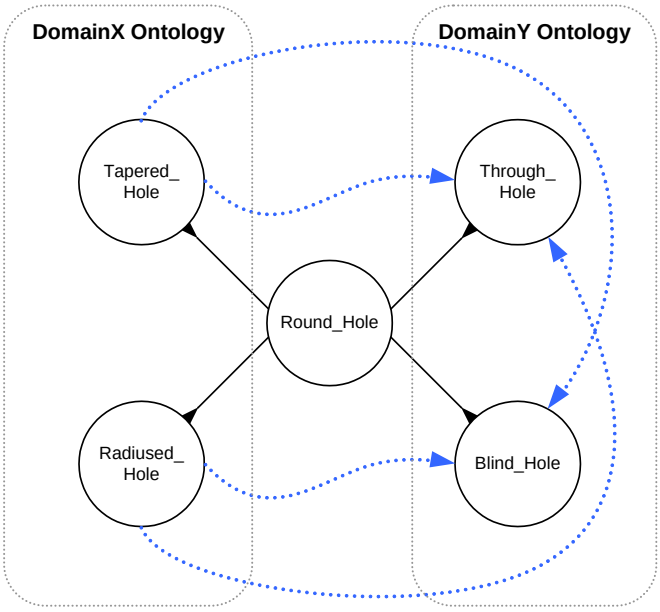


Figure 4-6 Example of a Semantic Mapping Concept

This semantic mapping relation is denoted by the dotted arrows in Figure 4-6. The example is relatively simple and the mapping information could be checked by browsing through the taxonomy of the two domain ontologies. The point here, however, is to indicate that the definition of other semantic mapping concepts, based on more complex logical statements, can allow

intricate interoperability scenarios to be modelled. These semantic mapping concepts can be exploited for situations arising at various levels of the structure of domain models, which constitutes an improvement over current methods, as discussed further in Chapter 6.

4.5.2 Ontology Mapping Process Concepts

The process of ontology mapping in the Semantic Reconciliation Layer can be performed for two domain models at a time. The process comprises a first stage of loading two domain models together, i.e. a simple merging process, in such a way that the content from both models stays distinct for each of them. Then, semantic mapping concepts are loaded into the merged model. During this ontology alignment process, where a collection of binary relations are established between the vocabularies of the two ontologies (Kalfoglou and Schorlemmer, 2003a), semantic mapping concepts are automatically fed to the merged models. If semantic mapping relations hold true between cross-domain arguments, based on the logic that defines these relations, then the relevant relations are automatically allocated between the relevant cross-domain arguments.

4.6 Interoperability Evaluation Layer

The Interoperability Evaluation Layer is at the fourth level of the framework. At this level, interoperable knowledge queries can be executed with the intention of finding correspondences between arguments from two domain models that have been processed in the third level of the framework. The main activity involved in the Interoperability Evaluation Layer is concerned with the retrieval, i.e. inference, of semantic mapping concepts which carry the type of interoperable knowledge. Results obtained can be verified through logical proof. In other words, query responses can be reviewed with the intention of finding the truth behind their occurrence. Another element of the Interoperability Evaluation Layer entails the process of easily creating, running and managing queries which is facilitated using a developed user interface further explained in chapters 6 and 7.

4.6.1 Interoperability Evaluation through Queries

There are two ways by which the discovery of cross-domain correspondences can be made. They both revolve around the formulation of logic-based queries which are written in a form similar to the Common Logic Interchange Format (CLIF). The first, remains relatively straight forward and requires the user selecting a particular semantic mapping concept and querying the concept to see whether any results are retained for the query. For instance, assuming the scenario in Figure 4-6, but the user is not aware of it, then on running a query in the form:

Find all arguments that are bound to the specific semantic mapping concept,
The result of the query should be:

All possible pairs of cross-domain subclasses, for example, "Tapered_Hole and Through_Hole", "Radiused_Hole and "Through_Hole" out of a total of four possible combinations of matches.

The other method of inference is concerned with the creation of logical query statements that retrieve all semantic mapping concepts common between two cross-domain arguments in a single transaction. Figure 4-7 informally visualises the content of such a query and the corresponding response. The expected query response(s) obviously includes the semantic mapping relation (see dotted arrow) in Figure 4-6. This special form of knowledge querying is preferred over the first one since it can more effectively deduce all cross-domain semantic mapping concepts that hold for two known arguments across domain models, thereby optimising the sharable knowledge discovery process. Hence, the awareness of the occurrence of semantic mapping concepts between cross-domain arguments in the Interoperability Evaluation Layer provides the essential knowledge sharing capability.

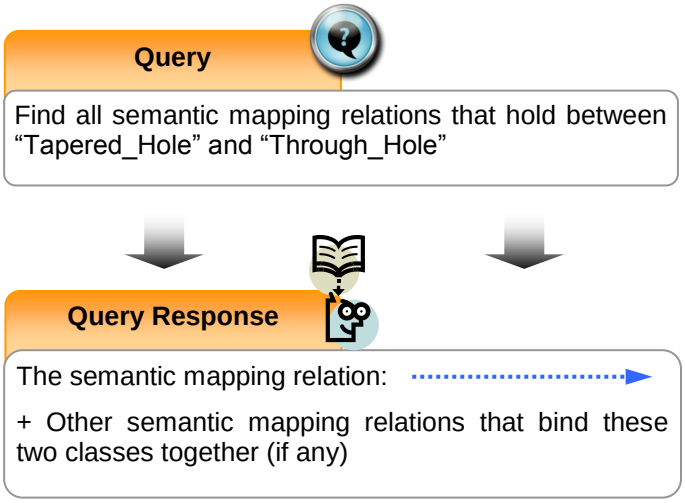


Figure 4-7 Example of an Informal Knowledge Query and the Query Response

4.6.2 Interoperability Evaluation Assistant

As a consequence of the large number of semantic mapping concepts that can possibly be developed, this imposes an important issue on the implementation aspects of the Interoperability Evaluation Layer. This issue is concerned with the management of executable interoperable knowledge queries for reuse. The fourth level of the framework additionally focuses on an appropriate User Interface (UI), which facilitates user-system interaction (Chungoora and Young, 2008b). The Web-based UI, called the Interoperability Evaluation Assistant, most importantly provides a method for the classification of queries and the ability to dynamically retrieve queries for improved performance during mapping knowledge discovery, an aspect that remains distinct to this work.

4.7 System Boundaries and Assumptions

The development of the Semantic Manufacturing Interoperability Framework (SMIF) and its constituent levels requires the identification of relevant system boundaries and assumptions. This is because the proposed framework is being developed aligned to the research scope, which considers specific areas of interoperability in product design and manufacture. In the Foundation Layer, it is obvious that a boundary is placed on the extent to which entity

information and process semantics enable product and process representation respectively. Thus, only the most relevant subsets of the Core Product Model (CPM), ISO 10303 AP224 and the Process Specification Language (PSL) are being targeted.

Furthermore, since the Knowledge Framework Language (KFL) provides an expressive knowledge representation formalism, this implies that ontologies that employ less expressive formalisms can be mapped to KFL without any loss of semantics while the converse is not likely to be completely achievable. This issue remains peripheral to this work, and for this reason, this investigation does not portray the semantic interoperability of distributed ontologies that are formalised using ontological formalisms other than the Common Logic-based KFL.

In the Semantic Reconciliation Layer it is assumed that the extent of semantic mismatches is viewed from the knowledge representation perspective (Visser *et al*, 1997; Hameed *et al*, 2004). However, it has been acknowledged that semantic discrepancies may well be considered from various other viewpoints (Klein, 2001). Moreover, because semantic mapping concepts can be used to capture a range of reconciliation scenarios, different mappings levels are likely to exist. This suggests that interoperable knowledge queried in the Interoperability Evaluation Layer can have different levels of importance to the expert. As a result, it is evident that the intended interoperable knowledge, to be discovered between two domain models, remains dependent on its perceived importance.

Furthermore, the framework as a whole assumes a static view on ontologies and KBs. In reality, different versions of domain ontologies and KBs are a common case, where it becomes important to support the management of evolving domain model content. In the framework, ontology versioning (Klein, 2001) is not taken into account, meaning that considerations for ontology evolution would imply the additional management of the mechanisms exploited in all four layers of the framework.

4.8 Aligning the Framework with Semantic Requirements

As previously mentioned, the Semantic Manufacturing Interoperability Framework has been developed with strong considerations made to satisfy the set of requirements investigated in Chapter 3. This section establishes how the different elements of the framework, as well as the framework in its entirety, satisfy these requirements.

The matrix shown in Figure 4-8 matches the framework and its components to the set requirements. Requirement 1 is met through the combined approach involving PSL, CPM and ISO 10303 AP224 in the Foundation Layer (see Figure 4-8 label **(A)**). Capturing the semantics from these methods enables a number of viewpoints to be considered in product design and manufacture. These viewpoints include, for example, the GD & T, functional, machining and process planning.

Semantic relationships between different but overlapping viewpoints are targeted through the specification of entity information, process semantics and the participation relationships that hold between them, based on the combined approach used to meet Requirement 1 **(B)**. This, therefore, helps to satisfy Requirement 2.

In order to support an effective basis for the provision of shared meaning, the heavyweight manufacturing ontological foundation is exploited. Since providing shared meaning is where ontological approaches have been pursued (Young *et al*, 2007), this clearly implies that the ontology-based slant within the framework is favoured. The types of semantics explored in the ontological foundation pertain to an array of core feature-based concepts that can be reused and extended by a multitude of domains. This depicts that the Foundation Layer and the interactions that it supports with the Domain Ontology Layer help meet Requirement 3 **(C)**.

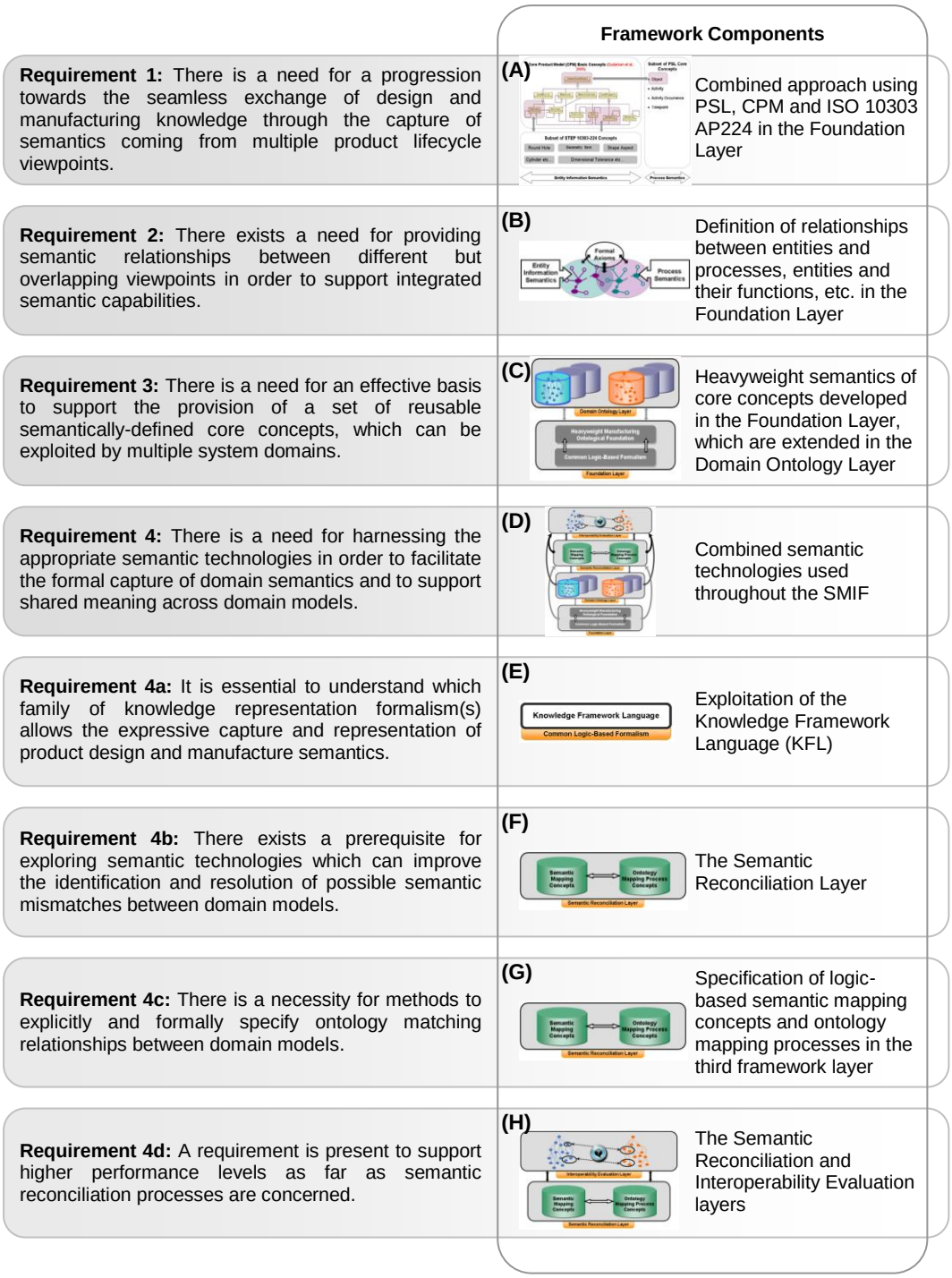


Figure 4-8 Aligning the SMIF and Its Components to Semantic Requirements

To be able to harness the appropriate semantic technologies to facilitate the capture of domain semantics and to support shared meaning across domains, the SMIF harmonises four different dimensions, i.e. four distinct layers which adopt specific semantic technologies into a single framework, thereby satisfying Requirement 4 **(D)**. In the sub-requirements of Requirement 4, such as the need to understand appropriate families of knowledge representation

formalisms (Requirement 4a), a study that leads to the choice of the Common Logic-based formalism, conveys this understanding (see Appendix B) **(E)**.

Furthermore, one of the purposes of the Semantic Reconciliation Layer is to deal with semantic heterogeneity across domain models, and provide mechanisms by which semantic mismatches can be identified and possibly resolved (Requirement 4b) **(F)**. The specification of rigorous semantic mapping concepts in the third layer of SMIF satisfies the need for improved methods of specifying ontology matching relationships (Requirement 4c) **(G)**. Moreover, interactions between the Semantic Reconciliation and Interoperability Evaluation layers and their implementations, help support higher performance levels as far as semantic reconciliation processes are concerned, as these are optimised for the SMIF **(H)**. By so doing, the third and fourth layers of SMIF aim at meeting Requirement 4d.

4.9 Summary

This chapter has exposed the author's ideas for a novel ontology-based approach to support semantic interoperability in product design and manufacture. This has helped to fulfil part of the third objective of this research, linked to the proposal of a framework solution (see Chapter 1 section 1.3.1).

The framework concept has been established with a strong view on the requirements previously analysed in Chapter 3. The Semantic Manufacturing Interoperability Framework (SMIF) employs a four-layer architecture which facilitates the interoperation of domain models as long as these models have been based on the same ontological foundation. A key contribution of the SMIF lies in its novel understanding which derives from the development a heavyweight manufacturing ontological foundation of feature-based entity information and process semantics. This foundation provides a ladder of capabilities including the fidelity-driven (i.e. semantically-sound yet flexible-enough) specialisation of domain models.

Other benefits involve the application of competitive semantic reconciliation techniques. Semantic mapping concepts which are defined ontological relations, backed by expressive logic (hence their heavyweight nature) are under exploration as part of these reconciliation techniques. The outcome from the third level provides a stepping stone for running intelligent queries in the Interoperability Evaluation Layer in order to derive valuable correspondences between cross-domain arguments. These correspondences are synonymous of sharable knowledge. Explanations of the different components of the SMIF and their interactions are examined in greater detail in chapters 5 and 6.

5 Foundation and Domain Ontology Layers

5.1 Introduction

This chapter is divided into two main sections. The first, explained in section 5.2, considers the Foundation Layer, paying particular attention to expose the different types of intuitions, assumptions over these intuitions, and the semantics captured at this level of the framework. These semantic structures, further developed in the sub-sections of section 5.2, include process semantics, entity information semantics and the key participation relationships that hold between them. The ontology development process follows the knowledge engineering methodology (Noy and McGuinness, 2001).

The second part of the chapter involves an explanation of the Domain Ontology Layer in section 5.3. The various ways in which domains reuse and specialise the semantics coming from the Foundation Layer are clarified and exemplified, in order to depict the differences and interactions between the Domain Ontology Layer and the Foundation Layer. Section 5.4 then summarises the key points from the chapter. It is to be noted that the semantic structures presented here intend to support the relevant set of requirements discussed earlier. Furthermore, ontology schematics featured in this chapter are represented using the IDEF5 schematic language.

5.2 Foundation Layer

The Foundation Layer is dependent of the Knowledge Framework Language (KFL), based on Ontology Works Upper Level Ontology (ULO) (Ontology Works Inc., 2009), for the formal specification of a heavyweight manufacturing ontological foundation. Such an ontological foundation, as perceived in this work, is regarded as an integration of intuitions that provide effective meta-concepts, with well-established human-perceived meaning, for modelling domain ontologies (Cho *et al*, 2006).

The heavyweight manufacturing ontological foundation possesses the property of capturing generic but constrained entity information and process semantics, together with the participation relationships that hold between entities and processes. Reusable concepts are captured within the ontological foundation. The concepts explored remain generic in terms of the underlying intuitions, constraints and definitions governing their existence. Axiomatised concepts at this level provide a reusable set of semantics and behaviours which can be individually specialised in the Domain Ontology Layer (see section 5.3) to meet the needs of individual product design and manufacture domains.

Traditional foundation ontology approaches, such as the Basic Formal Ontology (Grenon, 2003) and the Descriptive Ontology for Linguistic and Cognitive Engineering (Gangemi *et al*, 2003), generally define reusable core ontologies from a philosophical viewpoint. Unlike these traditional approaches, the heavyweight manufacturing ontological foundation in this work has been developed as a core ontology with a strong slant onto important principles arising in feature-based product design and manufacture. The nature of the Foundation Layer thus provides an understanding of how effective foundation ontology approaches can be tailored to support the communication requirements of manufacturing (Young *et al*, 2007). The constituent approaches and theories used in the Foundation Layer are next discussed.

5.2.1 Process Semantics

Process semantics used in the heavyweight ontological manufacturing foundation derive completely from the Core and Outer-Core theories of the Process Specification Language (ISO 18629, 2005). The most up-to-date version of PSL is available from the PSL Website (PSL Website, 2009), and this has been the primary source for obtaining the Core and Outer-Core theories in the Common Logic Interchange Format. The CLIF version of PSL Core and Outer-Core has been expressed using the Knowledge Framework

Language (KFL). Appendix C.1 documents relevant PSL concepts exploited in this work alongside the corresponding IDEF5 schematics.

The purpose of PSL Core is to axiomatise a set of intuitive semantic primitives that is adequate for describing the fundamental concepts of manufacturing processes (PSL Website, 2009). Figure 5-1 depicts the four classes defined in PSL Core namely “Object”, “Activity”, “Activity_Occurrence” and “Timepoint”. Note that the root class “Origin” is an abstract class defined in the Foundation Layer to keep the taxonomy tidy, and thus does not carry any formal semantics other than being the super-class of the four defined classes from PSL Core.

PSL Outer-Core, consists of a number of theories that together bring greater strength to PSL, in terms of logical expressiveness. PSL Outer-Core involves the: (1) Theory of Subactivities, (2) Theory of Occurrence Trees, (3) Theory of Discrete States, (4) Theory of Atomic Activities, (5) Theory of Complex Activities and (6) Theory of Activity Occurrences.

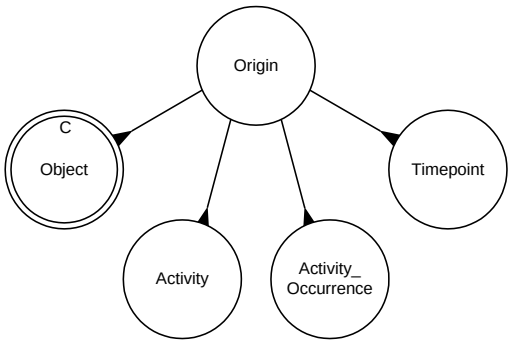


Figure 5-1 PSL Core Classes

5.2.2 Entity Information Semantics

The development of entity information semantics compensates for the limited ability of PSL to capture object-centric semantics (Young *et al*, 2007). Entity information semantics are explored in the heavyweight manufacturing ontological foundation to formalise a set of semantic structures for the formal representation of mechanical product definition using features. In other words,

entity information semantics help capture enriched product models by embedding the meaning associated to:

- Product feature geometries expressed as a collection of 2-D faces and their semantic relationships to produce 3-D features,
- The dimensional and dimensional tolerance parameters related to product feature geometries in design and manufacture,
- The functional aspects of product features, thereby providing a useful way to describe features from different viewpoints, and,
- The aggregation of features into complete artifacts or families of parts.

The sub-sections of section 5.2.2 document the progressive build up of core intuitions which help to capture the above-mentioned semantic capability in the heavyweight manufacturing ontological foundation. Entity information semantics in the heavyweight manufacturing ontological foundation are defined based on the fundamentals from the revised Core Product Model (CPM) (Fenves *et al*, 2004) as a proposed foundation for interoperability in next-generation product development systems (Szykman *et al*, 2001) and those from ISO 10303 AP224 because of its slant onto wide-ranging feature definitions and also because features support the integration between design and manufacture (Abouel Nasr and Kamrami, 2006; Dartigues *et al*, 2007; Nassehi *et al*, 2007).

This combined approach used in the Foundation Layer supports the ability to capture, represent and axiomatise important reusable and extensible entity information semantics. The approach shows that the specification of product definition semantics backed by the expressive Common Logic-based KFL is a novel aspect brought forward, that from a semantic viewpoint goes one step beyond related work. This is because documented work points to the fact that, so far, conceptualisations involving product definitions have at most exploited heavyweight Description Logics with rule languages (Patil *et al*, 2005; Kim at al, 2007; Rabe and Gocev, 2008).

5.2.2.1 Core Entities and Core Properties

As a starting point for capturing entity information semantics, the required ontological commitments have been identified and are based on the following intuitions:

- Concepts defined to capture and represent entity information semantics extend the “Object” concept from PSL.
- “Core_Entity” (Fenves *et al*, 2004) is a kind of abstract object from which the concepts “Artifact” and “Feature” originate.
- “Core_Property” (Fenves *et al*, 2004) is another kind of abstract object whose hierarchy captures relevant notions that embody core entities.

Thus, the two concepts found in the CPM namely “Core_Entity” and “Core_Property” initially categorise the “Object” class as shown in Figure 5-2.

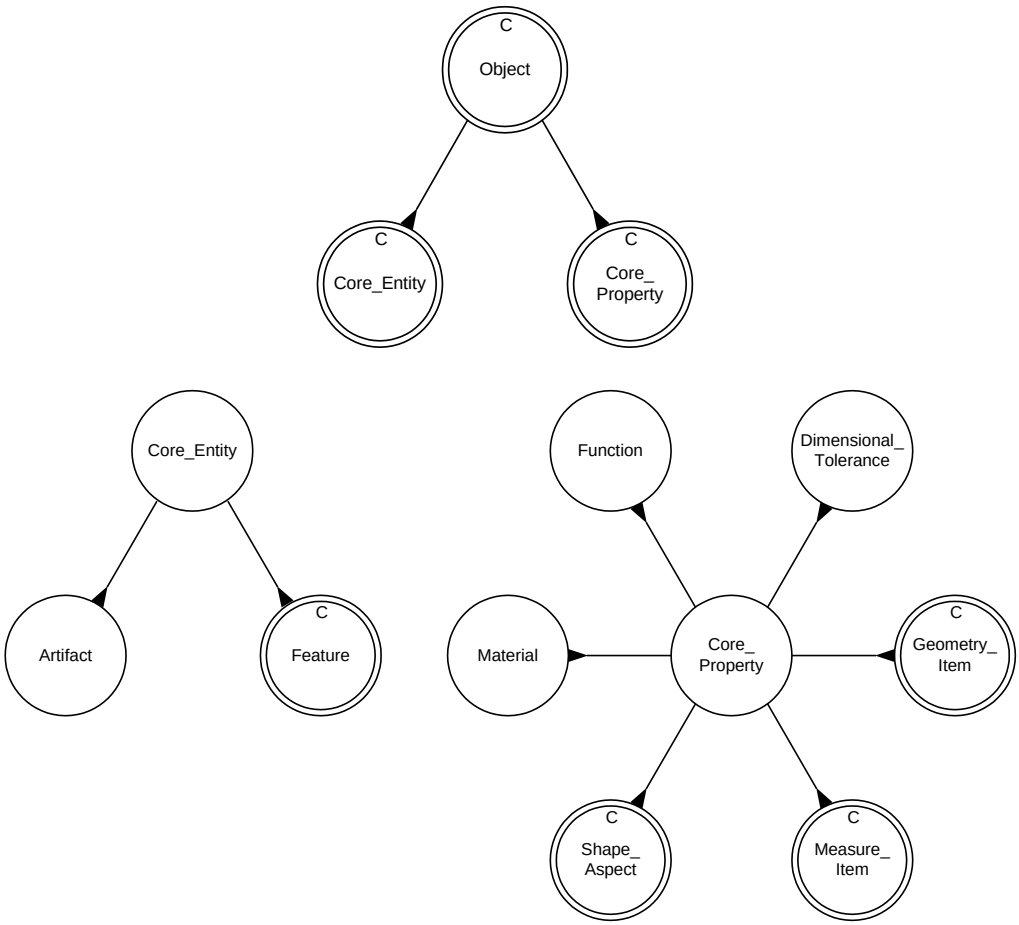


Figure 5-2 Class Hierarchy of “Core_Entity” and “Core_Property”

While the abstract “Core_Entity” concept involves the basic semantics of features and artifacts that hold features, the abstract “Core_Property” concept is present to provide more detail semantics, primarily used towards product feature definitions and their behaviours. Figure 5-2 also identifies the decomposition structure of “Core_Entity” and “Core_Property”. The concepts “Artifact”, “Feature”, “Function” and “Material” originate from the CPM while the remaining sub-classes of “Core_Property” are adapted from ISO 10303 AP224.

Two binary relations are specified to initially capture the idea that core entities may hold some function and some material, which are essential factors that govern the existence of entities in the first place (see Figure 5-3). By, for example, adding an axiom to capture the constraint that every core entity may hold some function, it is possible to enforce an optional necessary condition, which is also carried upwards to the Domain Ontology Layer. Expression 5-1 depicts the Common Logic Interchange Format (CLIF) statement of the integrity constraint (IC), i.e. axiom.

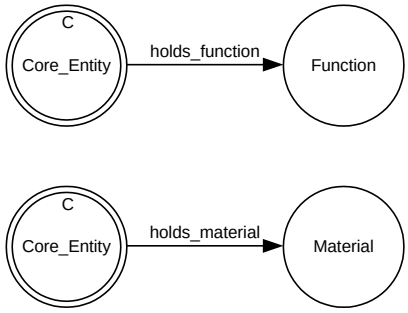


Figure 5-3 “holds_function” and “holds_material” Binary Relations

```
(forall (?coreEnt)
(=> (Core_Entity ?coreEnt)
    (exists (?func)
        (and (Function ?func)
            (holds_function ?coreEnt ?func)))))
```

Expression 5-1 IC: Every Core Entity Holds Some Function

Core properties provide the essential building blocks for core entities. In the Foundation Layer, the gradual build-up of formal entity information semantics is achieved by exploiting a number of inter-dependent sub-theories developed in an ascending process. These sub-theories start with geometry and measure items followed by shape aspects, features and artifacts, transition features and dimensional tolerances. This particular order has been chosen because within ISO 10303 AP224 and partly CPM:

- Shape aspects are 2-D profiles which are defined using geometry and measure items.
- Shape aspects are swept along 2-D paths to produce 3-D features.
- Artifacts are made up of an aggregation of features.
- Transition features only come into existence when standard features already exist.
- Dimensional tolerances can only be meaningfully captured from the shape aspects of features that make up artifacts.

As a consequence of the detailed and extensive nature of foundation entity information semantics, only some of the pertinent examples are illustrated in this chapter. The formal semantics, alongside the corresponding IDEF5 schematics of the developed entity information semantics can be consulted in Appendix C.2.

5.2.2.2 Measure and Geometry Items

Measure and geometry items provide the intuitions towards the very basic elements of entity information semantics from which more complex core property definitions can be derived. The following intuitions apply to measure and geometry items:

- Measure items provide the semantics for the representation of measure qualities. There are two kinds of entities that have been chosen for reasoning about measure qualities namely “Length_Measure” and

"Angle_Measure". These two concepts provide the description for qualities of lengths and angles respectively.

- "Measure_Item" is an abstract kind of "Core_Property" because any instance of "Measure_Item" can only exist as a "Length_Measure" or "Angle_Measure".
- A "Length_Measure" or an "Angle_Measure" can only be meaningfully described using some real number with some attached unit of measurement.
- "Geometry_Item" is an abstract kind of "Core_Property" for which the concepts "Point", "Vector_Direction" and "Placement" are sub-classes of.
- Points and vector directions are the fundamental information elements necessary to provide a description of the placement of an entity.
- Thus, geometry items help specify the spatial description of the elements that make up features and artifacts. Points, vector directions as well as placements are characterised by spatial descriptions that involve the informal notion of X, Y and Z Cartesian axes. These axes define three mutually perpendicular imaginary planes in space.

Figure 5-4 identifies the taxonomy of the classes "Measure_Item" and "Geometry_Item", following the previously identified intuitions. Figure 5-5 then depicts a unary function "mm". This instance of "UnaryFun" has an "argProp" which is a "RealNumber" and a "returnProp" which is a "Length_Measure" (note that "RootCtx" is the namespace for the KFL meta-ontology). This implies that the "mm" function attached to a real number, for example, (mm 10), denotes an instance of the class "Length_Measure".

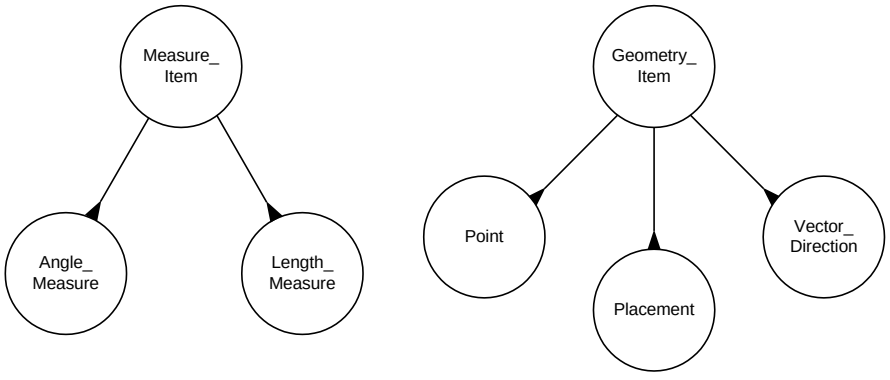


Figure 5-4 Class Hierarchy of "Measure_Item" and "Geometry_Item"

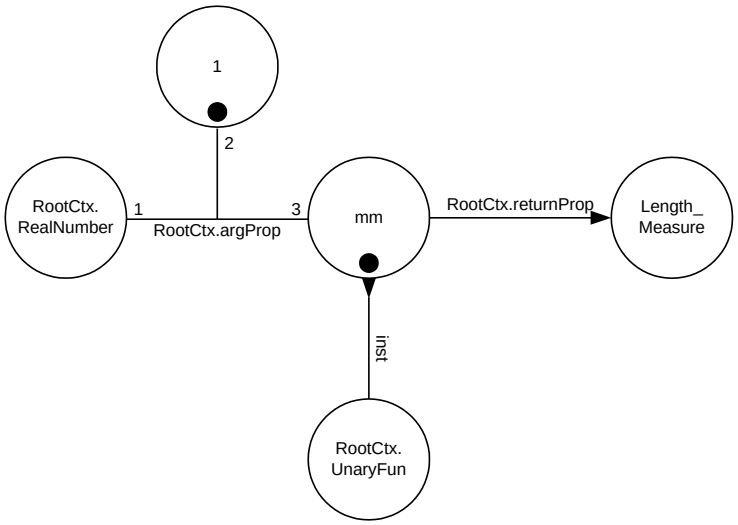


Figure 5-5 The "mm" Unary Function Used to Denote an Instance of “Length_Measure”

As a result of the expressive first order semantics of KFL, two ternary functions, “coordinates” and “direction”, have also been defined to denote instances of “Point” and “Vector_Direction” respectively. So, for example, the point given by “(coordinates (mm 10) (mm 10) (mm 10))” provides a spatial designation of a certain point with respect to the X, Y and Z Cartesian axes. One axiom related to this concept appears in Expression 5-2. The CLIF statement imposes a necessary condition that every specification of an instance of “Point” should be given by some X, Y and Z length measure coordinates. The specification of functions of the required arities is vital for capturing in an expressive and constrained way some of the lengthy structures from ISO 10303 AP224 used to capture the same intuitions.

```
(forall (?pt)
(=> (Point ?pt)
    (exists (?length1 ?length2 ?length3)
        (and (Length_Measure ?length1)
             (Length_Measure ?length2)
             (Length_Measure ?length3)
             (= ?pt (coordinates ?length1 ?length2 ?length3))))))
```

Expression 5-2 IC: Every Point Is Given by Some X, Y and Z Coordinates

5.2.2.3 Shape Aspects

A shape aspect is regarded as an entity that provides the geometric information necessary towards the creation of a feature, such as the identification of 2D shapes, which when swept along a path create 3D features (ISO 10303-224, 2006). The following intuitions apply:

- There can be several different types of entities whose semantics allow reasoning about shape aspects. In the context of this work, six fundamental types of shape aspect entities are considered namely “Circular_Closed_Profile”, “Rectangular_Closed_Profile”, “Linear_Path”, “Linear_Profile”, “Taper” and “Transition_Feature”. These kinds of shape aspects are sourced from ISO 10303 AP224.
- “Shape_Aspect” is an abstract kind of “Core_Property” from which the concepts “Circular_Closed_Profile”, “Rectangular_Closed_Profile”, “Linear_Path”, “Linear_Profile”, “Taper” and “Transition_Feature” are specialised.
- Circular closed profiles as well as rectangular closed profiles have their orientation positioned perpendicular to the centre of the profile surfaces.

Figure 5-6 indicates the taxonomy for the abstract class “Shape_Aspect”.

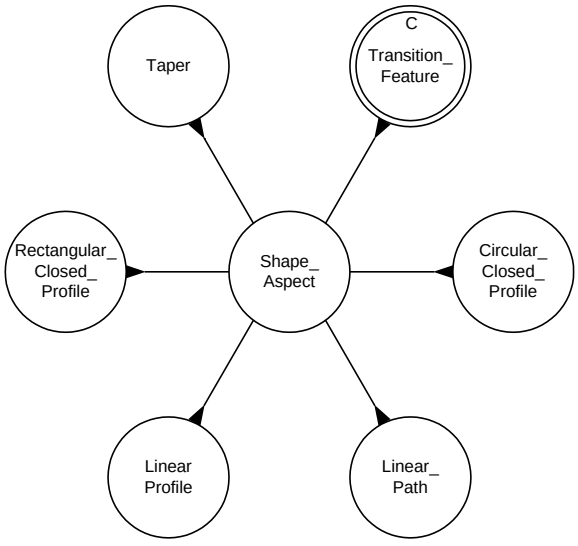


Figure 5-6 Class Hierarchy of “Shape_Aspect”

Consider the class “Linear_Profile” in the class hierarchy from Figure 5-6. The informal semantics of “Linear_Profile” state that:

- An instance ?lp of the class “Linear_Profile” is TRUE in an interpretation of the Foundation Layer if and only if ?lp is a member of a set of linear profiles. A linear profile is an open profile that involves exactly two connected points in a straight line of specified length.

In the context of this work, linear profiles are essential to provide semantic definitions linked to, for example, the axes of hole features and other basic features. To formalise the above informal semantics, two relations are specified: a ternary relation named “meets” and a binary relation named “measures”, the latter being applicable to the other sub-classes of “Shape_Aspect” as well. Figure 5-7 identifies the two relations.

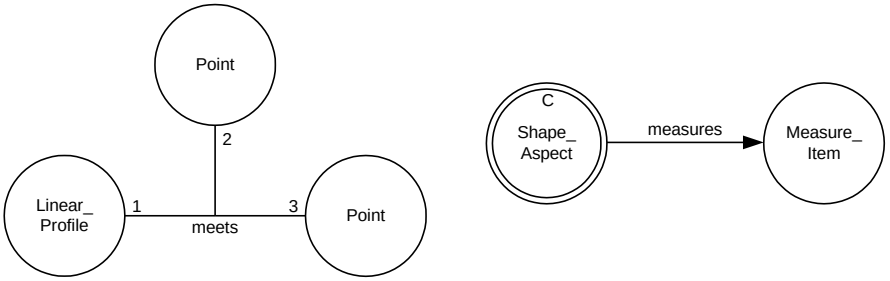


Figure 5-7 "meets" and "measures" Relations

The CLIF statements in expressions 5-3 to 5-6 (also refer to Figure 5-8) capture a majority of the foundation axioms that govern the behaviour of “Linear_Profile”. In Expression 5-3, the axiom is formulated to capture the intuition that if a linear profile ?lp “meets” two points ?pt1 and ?pt2, then ?lp also “meets” ?pt2 and ?pt1, hence the symmetry of the relation “meets”. Moreover, another axiom (Expression 5-4) involves the intuition that a linear profile ?lp cannot meet the same point ?pt twice. Hence, this implies that the necessary condition in Expression 5-5 holds in all cases, i.e. the definition of any instance of “Linear_Profile” should be followed by the identification of two distinct points that the linear profile instance “meets”.

Also, since the informal semantics state that a linear profile needs to have a specified length as a basis for its measure, this immediately conducts the importance of having Expression 5-6 as another IC. Similar chaining of ICs has been followed throughout the development of the heavyweight manufacturing ontological foundation in order to encase generic but constrained intuitions.

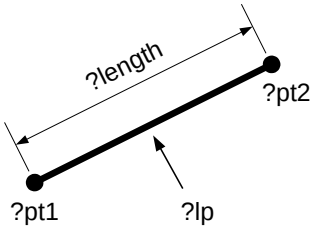


Figure 5-8 Linear Profile Semantics

```
(forall (?lp ?pt1 ?pt2)
(=> (meets ?lp ?pt1 ?pt2)
    (meets ?lp ?pt2 ?pt1)))
```

Expression 5-3 IC: The Relation "meets" is Symmetric over Linear Profiles and Points

```
(forall (?lp ?pt)
(=> (and (Linear_Profile ?lp)
         (Point ?pt))
    (not (meets ?lp ?pt ?pt))))
```

Expression 5-4 IC: The Relation "meets" is Irreflexive on Points

```
(forall (?lp)
(=> (Linear_Profile ?lp)
    (exists (?pt1 ?pt2)
        (and (Point ?pt1)
             (Point ?pt2)
             (/= ?pt1 ?pt2)
             (meets ?lp ?pt1 ?pt2)))))
```

Expression 5-5 IC: Every Linear Profile “meets” Two Distinct Points

```
(forall (?lp)
(=> (Linear_Profile ?lp)
        (exists (?length)
            (and (Length_Measure ?length)
                 (measures ?lp ?length)))))
```

Expression 5-6 IC: Every Linear Profile Has an Associated Length Measure

5.2.2.4 Features and Artifacts

An artifact is a distinct entity whether that entity is a component, part, subassembly or assembly, which can be defined in terms of the features that constitute it. Hence, a feature represents a portion or element of interest of an artifact's form (Fenves *et al*, 2004). The following intuitions apply:

- Features may have specific functions assigned to them depending on their purpose (Expression 5-2 previously explained captures this intuition).
- "Feature" is a kind of "Core_Entity" for which the chosen concepts "Round_Hole", "Cylinder" and "Block" are sub-classes of. Several other kinds of features can exist but fall outside the scope of this research.
- Round holes, cylinders and blocks as 3-D features consist of closed 2-D profiles that are swept along a 2-D linear path to produce 3-D features.
- Compound features are not considered a new categorisation of "Feature" since they consist of the aggregation of more than one simple feature. Thus, the "compound" property of a complex feature is such that the compound feature inherits its semantics from its individual constituent features.
- "Artifact" is a kind of "Core_Entity" and has its own containment hierarchy so that individual artifacts can be aggregated into more complex ones (Fenves *et al*, 2004).

Figure 5-9 illustrates the taxonomy of the "Feature" class, with two important binary relations "holds_feature" and "holds_shape" that allow artifacts to be described in terms of features, and features in terms of shape aspects, respectively. Consider the class "Round_Hole" from Figure 5-9. To capture part of the axioms governing the existence of an instance of "Round_Hole", Expressions 5-7 and 5-8 have been formulated. The logic captured in these axioms (also see Figure 5-10) imposes the necessary conditions that any specification of an instance of "Round_Hole" should be accompanied by the identification of two distinct instances of "Circular_Closed_Profile" (Expression 5-7) and one instance of "Linear_Path" (Expression 5-8) related to that instance of the "Round_Hole" through the "holds_shape" binary relation.

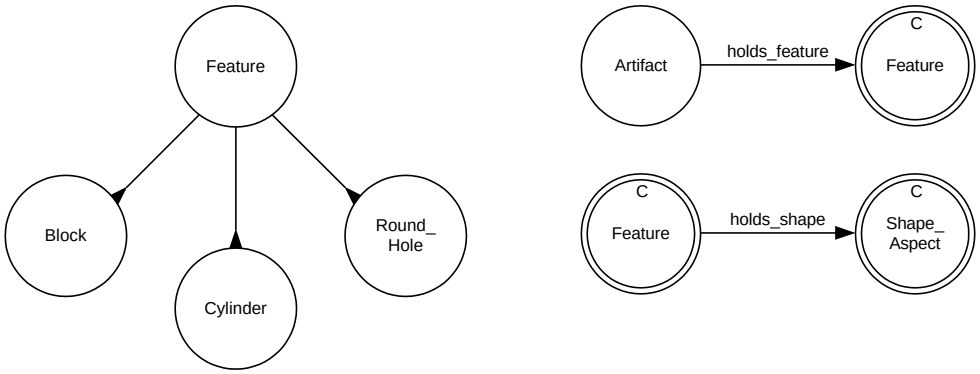


Figure 5-9 Class Hierarchy of "Feature" and Binary Relations "holds_feature" and "holds_shape"

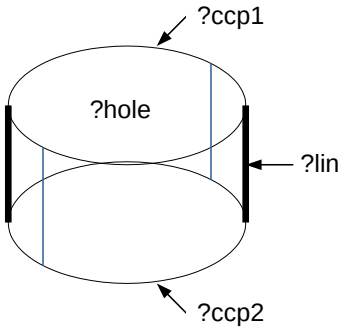


Figure 5-10 Round Hole Semantics

```
(forall (?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)))))
```

Expression 5-7 IC: Every Round Hole Feature Holds Two Distinct Circular Closed Profiles

```
(forall (?hole)
(=> (Round_Hole ?hole)
    (exists (?lin)
        (and (Linear_Path ?lin)
             (holds_shape ?hole ?lin)))))
```

Expression 5-8 IC: Every Round Hole Feature Holds One Linear Path

5.2.2.5 Transition Features

A transition feature is a kind of shape aspect that represents a transition region between two geometrically-defined faces. The main intuitions are:

- “Transition_Feature” is an abstract kind of “Shape_Aspect” from which the concepts “Constant_Radius_Edge_Round”, “Constant_Radius_Fillet” and “Chamfer” are specialised (ISO 10303-224, 2006).
- Transition features can only come into existence if proper features like cylinders and round holes already exist.
- Transition features require no orientation for placement since their positions are relative to predefined surfaces of proper features (ISO 10303-224, 2006). Hence because transition features do not exhibit the same fundamental behaviour as proper features like cylinders and round holes, this implies that transition features are essentially shape aspects.

Figure 5-11 depicts the class hierarchy of the “Transition_Feature” abstract class and one binary relation “blends” which holds between “Transition_Feature” and “Shape_Aspect”. This relation is used to capture the blending relationship that exists between transition features and shape aspects. The type of logical integrity constraints formulated for transition features follow a similar understanding explained so far in this chapter. Additionally, Appendix C.2 can be consulted for a more detailed insight into transition feature semantics.

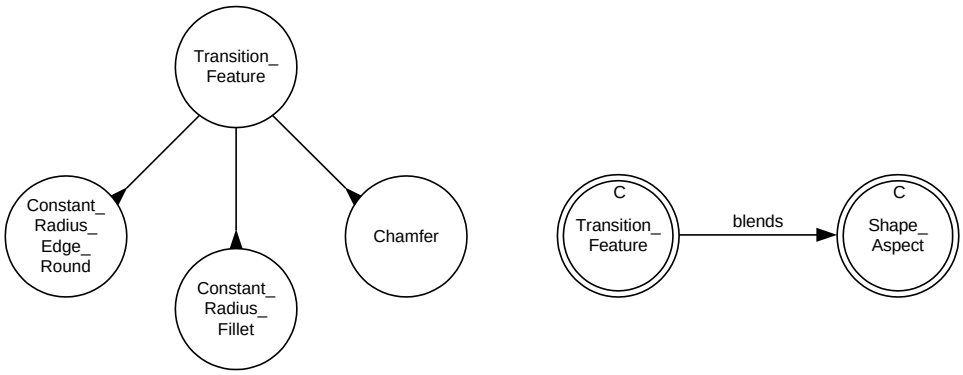


Figure 5-11 Class Hierarchy of "Transition_Feature" and Binary Relation "blends"

5.2.2.6 Dimensional Tolerances

A dimensional tolerance is the total amount a specific dimension is permitted to vary, which is the difference between maximum and minimum permitted limits of size (ISO 10303-224, 2006). The following intuitions apply to dimensional tolerances in the Foundation Layer:

- “Dimensional_Tolerance” is a kind of “Core_Property”. It does not have any further decompositions since a dimensional tolerance may be regarded as reusable element of information.
- The behaviour of a dimensional tolerance either as a size tolerance or location tolerance is dictated by the tolerance relationships that hold between shape aspects, features, measure items and dimensional tolerances.
- Tolerance values can only be meaningfully interpreted by having a lower-bound or minimum real value and an upper-bound or maximum real value, both of which are accompanied with units of measurement.

Figure 5-12 illustrates two higher-arity relations that can be used for the specification of size tolerances and location tolerances using reusable dimensional tolerance values. In the case of these two relations, the informal semantics play an important role in their interpretation at computational level.

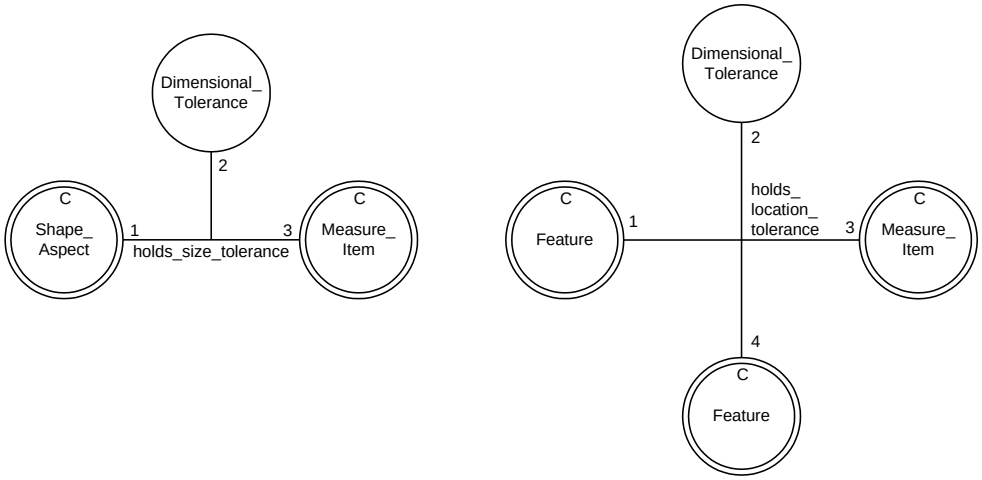


Figure 5-12 Ternary Relation "holds_size_tolerance" and Quaternary Relation “holds_location_tolerance”

The informal semantics for the “holds_size_tolerance” relation states that the relation is TRUE if and only if a shape aspect holds a given dimensional tolerance with respect to the toleranced measure item of the shape aspect. Similarly, the quaternary relation “holds_location_tolerance” is TRUE if and only if a feature holds a given dimensional tolerance with respect to the toleranced measure item, which separates the initial feature from another feature. On the other hand, an important rule in ISO 10303 AP224 regarding tolerance values is related to the value component of the lower limit being always less than that of the upper limit. To capture this fundamental intuition, Expression 5-9 has been defined. This expression imposes a constraint such that the first real number argument of the binary function “tolerance_value” is always less than the second real number argument. Thus, for example, “(tolerance_value (mm -0.1) (mm 0.1))” would be a correct instance of “Dimensional_Tolerance” while “(tolerance_value (mm 0.1) (mm 0.1))” would be incorrect and the irregularity would be flagged.

```
(forall (?dtol ?real1 ?real2)
(=> (and (Dimensional_Tolerance ?dtol)
         (RealNumber ?real1)
         (RealNumber ?real2)
         (or (= ?dtol (tolerance_value (mm ?real1) (mm ?real2)))
             (= ?dtol (tolerance_value (degree ?real1) (degree ?real2)))))
    (ltNum ?real1 ?real2)))
```

Expression 5-9 IC: The Lower-Bound Value of a Dimensional Tolerance Is Always Numerically Less Than Its Upper-Bound Value

5.2.3 Flow Objects

Most process models support the notion of input and output, which are data or objects provided to a behaviour execution before it starts, and data produced when it finishes, respectively (Bock and Gruninger, 2005). An additional set of basic concepts that hold between entities and processes has been explored, partly based on previous work performed by Bock and Gruninger (2005), in order to overcome the current limitations of PSL to relate to products inputs and outputs (Young *et al*, 2007). The following intuitions summarise the understanding behind the definition of relationships between entities and processes:

- A flow object is the property of an entity that can participate as a precondition and/or post-condition on runtime executions of activities. In other words, an object that has the property of being a flow object acts as an input and/or output on activity occurrences.
- Activity occurrences that depend on precondition entities, i.e. input flow objects, must be executed after other activity occurrences have provided these precondition entities as post-condition entities, i.e. output flow objects. An input flow object can also participate in the execution of a complex activity.
- Input and output flow objects can participate in activity occurrences that use the “min_precedes” ordering relation that provides a weaker ordering constraint, although the “next_subocc” relation can be used to provide a stronger ordering constraint as required. The two relations are introduced in the PSL Outer-Core Theory of Complex Activities.
- Entity information semantics explained in section 5.2.2 enable the explicit ontological definition of fundamental concepts relevant to mechanical products. It is obvious that during a complex activity occurrence several input and output flow objects are likely to exist. Intermediate input and output flow objects, for example, may not necessarily have explicitly-defined entity information semantics. These specific flow objects whose definitions are not explicitly captured are regarded as being implicit in nature.

Figure 5-13 depicts the fundamental nature of the intuition about explicit and implicit flow objects. In the diagram, a complex process sequence is identified, one that consists of a centre drilling operation followed by a drilling operation. A number of entities act as inputs and outputs to the subactivity occurrences within the complex process, for example, the explicitly-defined block “Object_A” is an input flow object to “Centre_Drill_Occ”. The resulting output flow object “Object_B”, whose formal representation using foundation entity information semantics is not explicitly captured (i.e. implicit), then becomes an input to “Drill_Occ”. The flow object “Object_C” which is an output from

“Drill_Occ” is an explicit object provided “Object_C” holds a complete representation using foundation entity information semantics.

Figure 5-14 identifies all the binary and unary relations defined to formalise the key participation relationships that hold between entity information and process semantics. The unary relations “flow_object”, “implicit” and “explicit” are used to differentiate between standalone objects and those that participate as inputs and outputs to activity occurrences. A full list of axioms governing these relationships is found in Appendix C.3.

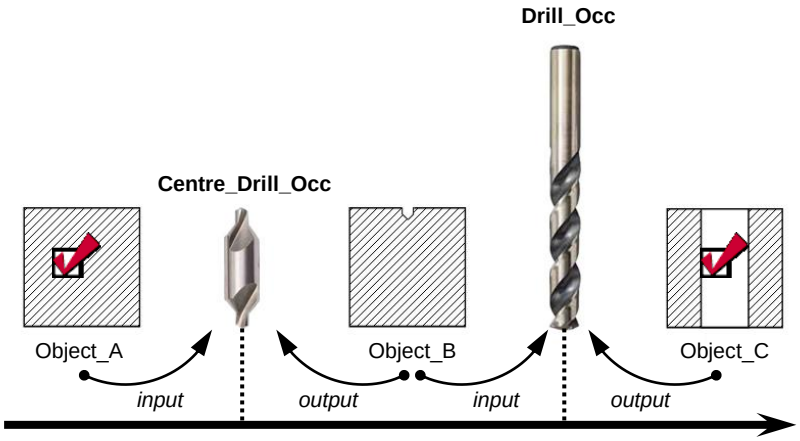


Figure 5-13 Explicit and Implicit Flow Objects

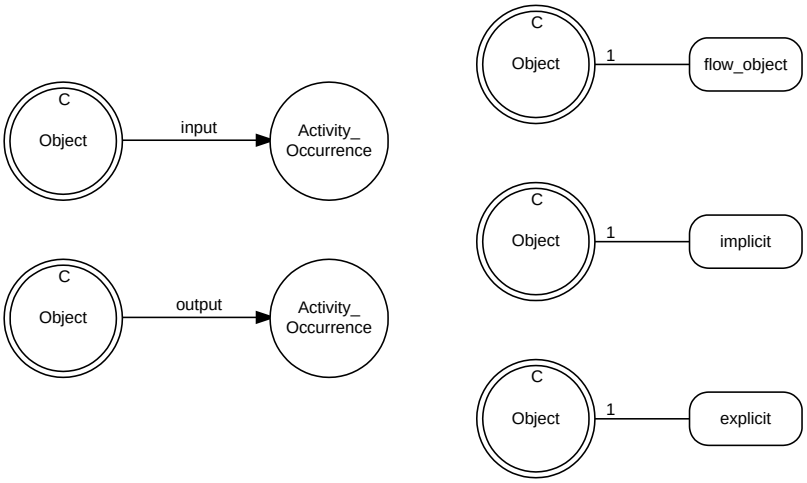


Figure 5-14 Binary Relations "input" and "output" and Unary Relations "flow_object", "implicit" and "explicit"

5.2.4 Summary of Foundation Layer

Section 5.2 of this chapter has exposed the main concepts and the intuitions exploited in order to conceptualise and formalise the Foundation Layer of the SMIF ontology-based approach. The main components of the first layer consist of:

- The expressive Common Logic-based Knowledge Framework Language (KFL).
- Concepts from PSL Core and PSL Outer-Core.
- The mergence, adaptation and improvement, from a heavyweight ontological viewpoint, of relevant object concepts originating from ISO 10303 AP224 and the Core Product Model (CPM).
- The definition of concept relationships that dictate how entities should participate in processes.

The foundation ontology approach, employed in the Foundation Layer, provides the initial vital building blocks to support the communication and interoperability requirements in product design and manufacture. Through the approach discussed in this chapter, it is clear that an integrated heavyweight manufacturing ontological foundation is a prerequisite. However, it is to be noted that the ontological foundation is multi-dimensional in nature, as it integrates different theories and combination of approaches, to help address the semantics of a range of system domains within design and manufacture.

5.3 Domain Ontology Layer

The Domain Ontology Layer is at the second level of the Semantic Manufacturing Interoperability Framework (SMIF). Reusable foundation semantics from the Foundation Layer can be specialised for the development of domain models. In essence a domain ontology classifies the most general information that characterises an entire domain (IDEF5 Method Report, 1994), where they are designed to provide common high-level knowledge related to system structures and controls and are designed for industry specific needs (Chandra and Kamrani, 2003).

It follows that in the Domain Ontology Layer, a domain model is an established view-specific model whose content is developed according to the knowledge assets, practices and preferences, terminologies and constraints that govern the domain in question. A domain shares an agreed commitment to its domain ontology. Figure 5-15 exemplifies the conceptual difference between sample concepts, coming from the heavyweight manufacturing ontological foundation, and possible domain-specific concepts that could be specialised, in a single ontology within the Domain Ontology Layer.

Figure 5-15 first identifies two entity information classes namely “Round_Hole” and “Function” **(A)** (also see figures 5-3 and 5-10) as well as the PSL-based process concepts “Activity” and “Activity_Occurrence” **(B)** respectively. Relevant semantics such as relations and ontological functions also apply to the example (here not illustrated for clarity). These sample foundation concepts are then specialised in the Domain Ontology Layer to establish new concept definitions such as “Positioning_Hole” **(C)** and to formalise domain-specific knowledge such as the machining constraints that apply to the production of positioning holes **(D)**.

In the example in Figure 5-15, the “Positioning_Hole” **(C)** concept demonstrates the prevalence of domain-assigned terminologies set with respect to the intended function of the feature concept. Similarly, the

knowledge of machining constraints on positioning holes **(D)** could potentially follow from the best practice knowledge that resides at factory level.

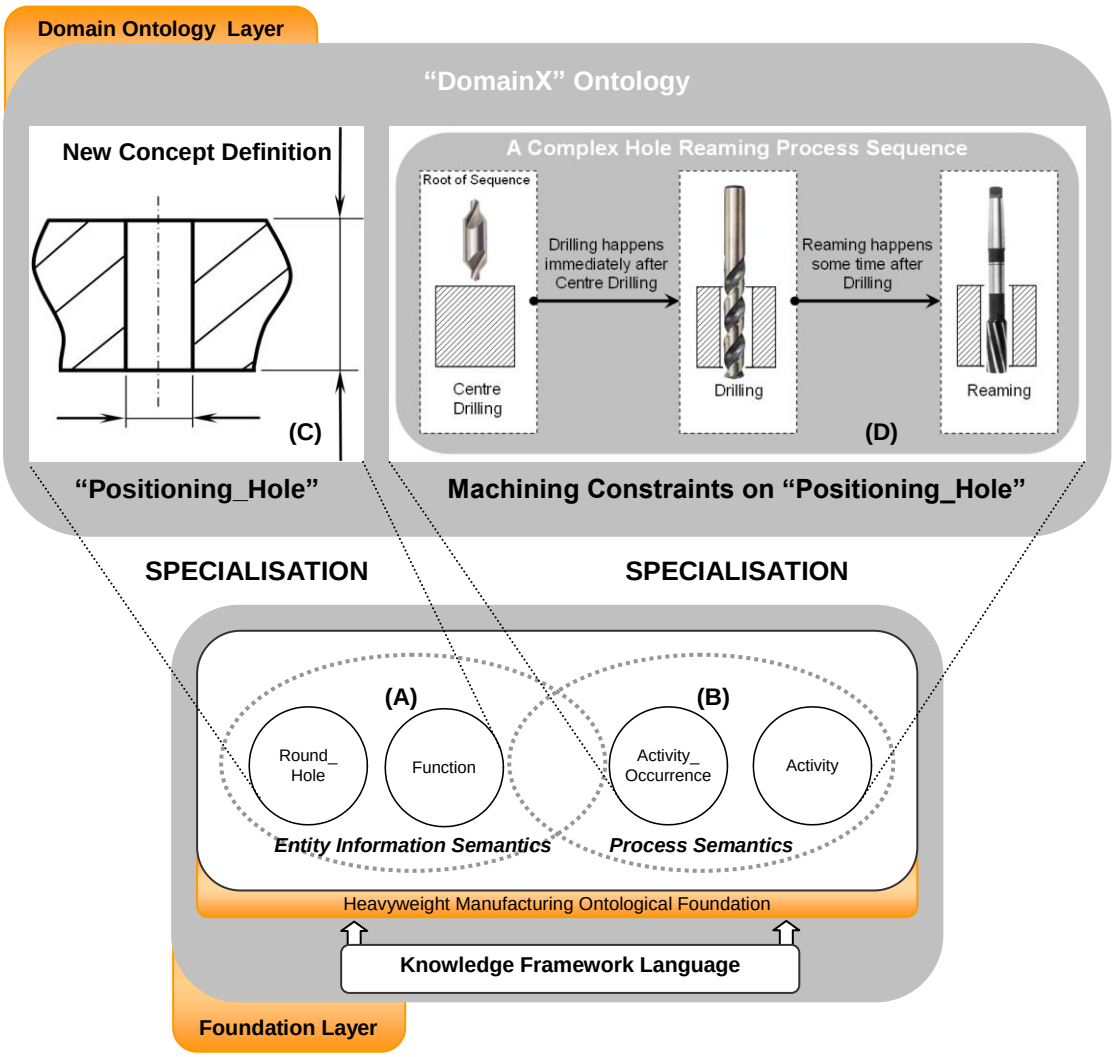


Figure 5-15 Example to Illustrate the Conceptual Difference between Foundation and Domain Concepts

The specialisation dimension between the Foundation Layer and the Domain Ontology Layer is key to the SMIF approach and consists of:

- The ontological mechanisms that allow specialisation to occur in the first place. This can be achieved through the specification of ontological relationships between foundation semantics and domain-centric semantics.

- The specification of new domain-defined integrity constraints and ontological definitions used for knowledge inference. Domain-defined integrity constraints and ontological definitions can exist as long they do not violate foundation axioms.
- The ability to instantiate domain and/or foundation concepts in the Domain Ontology Layer and use foundation and domain-defined semantics for discrete knowledge representation. For example, the specification of an instance of the class “Positioning_Hole” of known dimensions that is the output from a specific execution of a hole reaming process sequence.

A detailed account of ontology specialisation in the Domain Ontology Layer is documented next, based on the scenario introduced in Figure 5-15.

5.3.1 Domain Specialisation of Foundation Semantics

5.3.1.1 Contexts for Domain Models

In the SMIF, domain models are built “within contexts”. “Contexts” are very similar to namespaces applied to the Semantic Web. It is well known that the emerging layers of the W3C’s architecture are incorporating support for a multiple-ontology Semantic Web, founded on distributed information architecture standards such as URIs and XML namespaces for creating object identifiers that can be defined with respect to a local ontology, yet referenced globally (Hameed *et al*, 2004). Similarly, contexts for domain models in the framework have two main purposes namely:

- To distinguish between elements and attributes from different vocabularies with different meanings that happen to share the same name (Harold and Means, 2004).
- To group all related domain arguments from a single domain model together so that ontology implementation platforms can easily identify them.

During domain ontology construction, it is possible to envisage domains using concept terms, that are the same as in the heavyweight ontological manufacturing foundation, to refer to different domain concepts. Similarly, two separately-developed domain ontologies could be employing the same terms to mean different notions. At first sight this would lead to semantic reconciliation problems and matching conflicts. However, in the SMIF, because domain models are built “within contexts”, this implies that each term used to designate each argument is defined in a single context.

Using the understanding of contexts for ontologies, the two terms “Round_Hole” and “Positioning_Hole” (see Figure 5-15) are clearly disambiguated since “Round_Hole” is in fact “*Foundation*.Round_Hole” while “Positioning_Hole” is “*DomainX*.Positioning_Hole”, where “Foundation” and “DomainX” are the defined contexts for the heavyweight ontological manufacturing foundation and the domain ontology in question, respectively. Another domain ontology could be employing the term “Positioning_Hole” but the latter would avoid confusion with “*DomainX*.Positioning_Hole” as long as the context for that domain ontology be different, for example, “*DomainY*.Positioning_Hole”.

5.3.1.2 Ontological Relationships between Foundation and Domain Ontology Layers

Part of the mechanisms that allow specialisation to take place in the Domain Ontology Layer consists of three fundamental ontological relationships. The domain taxonomy (of classes and relations) can be made homogeneous and logical using the principle of specialisation through subsumption (Rector, 2003). Two subsumption relations that enable taxonomies of classes and relations to exist are: (1) super/sub-class relation and (2) super/sub-relation relation respectively. The third ontological relationship, which is not a subsumption relation, is (3) instance-of, which makes the population of facts possible through the instantiation of classes. These three ontological relations are key to the internal structure of any ontology-based model, and are thus accounted for in all meta-model ontologies such as the Ontology Works Upper

Level Ontology (Ontology Works, 2009), the Protégé knowledge model (Noy *et al*, 2000) and that of Ontolingua (Gruber, 1992).

Figure 5-16 depicts how subsumption relations may be used to specialise the "Round_Hole" and "Function" foundation classes as well as the "holds_function" foundation binary relation. The domain class "Positioning_Hole" is made a sub-class of "Round_Hole" through the "sup" relation that holds between classes. The relation "sup" is the super/sub-class relation as defined in the Ontology Works Upper Level Ontology for the KFL.

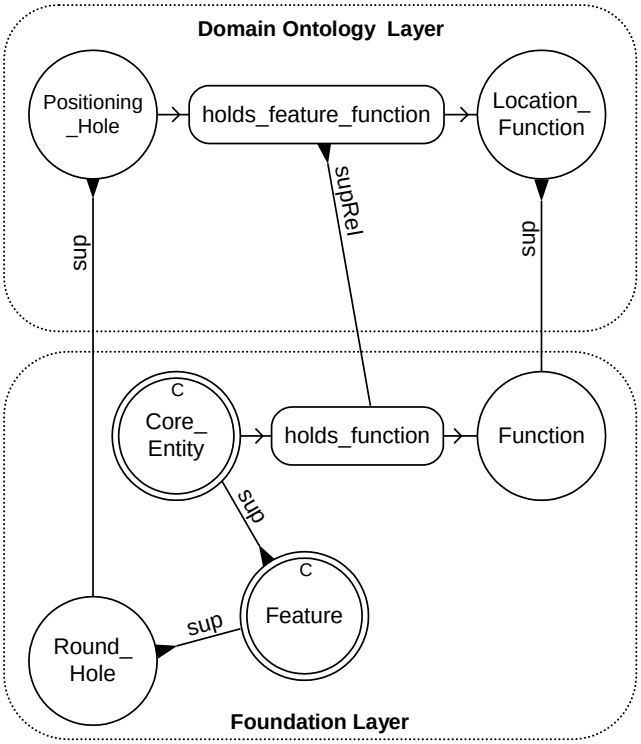


Figure 5-16 Example of Subsumption Relations between the Foundation and the Domain Ontology Layers

The "supRel" relation, also present in the KFL meta-ontology, is used to form taxonomies of relations. As can be seen in Figure 5-16, the binary relation "holds_feature_function" in the Domain Ontology Layer is a sub-relation of the foundation relation "holds_function". Note that this example does not illustrate instantiation of classes as this is treated in more detail in section 5.3.1.6 of this chapter. From the Foundation Layer, it is possible to provide the ability to enable or constrain the specialisation of domain taxonomies of classes and

relations. This consequently leads to two types of possible specialisation approaches in the SMIF namely (1) the flexible specialisation approach and (2) the controlled specialisation approach. These two specialisation approaches, explained next, have important repercussions on the capability of evaluating the interoperation between instantiated facts coming from pairs of domain models.

5.3.1.3 The Flexible Specialisation Approach

As its name suggests, the flexible specialisation approach enables domains to reuse foundation semantics without being imposed of domain ontology structural restrictions (apart from restrictions in violating foundation integrity constraints explained later in this chapter). In other words, the subsumption relations identified previously in Figure 5-16 are fully permitted as well as the declaration of instances. The consequence of creating relation taxonomies using “supRel” is a major concern to the reconciliation of instantiated facts across domain models. This is because the ability to evaluate the interoperation between cross-domain arguments at the instance level between domain models is drastically reduced.

Consider the example illustrated in Figure 5-17. Since the foundation relation “holds_function” holds between the classes “Core_Entity” and “Function” (also see Figure 5-16), this necessarily implies that “Round_Hole”, which is part of the taxonomy of “Core_Entity” is also an argument to the “holds_function” relation as shown in **(E)**. These foundation semantics are then specialised using the relations “sup” and “supRel” to form domain taxonomies of classes and relations in two domain model contexts here identified as “DomainX” and “DomainY”. For example, in “DomainX”, “Positioning_Hole” is a sub-class of “Round_Hole” while “holds_feature_function” is a sub-relation of the foundation relation “holds_function”.

Based on this specialisation scenario, suppose it is necessary to establish an inference reconciliation relation, called “Relation_XY” **(F)**, between instances of all domain-defined sub-classes of “Round_Hole” that are always

accompanied by the specification of some “Function" instance. An example of one such specification is “Hole_X holds_feature_function Location_Criteria_X” in “DomainX” while the statement of “Hole_Y holds_hole_function Position_Criteria_Y” is another similar example in “DomainY”. In order to logically replicate this specialisation scenario, which then leads to the ability to assign the “Relation_XY” **(F)** between applicable instances from both domain models, it is first required to formalise the scenario.

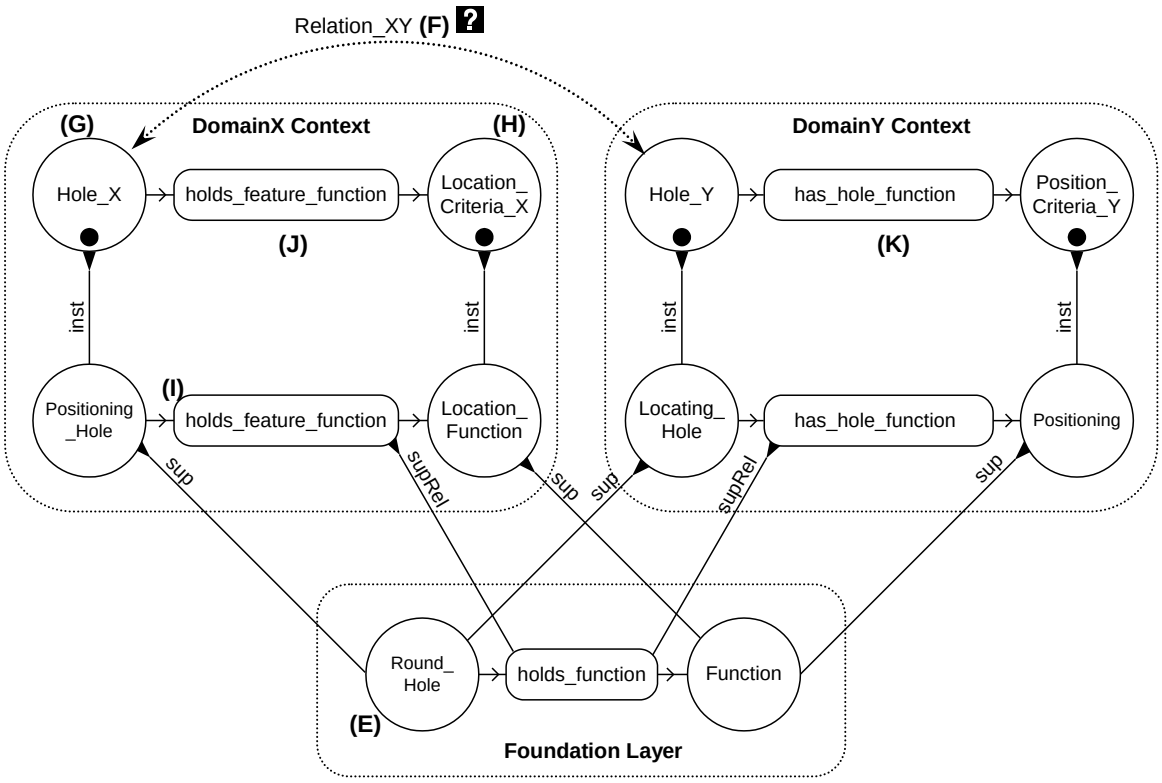
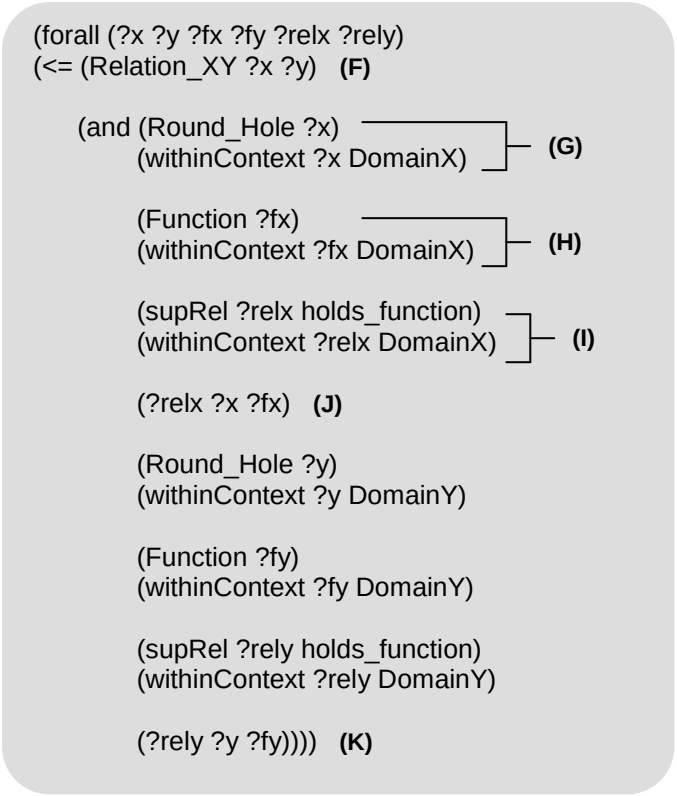


Figure 5-17 Example of the Flexible Specialisation Approach Involving Relation Subsumptions

Expression 5-10 identifies the inference axiom required for modelling the above scenario. The inference axiom has been broken down into sections in order to explain the relevance of each logic-based section in relationship to the example exposed in Figure 5-17.



Expression 5-10 Example of a Redundant Reconciliation Axiom as a Result of Unbound Relation Variables

The above expression states that the variables ?x and ?y are related through the binary relation "Relation_XY" **(F)** if and only if:

- ?x is an instance of the foundation class "Round_Hole" and is defined in the "DomainX" context and can, therefore, pertain to any sub-class of "Round_Hole" defined in "DomainX" **(G)**.
- ?fx is an instance of the foundation class "Function" and is defined in the "DomainX" context and can, therefore, pertain to any sub-class of "Function" defined in "DomainX" **(H)**.
- ?relx has the super-relation "holds_function" and is defined within the "DomainX" context **(I)**.
- ?relx is the relation that binds the instance ?x to the instance ?fx **(J)**.
- ?y is an instance of the foundation class "Round_Hole" and is defined in the "DomainY" context and can, therefore, pertain to any sub-class of "Round_Hole" defined in "DomainY".

- ?fy is an instance of the foundation class “Function” and is defined in the “DomainY” context and can, therefore, pertain to any sub-class of “Function” defined in “DomainY”.
- ?rely has the super-relation “holds_function” and is defined within the “DomainY” context.
- ?rely is the relation that binds the instance ?y to the instance ?fy **(K)**.

Although this axiom at first sight appears to be correct, it is vital to point out that there is a problem within Expression 5-10, which constitutes the primary drawback, from an ontology interoperability perspective, of enabling domain relation taxonomies. The lines **(J)** and **(K)**, i.e. (?relx ?x ?fx) and (?rely ?y ?fy) respectively, cannot be processed because of the presence of the variables ?relx and ?rely used to denote possible sub-relations of “holds_function” that become unbound in lines **(J)** and **(K)**. This inevitably occurs as a consequence of trying to capture possible relations specialisations, and prevents the desired level of deductive reasoning to be reached. Deduction (deductive reasoning) in this case refers to the process of reaching a conclusion on the basis of some given premises (Markovits, 2004), and is a fundamental part of logical reasoning. Hence, this example identifies the inference issues at the instance level arising from the creation of relation taxonomies in domain ontologies.

5.3.1.4 The Controlled Specialisation Approach

The controlled specialisation approach overcomes the issue of ontology interoperation at the instance level. By restricting domain models from specialising foundation relations, it is possible to carry out deductive reasoning at the instance level, across the KBs of domain models. Expression 5-11 depicts an integrity constraint which can be added to the heavyweight manufacturing ontological foundation to prevent domains from creating relation subsumptions and relation taxonomies.

```
(forall (?rel)
(=> (and (Relation ?rel)
         (not (Property ?rel))
         (withinContext ?rel Foundation))
         (not (exists (?subrel)
                  (and (Relation ?subrel)
                       (supRel ?subrel ?rel))))))
```

Expression 5-11 IC: Subsumptions Involving Foundation Relations Are Not Permitted

Expression 5-11 informally states that if there is purely a relation ?rel, where ?rel is defined within the “Foundation” context, then no specification of a sub-relation of ?rel identified as ?subrel is meant to exist. This integrity constraint immediately imposes a structural constraint at the Domain Ontology Layer. This constraint is portrayed in Figure 5-18, where it can clearly be discerned that the “holds_function” foundation relation is used as-is in both “DomainX” and “DomainY”. Consequently, a deductive reconciliation axiom can be written (see Expression 5-12) with the intention of reconciling all the instances of “Round_Hole” defined in “DomainX” and “DomainY” that happen to hold some function, for example, “Hole_X holds_function Location_Criteria_X” in “DomainX” and “Hole_Y holds_function Position_Criteria_Y” in “DomainY”.

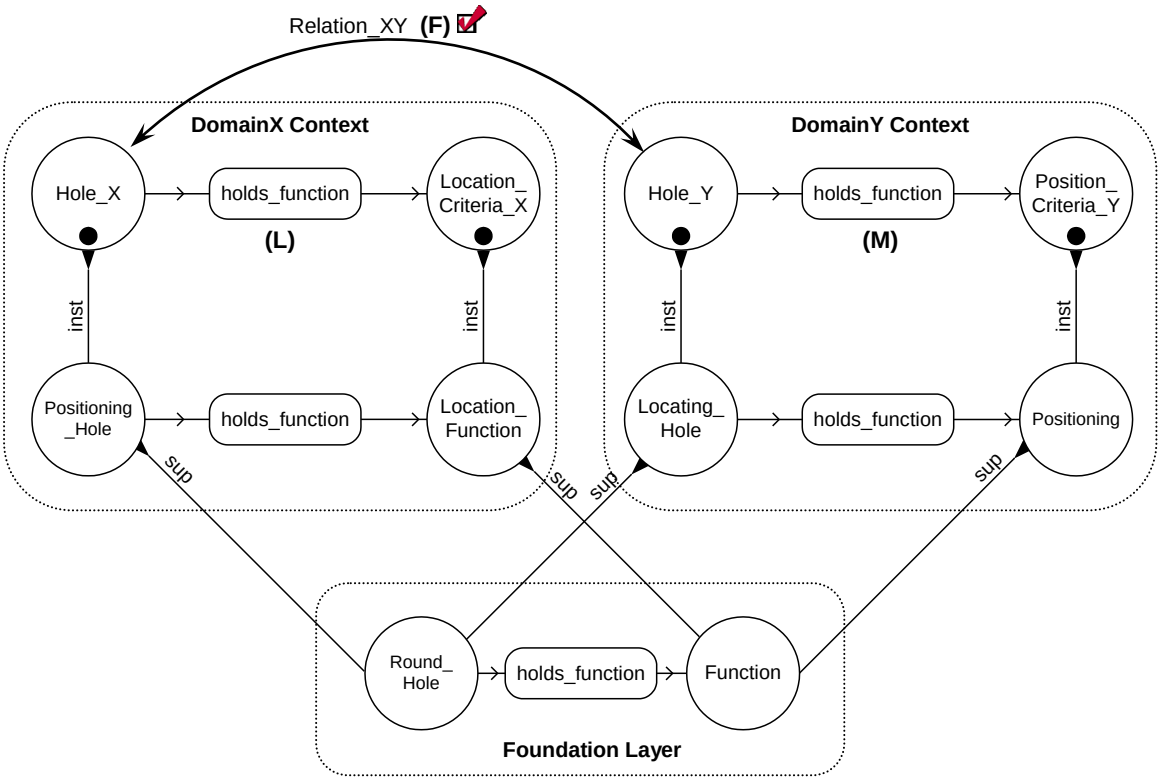


Figure 5-18 Example of the Controlled Specialisation Approach

```
(forall (?x ?y ?fx ?fy)
(<= (Relation_XY ?x ?y)   (F)

    (and (Round_Hole ?x)
         (withinContext ?x DomainX)
         (Function ?fx)
         (withinContext ?fx DomainX)

         (holds_function ?x ?fx)   (L)

         (Round_Hole ?y)
         (withinContext ?y DomainY)
         (Function ?fy)
         (withinContext ?fy DomainY)

         (holds_function ?y ?fy))))   (M)
```

Expression 5-12 Example of a Deductive Reconciliation Axiom

Expression 5-12 remains somehow similar to Expression 5-10. However, the difference lies in lines **(L)** and **(M)** where instead of having variables to denote relations, the known foundation relation “holds_function” is present. The arguments to the “holds_function” relation are also obvious, for example, line **(L)** comprises (holds_function ?x ?fx), meaning ?x is the first argument to the relation “holds_function” and ?fx is the second argument to the same relation, where it is known that ?x and ?fx refer to some instance of “Round_Hole” and some instance of “Function” in “DomainX” respectively. This understanding also applies to line **(M)**. Hence, Expression 5-12 is well-formed and for this reason, the controlled specialisation approach provides a way for enabling cross-domain inferences to be performed at the instance level of domain models.

5.3.1.5 Integrity Constraints and the Domain Ontology Layer

One of the features of integrity constraints (ICs), as a means to embed foundation ontological axioms as prescriptions to complement semantic knowledge (Mäs *et al*, 2005), has previously been exposed (see section 5.2). In addition to this, ICs also have a direct influence on the semantic conformance of domain models that are developed in the second layer of the SMIF. ICs ensure that the completeness of the heavyweight ontological

manufacturing foundation as a logical theory is met. As a consequence of domain models being specialised directly from foundation semantics, foundation ICs ascertain that the soundness in semantics is conveyed to domain-defined arguments too.

Consider the example shown in Figure 5-19, where a foundation IC is present in order to detect incorrect or incomplete specifications involving the binary relation “holds_function”. The IC is written in KFL and is appended in line **(U)** with a textual statement that reads: “The holds_function relation only holds between core entities and functions”. Note the “:IC soft” declaration at the beginning of line **(U)**, which is a KFL-permitted declaration. Suppose in one domain ontology the “holds_function” relation is specialised to “holds_feature_function” and the latter is asserted as being a binary relation whose arguments involve an incorrect class definition (see Figure 5-19 “Not a Core_Entity Class”). On loading the domain ontology, the loading process would be prevented because of infringements against the class argument declarations to the relation “holds_function” as well as the presence of the “:IC soft” declaration.

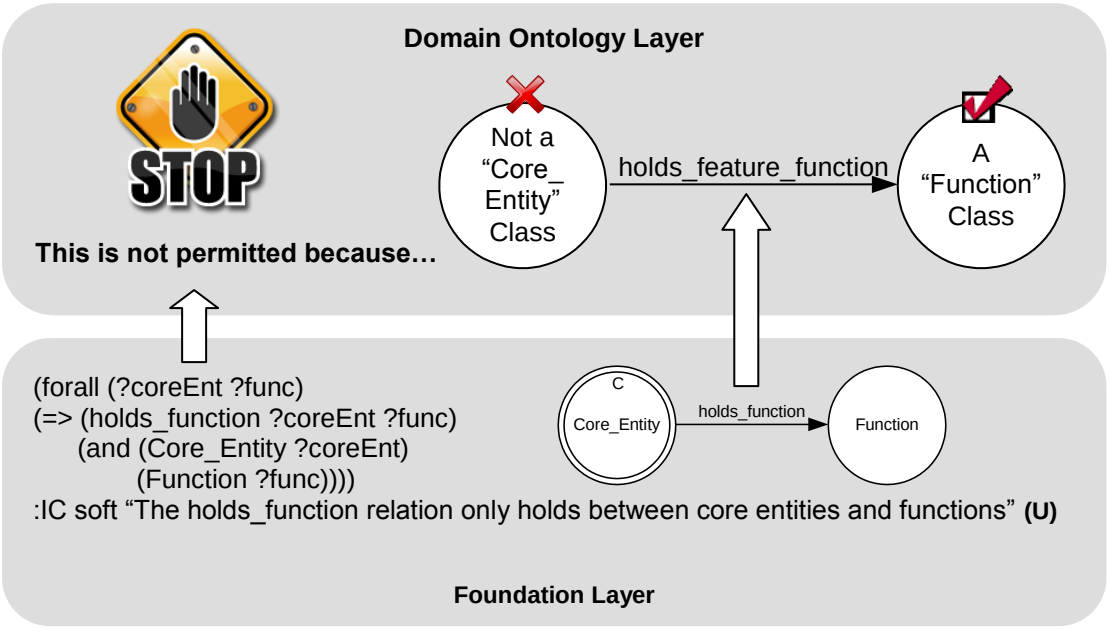


Figure 5-19 Example of an Integrity Constraint Violation

During the domain ontology loading process, the knowledge engineer is made aware of the nature of the infringement, thereby prompting a rectification action to be proceeded with. 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. On the other hand, a hard IC completely prevents a wrong action from being committed. An adamant IC is one which indicates a necessity and is destined to be used for the functioning of Ontology Works’ Upper Level Ontology system.

The simple example illustrated in Figure 5-19 demonstrates one of the important potentials of ICs. ICs extend a conceptual model (in this case the heavyweight ontological manufacturing foundation) to make the model precise and capable to ensure that domain semantics is rightly expressed (Halpin, 1999). In a similar way, ICs are intended to ensure correctness, consistency support and checking of data (instances) in the implementation in a database (in this case a domain KB) (Pakalnickiene and Nemuraite, 2007). Therefore, this advantage of ICs contributes positively early on during the domain ontology development phase and further downstream during instantiation and commitment of fact statements to a domain KB.

It is also to be pointed out that domain ontologies are able to formulate domain-specific ICs, provided these are in line with foundation ICs. As an example to illustrate this facet, consider Figure 5-20. Recall from Expression 5-1 that there is a foundation IC present in order to ensure that “every core entity holds some function” **(V)**. Based on the scenario previously exposed in Figure 5-18, it becomes possible to specialise the expression **(V)** in order to establish an IC in “DomainX”. For example, expression **(W)** in Figure 5-20 captures the semantics that “every positioning hole holds exactly one location function”. Expression **(W)** consistently follows from expression **(V)** and is, therefore, a completely suitable IC capable being declared in “DomainX”.

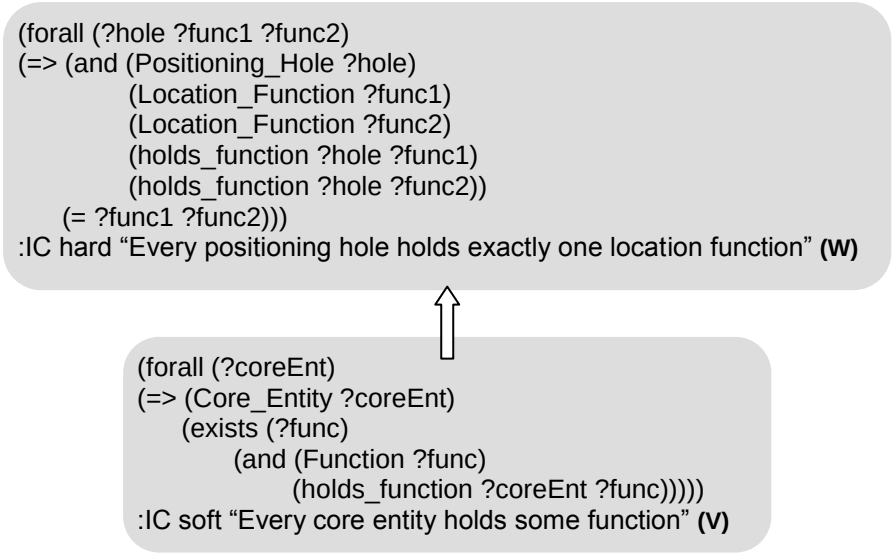


Figure 5-20 Example of a Consistently-Defined Domain Integrity Constraint

5.3.1.6 Instantiation and Discrete Knowledge Representation

Instantiation often becomes an important process after the domain ontology development phase is completed. The instance layer is the commitment layer which is concerned with the composition, constraining and instantiation of lexons (a fact type of some category or description, for example, a class) to represent the semantics of a particular fact (instance) (Pretorius, 2004). In other words, individual instances are the most specific concepts represented in a Knowledge Base (KB) (Noy and McGuinness, 2001).

It is necessary to emphasise that, from the point of view of this work, there exists a fine line between an ontology as a logical theory and a Knowledge Base (KB). An ontology aims to capture the conceptual structures of some field while a KB aims to specify a concrete state of the field, i.e. an ontology consists of intensional logical definitions (characteristics that distinguish concepts) while a KB comprises of extensional parts (instances) (Pretorius, 2004). A KB, therefore, may be regarded as being a form of database dedicated to the effective management of knowledge which is facilitated through the classification and constraining mechanisms coming from the domain ontology to which the KB is associated with. Thus, the structure of the

KB relies on semantic structures established in an ontology. For example, machine-readable KBs store knowledge in a computer-readable form where an ontology can be used to define the structure of the stored data (Wikipedia, 2009).

Figure 5-21 illustrates how through instantiation, discrete knowledge pertaining to a complex reaming process execution sequence can be represented. The example takes into account the instantiation of sample foundation semantics (primarily PSL-based process semantics) employed in the Domain Ontology Layer to formalise a complex hole reaming process sequence as a machining constraint on the production of an instance of the class “Positioning_Hole” (also refer to **(C)** and **(D)** on Figure 5-15 if needed). The example uses the controlled specialisation approach, where relation subsumptions are not allowed.

Part of the instance file which contributes to encoding the instance knowledge within “DomainX” in Figure 5-21 is captured in Expression 5-13. Instance files are required in the Domain Ontology Layer whenever facts, i.e. instances, have to be populated in the KB of domain models. Instance files are written in Simple Common Logic (SCL) (Kendall *et al*, 2004), which is very similar to the Common Logic Interchange Format (CLIF), except that SCL instance files used in the Domain Ontology Layer are dedicated to the population of instances rather than the manipulation of an ontology’s logical theory.

Based on the example illustrated in Figure 5-21 and Expression 5-13, it is shown that there are essential components that influence the instantiation of facts in order to capture domain instance knowledge. These components include:

- The specification of instances, for example, in **(N)**, “*DomainX*.Make_Hole” is an instance of the foundation class “Activity” identified as “Make_Hole” in the “DomainX” context. Note that the specification of instances should always be accompanied by the context, for example, “*DomainX*.Make_Hole”, for term disambiguation.

- The use of relations to bind instances together in order to create fact sentences, for example, in **(O)**, the instance “Make_Hole_X” is linked to the instance “Make_Hole” via the foundation binary relation “occurrence_of”. This understanding also applies to **(P)**, **(Q)**, **(R)** and **(T)**. Note that **(Q)** and **(R)** in Expression 5-13 are exploited to provide part of the semantics of the process sequence in Figure 5-19, while **(T)** is used to capture the knowledge that the instance “Hole_X” **(S)** is an output from the complex hole reaming activity occurrence “Make_Hole_X”.

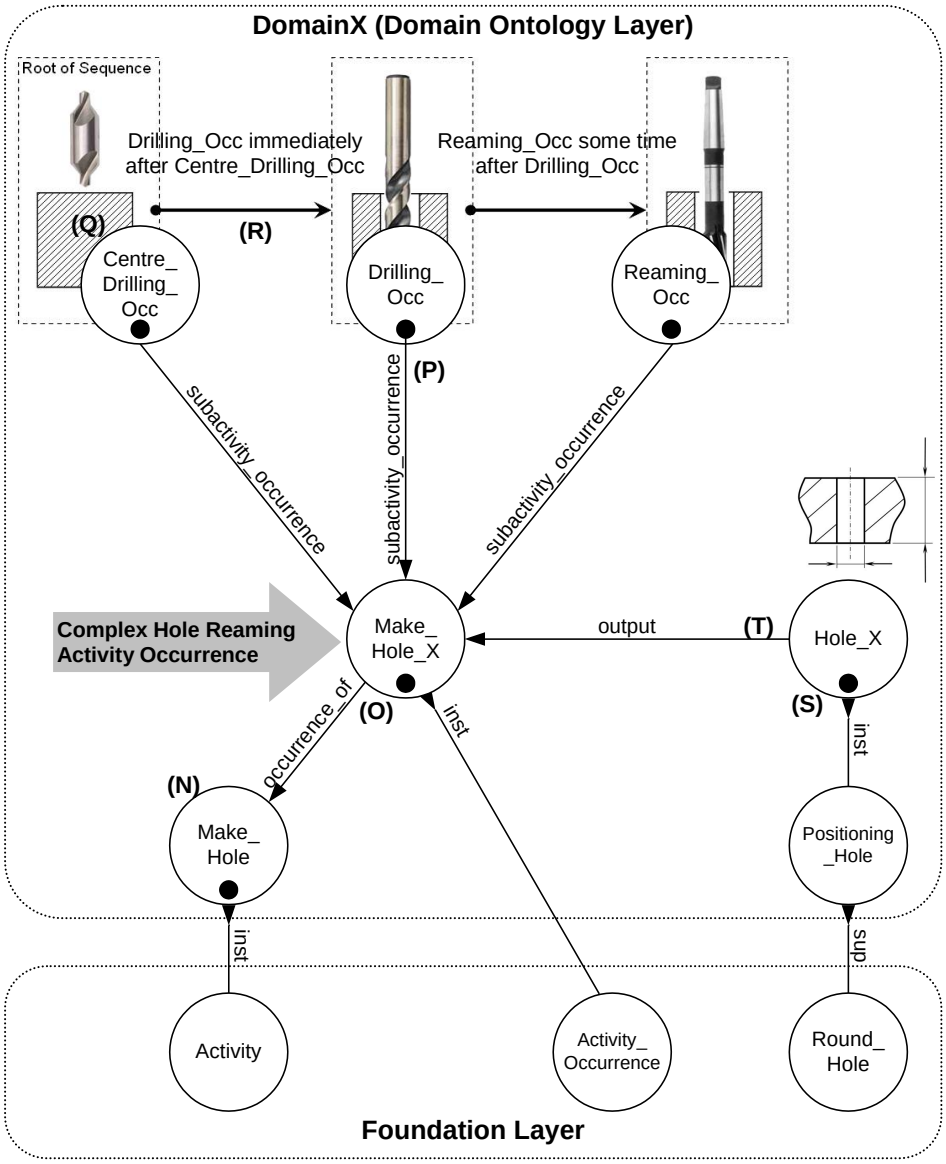


Figure 5-21 Example of Instantiation and Discrete Knowledge Representation in the Domain Ontology Layer

```
;;; DEFINE ACTIVITY INSTANCES

e.g. (Foundation.Activity DomainX.Make_Hole) (N)

;;; DEFINE ACTIVITY OCCURRENCE INSTANCES

e.g. (Foundation.Activity_Occurrence DomainX.Make_Hole_X)
e.g. (Foundation.Activity_Occurrence DomainX.Centre_Drilling_Occ)

;;; RELATE ACTIVITIES TO ACTIVITY OCCURRENCES

e.g. (Foundation.occurrence_of DomainX.Make_Hole_X
      DomainX.Make_Hole) (O)

;;; DEFINE SUBACTIVITY OCCURRENCE RELATIONSHIPS

e.g. (Foundation.subactivity_occurrence DomainX.Drilling_Occ
      DomainX.Make_Hole_X) (P)

;;; DEFINE PROCESS SEQUENCE RELATIONSHIPS

e.g. (Foundation.root_occ DomainX.Centre_Drilling_Occ
      DomainX.Make_Hole_X) (Q)
e.g. (Foundation.next_subocc DomainX.Drilling_Occ
      DomainX.Centre_Drilling_Occ DomainX.Make_Hole) (R)

;;; DEFINE POSITIONING HOLE INSTANCE

e.g. (DomainX.Positioning_Hole DomainX.Hole_X) (S)

;;; DEFINE OUTPUT RELATIONSHIPS

e.g. (Foundation.output DomainX.Hole_X DomainX.Make_Hole_X) (T)
```

Expression 5-13 Example of Part of an Instance File Written in Simple Common Logic (SCL)

The nature of instantiation elucidated in the previous example reveals the way in which the formalisation of discrete knowledge (such as the known machining constraints on a specific “Hole_X” in “DomainX”) is performed. Instances can be committed to the KB of a domain model as long as both foundation and domain integrity constraints are not violated. During the commitment transaction of facts to the KB, ICs ensure that the knowledge engineer always supplies correct and complete instance knowledge.

5.3.2 Summary of Domain Ontology Layer

Section 5.3 of this chapter has documented the essential principles adopted, in Domain Ontology Layer, to allow domain models to be developed from the heavyweight ontological manufacturing foundation. Domain model specialisation typically consists of:

- Ontological relationships and interactions between concepts from the Foundation and Domain Ontology layers.
- The declaration of domain-specific integrity constraints, which need to remain coherent with foundation semantics.
- The definition of instances in order to capture instance knowledge in the form of facts (instances) and sentences that relate facts together.

It has also been shown that there are two directions in which domain models could be specialised. The specialisation process can either be flexibly carried out or performed in a controlled manner. A simple understanding can be applied regarding the suitability of specialisation processes to specific domains. For example, if the main purpose of a domain ontology is to focus on the capture of domain concepts and constraints as a logical theory, and does not involve the population of instances, then the flexible specialisation approach is as convenient as the controlled approach.

However, if discrete knowledge representation is a significant aspect of a domain model, alongside the representation of domain concepts and constraints, then the controlled specialisation approach is preferred from an ontology interoperability viewpoint. Moreover, since relation specialisations are not allowed at domain level following the controlled specialisation approach, this implies that relation mismatches between domain models are avoided. Relation mismatches typically involve dissimilarities in the definition of relations, the way in which these are attributed and used to structure classes in disparate ontologies (Hameed *et al*, 2004; Chungoora and Young, 2008b). In this work, the controlled specialisation approach has been chosen

in order to explore higher capabilities for the reconciliation of instances and also to limit relation mismatches.

5.4 Summary

This chapter has explained the nature and implications of the first two layers of the SMIF, thereby satisfying part of the third objective of this work concerned with the exploration of concepts related to the framework (see Chapter 1 section 1.3.1). In order to develop the rigorous heavyweight ontological manufacturing foundation in this research, informal intuitions about entity information and processes need to be formalised. This formalisation process necessitates the accurate definition of heavyweight semantics, involving basic ontological concepts such as classes and relations backed with the efficient declaration of integrity constraints. Moreover, IDEF5 schematics have been used in order to visually explore the primary semantic structures within the heavyweight manufacturing ontological foundation (also see Appendix C).

With the heavyweight ontological manufacturing foundation in place, it then becomes possible to develop domain models in the Domain Ontology Layer. This process is enabled via ontology specialisation mechanisms such as the definition of domain contexts, the specification of subsumption relationships, and the definition of concept instances for the capture of discrete domain knowledge. Furthermore, because the Domain Ontology Layer interacts with the Foundation Layer, this implies that the benefits arising from the definition of foundation integrity constraints are passed on to the Domain Ontology Layer to ensure the integrity-driven specialisation of domain models. The interactions between the Foundation and Domain Ontology layers identified in this chapter have been experimentally tested and applied to a number of test cases in Chapter 8.

6 Semantic Reconciliation and Interoperability Evaluation Layers

6.1 Introduction

This chapter discusses in more detail the understanding behind the Semantic Reconciliation and Interoperability Evaluation layers of the Semantic Manufacturing Interoperability Framework (SMIF). Section 6.2 and its sub-sections focus on how a stepwise semantic reconciliation process is achieved through the application of ontology mapping process concepts to reconcile pairs of domain models developed in the Domain Ontology Layer. One fundamental stage of the ontology mapping process comprises semantic alignment, which relies on logic-based definitions of semantic mapping concepts. The different modes in which semantic mapping concepts occur are also further explained.

The automated association of semantic mapping concepts in the Semantic Reconciliation Layer provides a basis for evaluating and verifying the possible correspondences between cross-domain arguments. Section 6.3 documents the main mechanisms used for the evaluation and verification process carried out in the Interoperability Evaluation Layer. These mechanisms enable the formulation of interoperable knowledge queries. As a result of the complexity involved in constructing several interoperable knowledge queries, a method is then identified in order to assist knowledge querying procedures in order to maximise reusability of interoperable knowledge queries at the fourth level of the SMIF.

6.2 Semantic Reconciliation Layer

Following the framework approach, several collaborating domain models of feature-based design and manufacture are bound to exist in the Domain Ontology Layer. In the event that these domain models need to interoperate with the intention of sharing knowledge, domain semantics need to be

reconciled. The Semantic Reconciliation Layer covers applied ontology-based techniques relevant to enabling the reconciliation of domain semantics.

These techniques employ segments of known ontology matching methods such as (1) the computation of contexts for domain ontologies (Stumme and Maedche, 2001), (2) ontology merging (Noy and Musen, 2003) and (3) semantic alignment (Euzenat and Shvaiko, 2007). However, unlike known ontology matching methods, the combined approach exploited at the Semantic Reconciliation Layer provides a unique way to the reconciliation of domain semantics. This takes into account:

- Cross-domain arguments that may share the same terms but are semantically different, since from a semantic interoperability viewpoint, term similarity does not necessarily imply equivalence.
- A progression towards heavyweight Common Logic-based semantic alignment processes, to reinforce current knowledge on semantic alignment and the related methods to verify the integrity of cross-domain mappings.
- Interoperation at the instance level of domain models made possible through the controlled specialisation approach, an aspect which until now has remained problematic to the ontology mapping community.

Figure 6-1 illustrates the basic concepts involved in the mapping of domain models at the Semantic Reconciliation Layer. The process of semantic reconciliation can be performed between pairs of models at a time, as can be encountered with almost all current ontology mapping frameworks and methodologies (Kalfoglou and Schorlemmer, 2003). Ontology mapping process concepts involve a first stage of adjusting the contexts (namespaces in this case) of two domain models which are to be reconciled. Following this stage is a simple ontology merging process, where both models are loaded intact into a single Foundation Layer. The last procedure in the ontology mapping process is that of semantic alignment, where semantic mapping

concepts are loaded into the merged models. Semantic mapping concepts are further discussed in section 6.2.2.

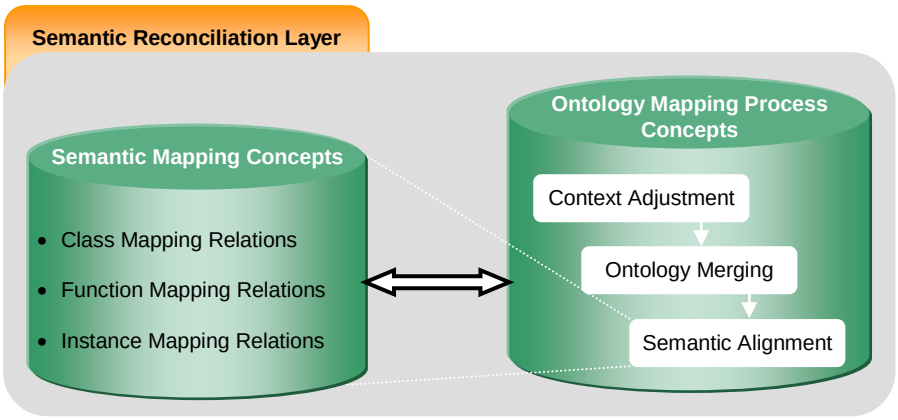


Figure 6-1 Concepts Explored in the Semantic Reconciliation Layer

6.2.1 Ontology Mapping Process Concepts

6.2.1.1 Domain Context Adjustment Process

Adjusting the contexts of pairs of domain models to be reconciled forms part of the initial stage of semantic reconciliation. It is relatively easy to understand the reason behind the adjustment of the contexts of two domain models to two standard contexts, which resembles the computation of contexts adopted in the FCA-Merge ontology merging method (Stumme and Maedche, 2001), but is simpler. From the preferences established in this research, any two domain models to be reconciled have their contexts adjusted to the standard contexts “DomainX” and “DomainY”.

For example, suppose there are two domain models which need to interoperate and one uses a context called “Design” while the other uses a context called “Manufacture”. Following the approach of domain context adjustment, the “Design” context would be renamed to “DomainX” and the “Manufacture” context to “DomainY”, or vice versa, where “Design” is renamed to “DomainY” and “Manufacture” to “DomainX”. In short, any two domain contexts are renamed to either “DomainX” or “DomainY” as standard contexts. Context adjustment also needs to be performed for instance files

pertaining to each domain model, if reconciliation needs to be carried out at the KB level.

The context adjustment procedure is important because the semantic alignment process, which takes place later on during ontology mapping, involves semantic mapping concepts based around the two predefined contexts “DomainX” and “DomainY”. The process of context adjustment is straight forward and only requires a substitution of the names of domain model contexts. Figure 6-2 captures this understanding.

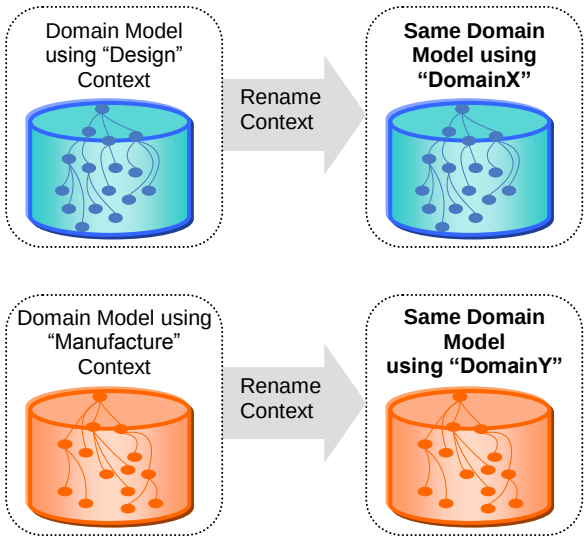


Figure 6-2 Adjusting Domain Contexts to Standard Contexts "DomainX" and "DomainY"

6.2.1.2 Simple Ontology Merging Process

The second stage in the ontology mapping process is concerned with a simple ontology merging procedure, which uses the domain models with their adjusted contexts and loads both in a single Foundation Layer. During this simple ontology merging process, all domain arguments present in “DomainX” and “DomainY” remain distinct to each domain model. The merging process also applies to the instances adjusted to “DomainX” and “DomainY”, if these instances exist. Figure 6-3 identifies the consequence of merging two domain models under the simple ontology merging process adopted.

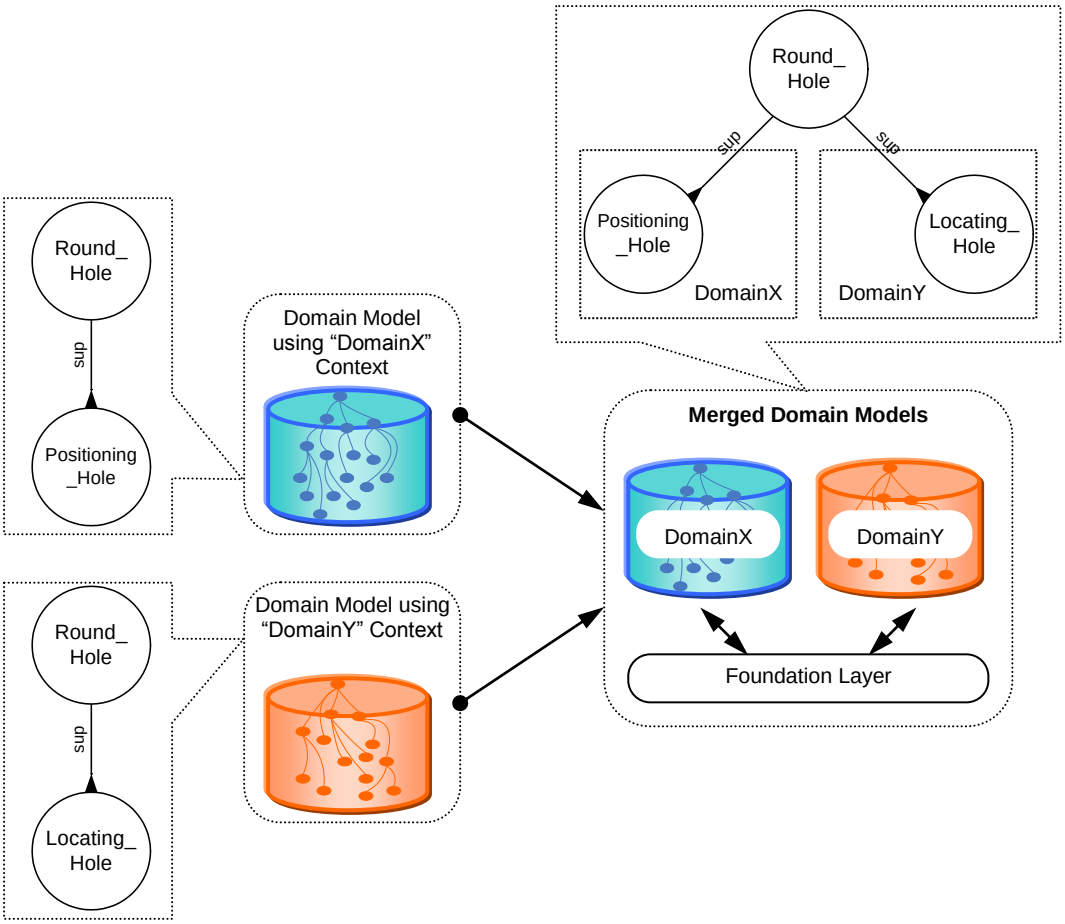


Figure 6-3 Simple Ontology Merging Process

The figure first illustrates two specialisations of the foundation class “Round_Hole” that are defined separately in the “DomainX” and “DomainY” contexts. Under the simple merging process, all ontology-based content from both domain models are brought under one platform. Notice in Figure 6-3 how during merging, the two classes “Positioning_Hole” and “Locating_Hole” stay distinct to their contexts but both still appear as specialisations of “Round_Hole”.

It is to be noted that the ontology merging process is referred to as being simple, on the basis that no similarity computation is made during the mergence of both domain models unlike other dedicated ontology merging approaches such as FCA-Merge (Stumme and Maedche, 2001), the system developed by Fernández-Breis and Martínez-Béjar (2002) and PROMPT (Noy and Musen, 2003). The procedure here simply ensures that all domain model

content is captured in a single environment prior to the semantic alignment process.

6.2.1.3 Semantic Alignment Process

The semantic alignment process is at the heart of the Semantic Reconciliation Layer and is illustrated in Figure 6-4. The alignment process is enabled by feeding semantic mapping concepts to the merged models (see section 6.2.2 for a description of semantic mapping concepts). Based on the heavyweight logical conditions that define these semantic mapping concepts, mapping relations may become associated to cross-domain arguments, if during the merging of a model in “DomainX” and another model in “DomainY”, there exist arguments from both contexts that happen to satisfy these logical conditions. The semantic alignment process is almost entirely automatic.

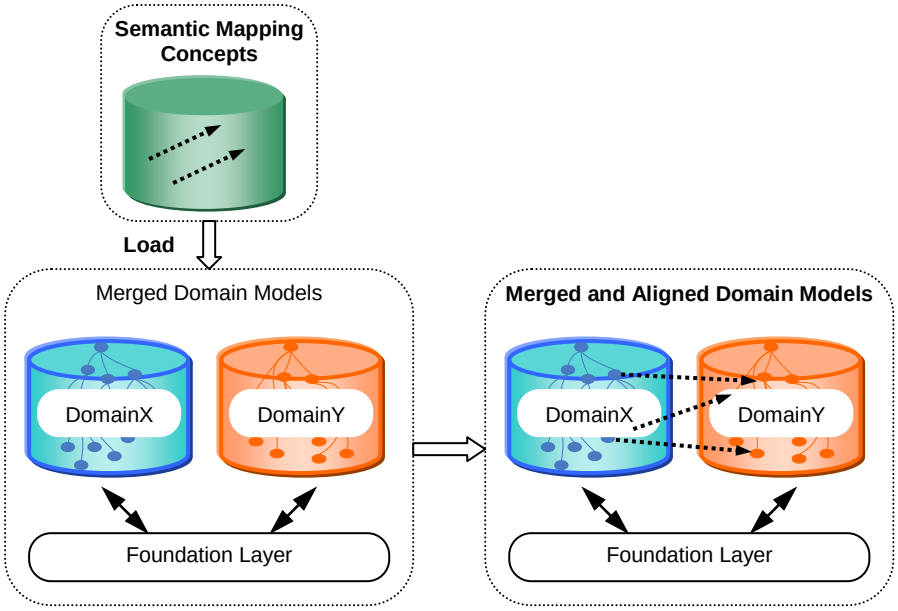


Figure 6-4 Semantic Alignment Process

The view on the semantic alignment process exposed in this work falls under the category of tools responsible for the discovery of mappings between two domain models. This is performed by finding pairs of related arguments, through the process of alignment and the reconciliation of specific portions of two domain models through an intermediate articulation ontology (in this case the Foundation Layer). It has been acknowledged that alignment and

articulation mapping processes are related in the sense that binary relations can be used to align two ontologies. These binary relations can themselves be decomposed into a pair of functions emanating from a common intermediate source where the intermediate ontology serves as the articulation ontology (Kalfoglou and Schorlemmer, 2003).

6.2.2 Semantic Mapping Concepts

A semantic mapping concept in the Semantic Reconciliation Layer consists of a formally-defined semantic mapping relation (using logic programming) and written informal remarks that accompany the relation. A semantic mapping relation binds two cross-domain arguments (such as classes) when certain logical conditions, that define the semantic mapping relation, become true between these arguments.

In addition to these formal semantics, semantic mapping concepts also include the statement of informal remarks for human interpretation. This is because alignments produced by matching systems may not be intuitively obvious to human-users and, therefore, need to be explained (Shvaiko and Euzenat, 2008). These remarks generally include the informal way of interpreting the mapping concept. In certain cases, depending on the reliability of a semantic mapping concept, other remarks may be added to capture possible limitations of the extent to which the semantic mapping concept is applicable to cross-domain arguments, and possible example remarks which further reflect the understanding behind the semantic mapping concept.

Figure 6-5 conceptually summarises the above-mentioned components of semantic mapping concepts. The diagram shows that if the argument ?x satisfies certain conditions and is defined within the “DomainX” context and the argument ?y satisfies certain conditions and is defined within the “DomainY” context, then the “semanticMappingRelation” holds true between ?x and ?y where ?x is to be interpreted in the first argument position and ?y in the second argument position to the “semanticMappingRelation”. Information carried by the relations, both formally in terms of logical definition and

informally in terms of remarks, represents the nature of semantic interoperation between cross-domain arguments.

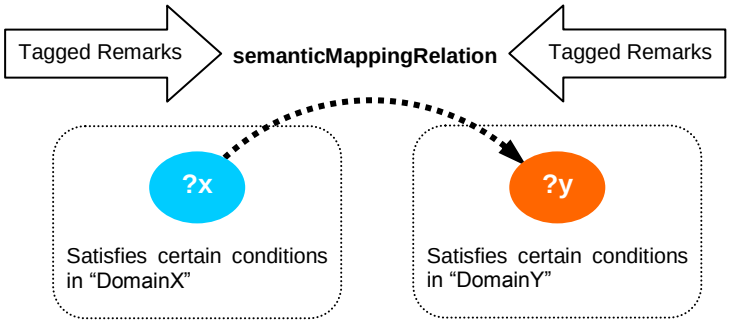


Figure 6-5 Understanding Semantic Mapping Concepts

The predefined standard contexts "DomainX" and "DomainY" present in the definition of logical conditions justify the domain context adjustment stage discussed earlier. Furthermore, it is to be noted that the arguments ?x and ?y from Figure 6-5 could be classes, instances or ontological functions but not relations. The reconciliation of relations is not under consideration in the chosen method, since the controlled specialisation approach is taken, where relation subsumptions are not permitted within domain models, in order to optimise reconciliation at the instance level.

Semantic mapping concepts embrace different levels of granularity based on foundation semantics and the user's knowledge of domain semantics. This leads to the ability to define (1) reusable semantic mapping concepts based directly on foundation semantics, (2) reusable semantic mapping concepts that are only relevant to the two domains to be reconciled and (3) reusable semantic mapping concepts based on domain knowledge that does not reside in neither the Foundation Layer nor the two domains to be reconciled. These different implications are next discussed.

6.2.2.1 Semantic Mapping Concepts Based on Foundation Semantics

A standard set of semantic mapping concepts derive from foundation semantics (see Figure 6-6). This set of mapping concepts can be reused in all reconciliation scenarios since, following the SMIF approach, all domain

models are essentially specialisations of the Foundation Layer, and therefore all share a common semantic ground.

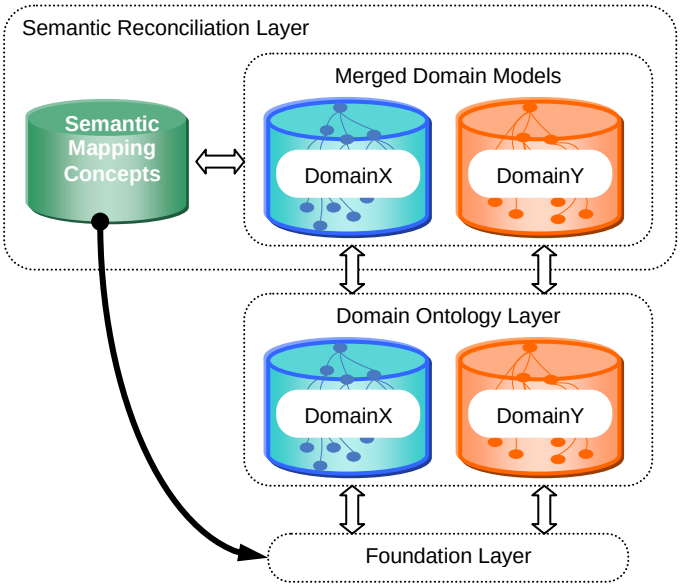


Figure 6-6 Semantic Mapping Concepts Based on Foundation Semantics

Consider Figure 6-7 which illustrates how a semantic mapping concept can be specified for the reconciliation of cross-domain sub-classes of the foundation class “Round_Hole”. The expression accompanying the diagram captures the intuition that the “classMappingRelation_018” **(A)** be inferred as true between the arguments ?x and ?y if and only if ?x is a sub-class of “Round_Hole” defined within the “DomainX” context **(B)** and ?y is another sub-class of “Round_Hole” defined within the “DomainY” context **(C)**.

```
(forall (?x ?y)
(<= (classMappingRelation_018 ?x ?y) (A)
    (and (sup ?x Round_Hole)
         (withinContext ?x DomainX) (B)
         (sup ?y Round_Hole)
         (withinContext ?y DomainY)))) (C)
```

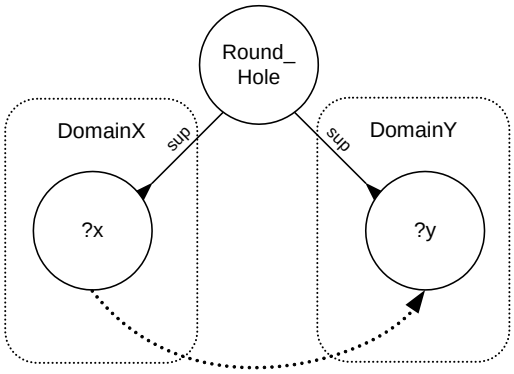


Figure 6-7 Example of a Class Semantic Mapping Concept Based on Foundation Semantics

Although the name tag of the “classMappingRelation_018” carries very little information, yet, it is formally defined (refer to the logical expression in Figure 6-7). Informal remarks can be tagged to the semantic mapping relation, based on the formal logical conditions, to enhance the meaning of “classMappingRelation_018” between ?x and ?y for human interpretation. This can be achieved by stating, for example, that:

- There exists a commonality between the class ?x in the “DomainX” context and the class ?y in the “DomainY” context as a result of both ?x and ?y being subclasses of the foundation class “Round_Hole”. Both ?x and ?y capture the notion of a feature that is of cylindrical or conical negative (removal) volume. It is necessary for instances of ?x and ?y be defined in terms of a first instance of “Circular_Closed_Profile” swept along an instance of “Linear_Path” resulting in a second instance of “Circular_Closed_Profile” of identical or different dimensions. Every instance of ?x and ?y may be specified as holding a “Linear_Profile” axis.

The above informal statement is highly relevant in terms of interoperable semantics between the possible classes ?x and ?y, since the “Round_Hole” concept possesses formal necessary conditions, captured as integrity constraints, which restrict its interpretation (see also Chapter 5 section 5.2.2.4). Besides informal remarks about the semantic commonality, there is also the issue of dealing with uncertainties in ontology matching (Shvaiko and Euzenat, 2008) and in the case of the “classMappingRelation_018”, one way to specify this is to tag a limitation remark such as:

- Without reference to the terms assigned to the concepts ?x and ?y, there could potentially be class mismatches present. This is because ?x and ?y could have been defined with a view on specific domain preferences, which vary across domains. Varying levels of abstraction of the foundation class “Round_Hole” in both domains could also result in class mismatches.

The statement identified above is also relevant to semantic reconciliation in terms of the inconclusive correspondences that could exist between ?x and ?y. This is because the logical conditions for “classMappingRelation_018” do not entail term similarities nor the identification of the number of sub-class levels of “Round_Hole” in “DomainX” and “DomainY”. Hence, it is clear that possible semantic mismatches could still prevail even though the capability is present to infer similarities between ?x and ?y.

In a very similar way to the one explained, other semantic mapping concepts can be defined based on foundation semantics. Figure 6-8 depicts a scenario where a semantic mapping relation named “instanceMappingRelation_041” **(D)** has been specified in order to partly reconcile domain-defined instances of the class “Round_Hole”. The logical expression accompanying the figure states that the “instanceMappingRelation_041” **(D)** be inferred as true between the arguments ?holex and ?holey if and only if ?holex is an instance of “Round_Hole” defined within the “DomainX” context **(E)** and ?holey is another instance of “Round_Hole” defined within the “DomainY” context **(F)** and that both instances ?holex and ?holey share the common condition of having blind circular closed profiles **(G)**.

```
(forall (?holex ?holey ?ccpx ?ccpy)
(<= (instanceMappingRelation_041 ?holex
      ?holey) (D)
    (and (inst ?holex Round_Hole)
         (withinContext ?holex DomainX) (E)
         (inst ?ccpx Circular_Closed_Profile)
         (holds_shape ?holex ?ccpx)
         (blind ?ccpx) (G)
         (inst ?holey Round_Hole)
         (withinContext ?holey DomainY) (F)
         (inst ?ccpy Circular_Closed_Profile)
         (holds_shape ?holey ?ccpy)
         (blind ?ccpy)))) (G)
```

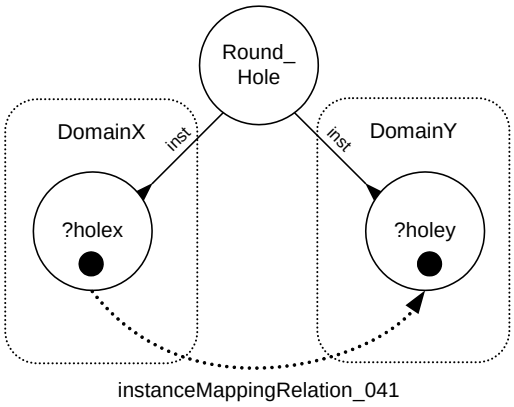


Figure 6-8 Example of an Instance Semantic Mapping Concept Based on Foundation Semantics

The informal remarks which support the definition of the semantic mapping concept to partly reconcile round holes in “DomainX” and “DomainY” state that:

- There exists a commonality between the instances ?holex and ?holey as a result of both being asserted instances of the foundation class “Round_Hole” declared in “DomainX” and “DomainY” respectively. ?holex and ?holey both share in common the property of having blind hole bottom conditions.

In this case the logical conditions that define “instanceMappingRelation_041” are very constrained and for this reason, no potential limitation could be envisaged over the semantic mapping concept. In other words, if the “instanceMappingRelation_041” holds true for two instance arguments ?holex and ?holey, then there is a total certainty that the semantic mapping concept applies under all circumstances (refer to Appendix E.1 for more information on other similar types of semantic mapping concepts that derive from foundation semantics).

6.2.2.2 Semantic Mapping Concepts Based on Known Cross-Domain Correspondences

The definition of semantic mapping concepts can also be based on the user’s knowledge of the concepts and conditions present in two domain models to be reconciled (see Figure 6-9). It becomes possible to specify domain-derived semantic mapping concepts depending on the user’s knowledge of the commonalities and differences between the two domain models. This knowledge can, for example, be gathered through historical cross-domain information correspondences which are at the disposal of the knowledge engineer. This understanding falls in line with the discovery of missing background knowledge to improve matchability acknowledged by Shvaiko and Euzenat (2008) in their analysis of ten challenges for ontology matching.

Unlike the semantic mapping concepts that are based on foundation semantics and are fully reusable in all reconciliation scenarios, semantic mapping concepts based on the semantics of reconcilable domains are much more specific and are only generally reusable for two domain models, for which the semantic mapping concepts have been designed to reconcile.

Consider the example portrayed in Figure 6-10 where the foundation class “Activity” has one specialisation called “Drilling_Process” defined in the “DomainX” context and another specialisation called “Drilling_Operation” defined in the “DomainY” context.

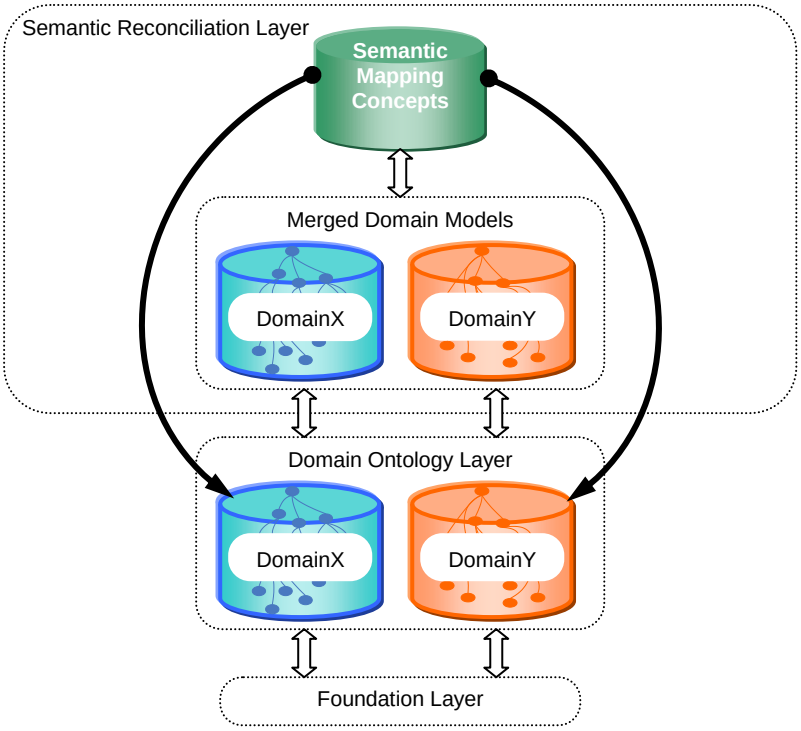


Figure 6-9 Semantic Mapping Concepts Based on Known Cross-Domain Correspondences

Purely based on the “sup” subsumption relation of the domain-defined classes to the class “Activity”, it is possible to infer that both “Drilling_Process” and “Drilling_Operation” originate from the same parent class. However, if the knowledge engineer already understands the implications of “Drilling_Process” and “Drilling_Operation”, the latter could specify a domain-derived semantic mapping concept that directly holds between these two classes. In this case, assuming that the knowledge engineer understands that the two classes are in fact the same, a semantic mapping concept called “classDomainMappingRelation_001” **(H)** can be used to infer that “Drilling_Process” and “Drilling_Operation” are conceptually similar types of reusable process behaviours as depicted in the expression in Figure 6-10.

```
(<= (classDomainMappingRelation_001
      Drilling_Process Drilling_Operation)
    (and (Property Drilling_Process)
         (Property Drilling_Operation)))
```

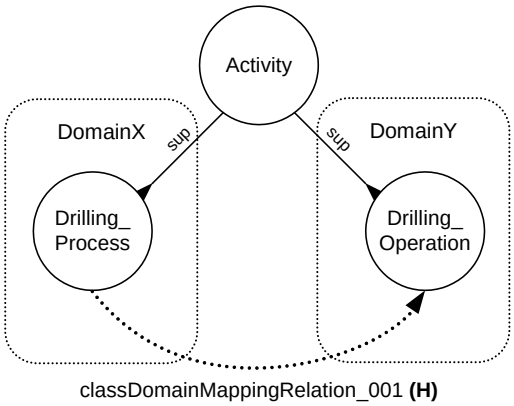


Figure 6-10 Example of a Class Semantic Mapping Concept Based on Known Cross-Domain Correspondences

User-defined informal remarks listed below can also be added for enhancing human interpretation:

- The “Drilling_Process” class in “DomainX” is a conceptually similar class to the “Drilling_Operation” class in “DomainY”.
- There is a term mismatch between the class “Drilling_Process” and “Drilling_Operation”.

The process of establishing semantic mapping concepts based on known cross-domain correspondences is regarded as being an important method currently used in ontology mapping research. In Description Logics (DL), pre-defined semantic mapping relations such as “owl:sameClassAs” (Lin and Harding, 2007), other related DL comparison relationships (Lazenberger *et al*, 2008; Rabe and Gocev, 2008) and the exploration of “semantic bridges” such as the “ConceptBridge” (Maedche and Staab, 2002) provide similar capabilities to the “classDomainMappingRelation_001” **(H)** discussed earlier.

The fundamental difference between the semantic mapping concepts based on known cross-domain correspondences explored in this work, for example “classDomainMappingRelation_001”, and other related concepts like “owl:sameClassAs” lies in the degree of formality and flexibility in the definition of the former. Heavyweight Common Logic-based rules are used to reinforce the semantics of semantic mapping concepts based on known

cross-domain correspondences. Furthermore, the tagging of informal remarks, which informally identify the criteria for matching and the limitations to the matching, improves the interpretability of semantic mapping concepts.

6.2.2.3 Semantic Mapping Concepts Based on External Domains

There is a third mode in which semantic mapping concepts could be defined following the SMIF approach. In Figure 6-11, the two domains to be reconciled are “DomainX” and “DomainY”. The knowledge engineer is able to specify other types of semantic mapping concepts based on some external domain model if the knowledge contained within the external model can potentially be used during the reconciliation of “DomainX” and “DomainY”. This external domain model also needs to have been developed from the Foundation Layer.

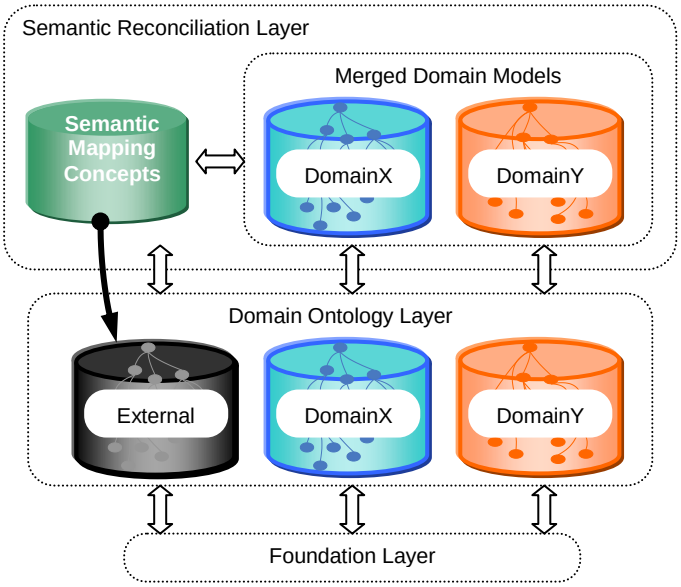


Figure 6-11 Semantic Mapping Concepts Based on External Domains

The diagram in Figure 6-12 exemplifies the understanding behind the specification of semantic mapping concepts based on external domain knowledge. In this example, the external domain model is that of the ISO Tolerance Band and machining processes associated with ISO IT Tolerance Grade (ISO 286-2, 1988). In this external domain model, the knowledge that originates from ISO Tolerance Band and machining processes has been

formalised in such a way that it is possible to infer suitable machining methods based on the dimensional parameters of “Round_Hole” instances.

The expression in Figure 6-12 captures the ISO Tolerance Band domain condition that if a “Round_Hole” instance **(I)** has an entry hole diameter that is between 6 mm (exclusive) and 10mm (inclusive) **(J)** accompanied with an upper diameter tolerance value which is between 0.06 mm (inclusive) and 0.36 mm (inclusive) **(K)** and a lower diameter tolerance value which is between -0.36 mm (inclusive) and -0.06 mm (inclusive) **(L)**, then it is possible to infer that the unary relation “toleranceBandRelation_04” **(M)** holds for that specific instance of “Round_Hole”. In this case, the informal remarks that accompany the declaration of “toleranceBandRelation_04” state that:

- Based on the entry diameter and entry diameter size tolerance of the queried “Round_Hole” instance, it can be inferred that this hole feature can be produced using a Reaming machining process. This criteria is only satisfied under the ISO Tolerance Band domain model.

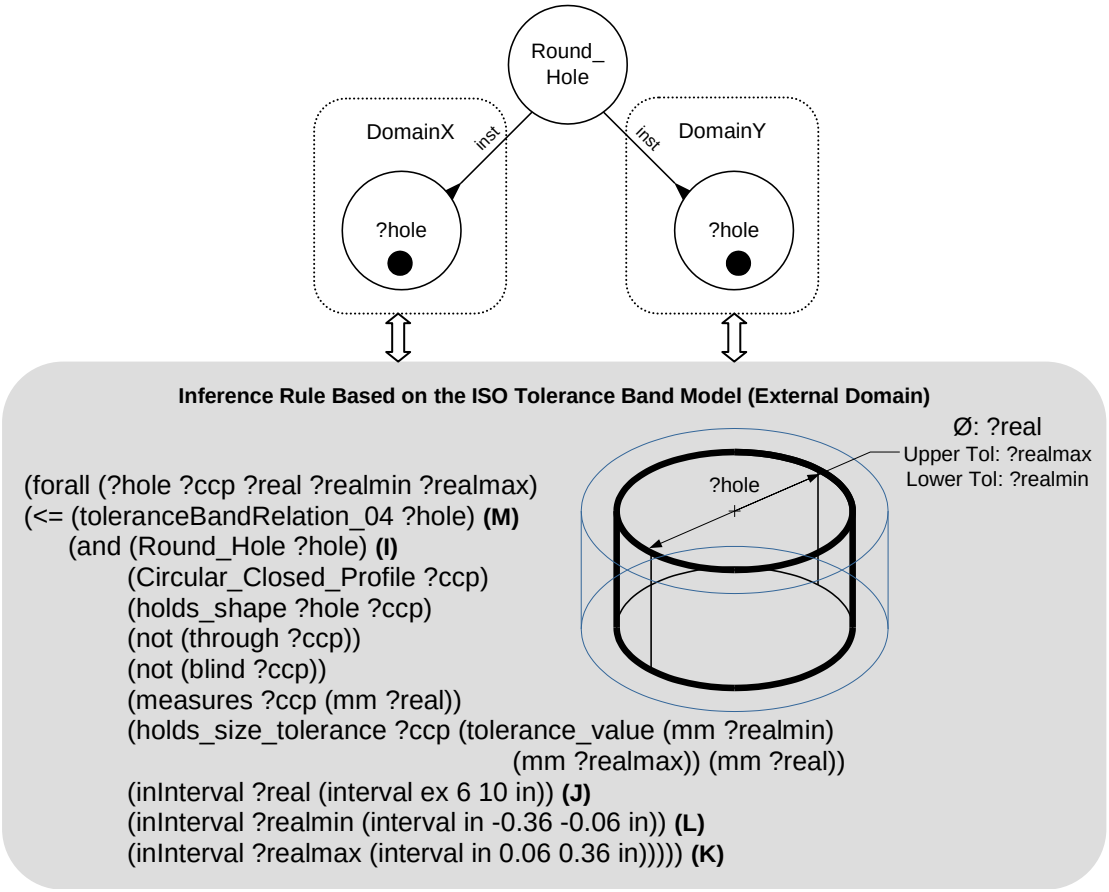


Figure 6-12 Example of an Instance Semantic Mapping Concept Based on an External Domain

The knowledge captured in the expression in Figure 6-12 shows an example of how a semantic mapping concept could be specified using an external domain model as an articulation model for reconciling two other domain models. The unary relation “toleranceBandRelation_04” **(M)** can act as a semantic mapping concept when used to reconcile “Round_Hole” instances in “DomainX” and “DomainY”, where these “Round_Hole” instances happen to share the commonality of being able to be machined using reaming processes under the ISO Tolerance Band conditions set. In other words, SMIF provides the potential for reusing the knowledge contained in ISO standard-based domain models towards the reconciliation of pairs of other domain models.

6.2.3 Summary of Semantic Reconciliation Layer

The Semantic Reconciliation Layer is based on the interactions between three key stages of the ontology mapping process between two domain models to be reconciled. Ontology mapping process concepts adopted in the third layer of the research framework involve a domain context adjustment process followed by a simple ontology merging action. The next stage is that of semantic alignment, where a number of pre-defined semantic mapping concepts aligns the arguments present across domain models.

Semantic mapping concepts may be developed from three distinct angles. A reusable set of semantic mapping concepts can be defined from foundation semantics. Also, depending on the experience of the knowledge engineer, it is possible to define semantic mapping concepts based on known cross-domain correspondences that only apply to the specific pair of domain models to be reconciled. The third method of specifying semantic mapping concepts is concerned with the use of knowledge, coming from other domain models external to the pair of domain models to be reconciled, which serves as articulation knowledge. The case study in Chapter 8 explores all three means of defining semantic mapping concepts (also consult Appendix E for a sample of explored semantic mapping concepts).

Semantic mapping concepts are not limited to one-to-one mappings. They can involve, in addition to (1) one-to-one mappings, (2) many-to-one, (3) one-to-many and (4) many-to-many mappings, which are governed through the logical conditions that define semantic mapping concepts. Although semantic mapping concepts help identify the correspondences that may exist between two distinct domain representations, it is nevertheless appreciated that semantic mismatches could still occur. This has helped identify a suitable way, by using tagged remarks, to flag the uncertainties or possible mismatches that might exist even after a semantic mapping relation has been established. In this way, not only is the user able to understand what is sharable between two cross-domain arguments, but the latter is also made aware of the extent to which it is not possible to infer about their resemblance.

6.3 Interoperability Evaluation Layer

One of the most active areas of research in ontology alignment is the automatic and semi-automatic mapping discovery (Noy and Stuckenschmidt, 2005). The Interoperability Evaluation Layer which is at the last level of the SMIF uses semi-automatic mechanisms for enabling mapping discovery. This layer builds on top of the Semantic Reconciliation Layer and, therefore, uses the capabilities achieved in the third level to help evaluate the commonalities, differences and uncertainties (i.e. correspondences) across domain models.

Figure 6-13 illustrates how the last stage of the ontology mapping process in the Semantic Reconciliation Layer contributes to the ability to formulate interoperable knowledge queries in the Interoperability Evaluation Layer. The alignment of pairs of domain models provides a basis for the retrieval of mappings across models (interoperability evaluation process). Moreover, since all the semantics in the merged and aligned representations are logically defined, it is possible to verify the conformance of retrieved evaluated results (verification process).

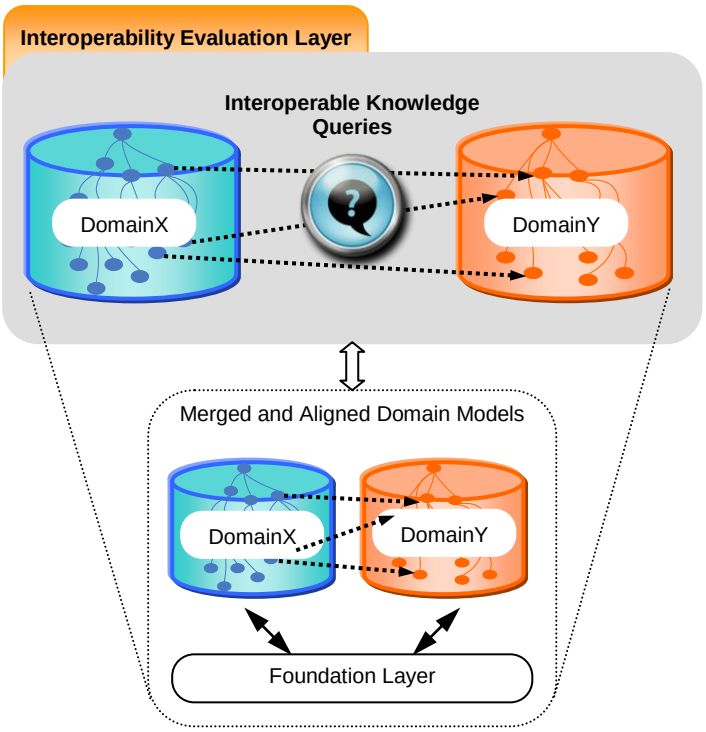


Figure 6-13 Interoperable Knowledge Queries in the Interoperability Evaluation Layer

6.3.1 Interoperable Knowledge Queries

Interoperable knowledge queries are Common Logic-based queries that allow (1) the retrieval of cross-domain arguments over known semantic mapping concepts and (2) the retrieval of semantic mapping concepts over known cross-domain arguments. These queries fall in the category of structured query processing of alignments supported by ontologies (Noy and Stuckenschmidt, 2005) and the explanation of matching results in ontology matching (Shvaiko and Euzenat, 2008). The integrity of results obtained from an interoperable knowledge query can be verified via logical proof. This proof is traced back from the source logic of the semantic mapping concepts, the cross-domain arguments in question and a breakdown of the conditions, as to why certain cross-domain arguments satisfy certain semantic mapping concepts, in order to provide a valid logical justification. The next sub-sections of this chapter explain, in an exemplified fashion, the relevant mechanisms involved in evaluating and verifying interoperable knowledge queries.

6.3.1.1 Querying Cross-Domain Arguments over Known Semantic Mapping Relations

One possible way of formulating interoperable knowledge queries in the Interoperability Evaluation Layer is to create a general query over a known semantic mapping relation and deduce whether or not there are cross-domain arguments that have become bound to the semantic mapping relation during the semantic alignment process previously discussed. Consider the example illustrated in Figure 6-14 which is based on Figure 6-7.

In the diagram, the class “Locating_Hole”, defined in the “DomainX” context is a sub-class of “Round_Hole” and the class “Gate_Hole”, defined in “DomainY” is another sub-class of the foundation class “Round_Hole”. During the semantic alignment process, based on the logical conditions that define the relation “classMappingRelation_018” **(A)**, the classes “Locating_Hole” and “Gate_Hole” are inferred as being valid ?x and ?y arguments to the “classMappingRelation_018” respectively (refer to **(A)**, **(B)** and **(C)** in the expression in Figure 6-14). Recall that this semantic mapping concept helps to establish a correspondence between cross-domain sub-classes of “Round_Hole” and also increases awareness about possible class mismatches between these cross-domain sub-classes.

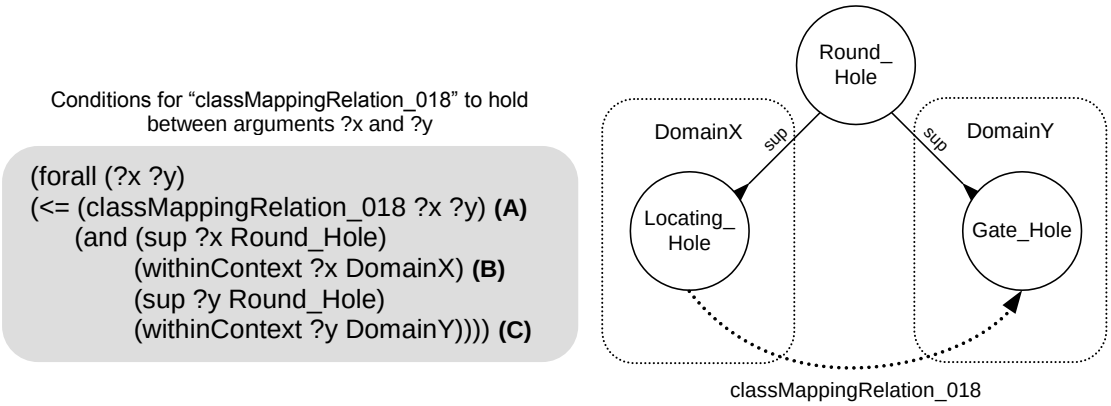


Figure 6-14 Example of a Scenario to Be Queried which Returns a One-to-One Mapping Result

Assuming that the user is unaware of the semantic mapping relation between “Locating_Hole” and “Gate_Hole”, the person could formulate a query by selecting “classMappingRelation_018” to find out whether or not there are

arguments bound by the relation. This query would be in the Common Logic statement of: *(classMappingRelation_018 ?x ?y)*

If the query transaction returns results, then the results would be in the form of a list of all the possible combinations under “classMappingRelation_018”. In the example depicted in Figure 6-14, the argument ?x would be “Locating_Hole” while the argument ?y “Gate_Hole”, thereby returning a one-to-one mapping. In the event that there were two or more specialisations of “Round_Hole” defined in the “DomainX” context and one specialisation of the same foundation class declared in the “DomainY” context as shown in Figure 6-15, then there would be a many-to-one mapping under the same logical conditions that define “classMappingRelation_018”. Many-to-many mappings would occur in the presence of a plurality of sub-classes of “Round_Hole” in both “DomainX” and “DomainY”.

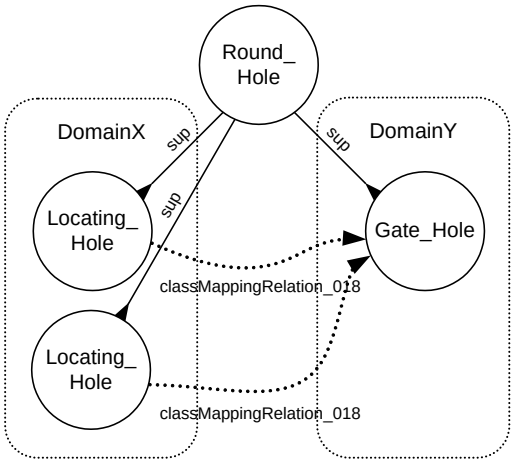


Figure 6-15 Example of Many-to-One Mapping Results

The type of querying method mentioned in this section is highly useful when the user readily understands the implied semantics of the queried semantic mapping concept and wants to discern what cross-domain arguments are bound by the relation. However, not in all circumstances is the user expected to be an expert in interpreting semantic mapping concepts. For this reason, this querying method is not always preferred from a user perspective. The next sub-section explains an alternative direction to optimise interoperable knowledge querying procedures. Furthermore, a potential problem occurs when there is a large number of semantic mapping concepts that needs to be

managed. Section 6.3.2 suggests a method to facilitate the management of semantic mapping concepts for reusability.

6.3.1.2 Querying Semantic Mapping Relations over Known Cross-Domain Arguments

An alternative way of formulating queries, in the Interoperability Evaluation Layer, is to discover in a single querying transaction all the semantic mapping relations that hold between two chosen cross-domain arguments. Selecting cross-domain arguments can be performed by browsing through the merged domain models. It is to be noted that the selection of cross-domain arguments is dependent on the user's objectives and intentions during the querying procedure. Consider the example portrayed in Figure 6-16 which is based on Figure 6-8.

In the illustration, the instance "Hole_X", defined in the "DomainX" context is an instance of "Round_Hole" and the instance "Hole_Y", defined in "DomainY" is another instance of the foundation class "Round_Hole". Assuming that "Hole_X" and "Hole_Y" both satisfy the given logical conditions for holding blind hole bottom parameters (according to **(E)**, **(F)** and **(G)**), the relation "instanceMappingRelation_041" **(D)** infers the instances "Hole_X" and "Hole_Y" as being valid ?holex and ?holey arguments to the "instanceMappingRelation_041" respectively.

Conditions for "instanceMappingRelation_041" to hold between arguments ?holex and ?holey

```
(forall (?holex ?holey ?ccpx ?ccpy)
(<= (instanceMappingRelation_041 ?holex
      ?holey) (D)
    (and (inst ?holex Round_Hole)
         (withinContext ?holex DomainX) (E)
         (inst ?ccpx Circular_Closed_Profile)
         (holds_shape ?holex ?ccpx)
         (blind ?ccpx) (G)
         (inst ?holey Round_Hole)
         (withinContext ?holey DomainY) (F)
         (inst ?ccpy Circular_Closed_Profile)
         (holds_shape ?holey ?ccpy)
         (blind ?ccpy)))) (G)
```

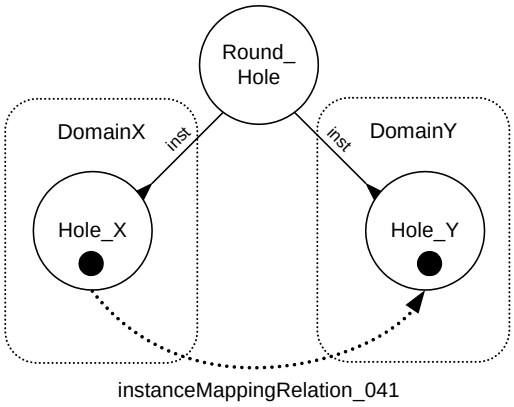


Figure 6-16 Example of a Scenario to be Queried between Known Cross-Domain Instances

Knowing that “Hole_X” and “Hole_Y” exist in the merged domain models, the user is able to write a query with the intention of retrieving all possible semantic mapping relations based on foundation semantics that bind these two instances together, where “Hole_X” is in the first argument position and “Hole_Y” in the second argument position. The person would do so by stating a query in the form:

(and (BinaryRelation ?rel) (withinContext ?rel foundationMapping) (holdsArg ?rel 1 DomainX.Hole_X) (holdsArg ?rel 2 DomainY.Hole_Y))

When the query is run, the user is able to gather a list of all the semantic mapping relations based on foundation semantics, that apply to the instances “Hole_X” and “Hole_Y”. The query should return the binary relation “instanceMappingRelation_041” as a relation that binds “Hole_X” and “Hole_Y”. The user is then able to browse “instanceMappingRelation_041” in order to view the informal remarks that are tagged to the relation for further interpretation of the correspondence. It is to be noted that the way to formulate queries is dependent on the expertise of the user in the use of KFL. It is also possible to provide user interfaces for guiding the user through querying procedures as explained in sections 6.3.2 and Chapter 7 section 7.3.4. This helps to retrieve accurate queries that do not demand a solid knowledge of KFL on behalf of the user.

6.3.1.3 Verification of Reconciliation Correspondences

It has been acknowledged that the verification of alignment results (Lazenberger *et al*, 2008) forms an important facet of ontology alignment for knowledge sharing and reuse. In the research framework, by committing to the Foundation Layer, multiple KBs (in this case domain models) are enforced a common set of rules and constraints, which is particularly useful when attempting to verify the interactions of multiple KBs (Cochrane, 2006). Hence, following the framework approach, the verification of reconciliation correspondences between cross-domain arguments is the procedure by which the results obtained from a query action are checked for conformance

to (1) the logical conditions set in the query and (2) any inferred logical conditions based on semantic mapping concepts.

Based on the scenario in Figure 6-16 and Figure 6-17, the verification process entails the action of proving the reason why “instanceMappingRelation_041” corresponds to the queried variable ?rel. The logical proof in this case reflects the fact that in the query:

- ?rel is a binary relation.
- ?rel has been defined in the “foundationMapping” context and is, therefore, a semantic mapping relation based on foundation semantics.
- ?rel holds the argument “Hole_X” in the first argument position.
- “Hole_X” is an argument defined in the “DomainX” context.
- ?rel holds the argument “Hole_Y” in the second argument position.
- “Hole_Y” is an argument defined in the “DomainY” context.

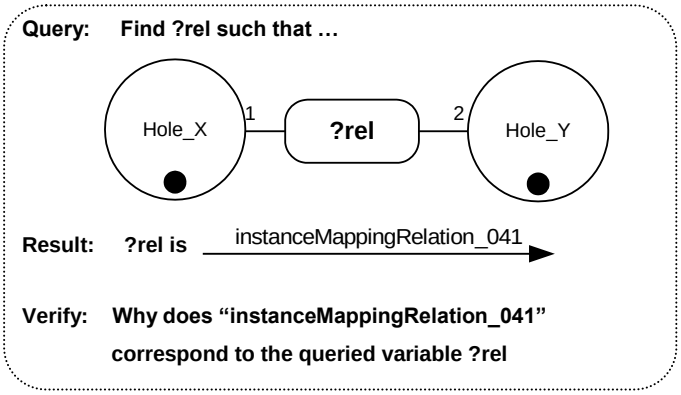


Figure 6-17 Example of a Verification Procedure

Since “instanceMappingRelation_041” satisfies all the above-mentioned conditions and “Hole_X” and “Hole_Y” also satisfy the criteria for the relation to bind them together (through inferred logical conditions), this implies that “instanceMappingRelation_041” is in fact ?rel. Hence, the reconciliation correspondence is a verified semantic mapping relation that holds for “Hole_X” and “Hole_Y”, since its occurrence can be proved.

Verification processes are particularly significant when different parties involved in multi-domain model reconciliation wish to become aware of the logical conditions pertaining to the reasons as to why certain semantic mapping concepts exist between cross-domain arguments. Therefore, automated verification through the exploitation of heavyweight logic is key to ensuring the integrity of sharable knowledge between systems.

6.3.2 Assisting Knowledge Querying Procedures

Section 6.2.2 has identified three ways in which semantic mapping concepts can be defined based on (1) foundation semantics, (2) known cross-domain correspondences and (3) external domains. Another viewpoint from which semantic mapping concepts are being developed under this work is to consider different levels of domain models namely the class, instance and function levels. As a result of these various possible ways of categorising semantic mapping concepts, this implies that querying procedures naturally follow a similar breakdown. As the extent of foundation and domain semantics grows, so does the rate of development of semantic mapping concepts and their associated queries. This clearly demonstrates that there is a need for the effective management of queries, which would assist the user during knowledge querying procedures.

In ontology reconciliation research, it has been acknowledged that semantic mapping management systems is needed to support users and applications in creating, reusing, managing and applying such semantic mappings in order to handle these multiple and complex semantic mappings (Thomas *et al*, 2008). Furthermore, the development of suitable infrastructure and support for alignment management still remains a challenge for ontology matching (Shvaiko and Euzenat, 2008). This challenge is clear in this work as a result of an extensive range of semantic mapping concepts and their associated queries.

In order to assist knowledge querying procedures, a matrix approach has been devised. The main purpose of the matrix is to support the ability to

configure the different modes in which semantic mapping concept queries occur according to the different levels involved in domain models. Figure 6-18 depicts the matrix configuration adopted.

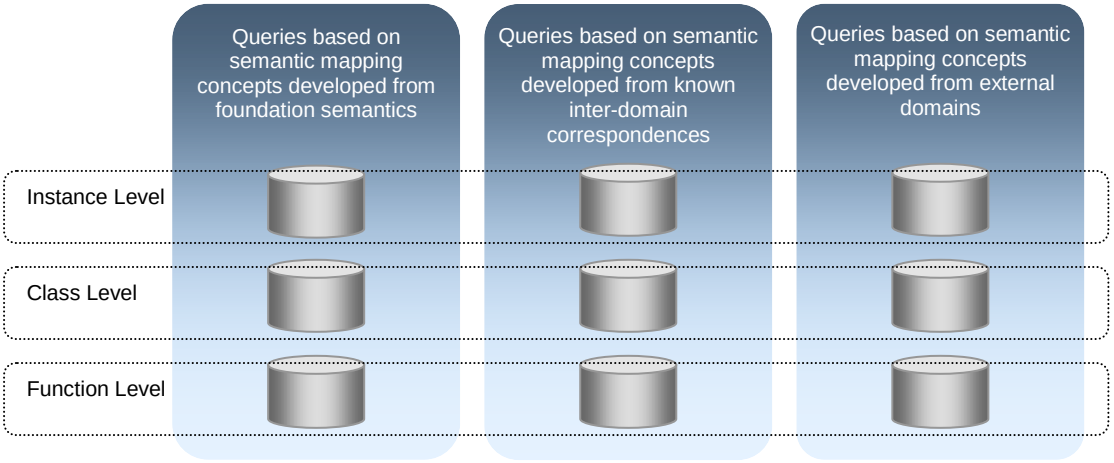


Figure 6-18 Matrix Configuration to Assist Knowledge Querying Procedures

In the figure, the matrix follows an intuitive categorisation approach, for example, queries based on semantic mapping concepts, that derive from foundation semantics, can be carried out at the instance level, class level and function level of merged and aligned domain models. This also applies to the other two modes in which semantic mapping concept queries occur. Each cell of the matrix correspond to the collection of designated queries at the relevant instance, class or function level. The matrix approach is simple in essence, yet it can facilitate user-system interactions during knowledge querying procedures.

6.3.3 Summary of Interoperability Evaluation Layer

The capability for interoperability evaluation in the fourth layer of the research framework is based on the formulation of interoperable knowledge queries expressed in Common Logic, while the verification process involves the logical reasons for which certain results are obtained when queries are run. There are two main ways in which queries can be executed namely (1) by querying for cross-domain arguments over known semantic mapping relations and (2) by querying for semantic mapping concepts based on known cross-domain arguments.

Unlike the first querying method, the second provides an optimised way of retrieving all the mapping correspondences between known pairs of arguments present across domain models. Moreover, this querying mode does not necessarily require the user to be an expert in understanding semantic mapping concepts. Nevertheless, since both methods of querying are useful in the Interoperability Evaluation Layer, both remain under consideration. Furthermore, a matrix-based configuration has been proposed in order to support the management and retrieval of all querying possibilities within a reconcilable system.

6.4 Summary

This chapter has documented the nature and implications of the second two layers of the Semantic Manufacturing Interoperability Framework (SMIF), which helps meet the third objective of this work (see Chapter 1 section 1.3.1). The Semantic Reconciliation Layer exploits a combined improved approach based on known methods of ontology reconciliation. This combined approach imparts a unique facet to the third layer of the framework, by enabling the meaningful capture of the semantics of mapping concepts using the Knowledge Framework Language (KFL). IDEF5 schematics have been employed in this chapter to visually communicate various scenarios that take place during semantic reconciliation.

The Interoperability Evaluation Layer closely interacts with the Semantic Reconciliation Layer to provide the capability to evaluate and verify the correspondences that hold between the entities from two domain models. Querying techniques act as mapping discovery methods to understand the consequence of ontology mapping performed in the third layer of the framework. The case study in Chapter 8 experimentally tests the concepts explored in the Semantic Reconciliation and Interoperability Evaluation layers.

7 Experimental System Development

7.1 Introduction

The development of the experimental system to explore and experiment with the different constituent layers of the Semantic Manufacturing Interoperability Framework (SMIF) and their interactions are documented in this chapter. Section 7.2 concentrates on providing an overview into the design of the experimental system for the framework. The various tools exploited for this purpose are first presented. Section 7.3 then provides further details on the experimental system, by targeting the implementation side of the four layers of the SMIF, while emphasising the relevant development methodologies employed.

A number of key facets has been identified as part of the experimental system development process. These are namely:

- The formalisation of the heavyweight manufacturing ontological foundation of the Foundation Layer in an appropriate ontological environment.
- The exploitation of the Foundation Layer for the purpose of developing semantically-sound domain models in the Domain Ontology Layer (see Chapter 8 for more details).
- The formalisation of a set of semantic mapping concepts, based on the three different modes in which these occur (refer to Chapter 6 if required) and explore the ontology mapping process concepts pertinent to the Semantic Reconciliation Layer.
- The exploration of querying methods via the use of a suitable interface for assisting knowledge querying procedures and through appropriate query tools.

7.2 Design of the Experimental System

There exist two critical aspects involved in the design of the experimental system for testing the research framework, namely (1) the selection of relevant software applications and (2) the knowledge representation formalism. A list of the software tools and the knowledge representation formalism applied to meet the needs of the experimental system are identified next. These resources have been selected based on their availability for research and other set preferences for this work.

- Integrated Ontology Development Environment (IODE) V2.1.1 developed by Ontology Works Inc. (Ontology Works Inc., 2009). IODE is an ontological environment that is capable of handling heavyweight Common Logic-based ontologies and KBs. This ontology development tool constitutes the primary environment for deploying the experimental system.

- Knowledge Framework Language (KFL). KFL is a Common Logic-based ontological formalism, developed by Ontology Works Inc., that provides the syntax and first order semantics required for developing heavyweight ontological models. The ability to encode ontological content in KFL derives from Ontology Work's Upper Level Ontology (ULO).

- Unlike other ontological environments, like Protégé (Protégé Website, 2009) for instance, IODE makes use of ontology files that are "written" in KFL format outside the ontological environment before these files can be loaded and saved into IODE. For the purpose of "writing" these files, the software tool Notepad++ has been sought (Notepad++ Website, 2009). Notepad++ is a free source code editor that is particularly useful for manipulating programming in various forms. SCL files (instance files) can also be written using this application.

- Ontology development processes have been aided through the use of the IDEF5 schematic language (Knowledge Based Systems Inc., 1994) for

diagrammatically representing ontological content such as classes and their taxonomies, relations and ontological functions, before these ontologies are coded in the KFL format. As a result of the unavailability of a dedicated tool for constructing these schematics, a template (see Appendix A) has been purposely developed for providing the required IDEF5 schematic constructs using Microsoft Office Visio 2003 (Microsoft Office Visio Homepage, 2009). This application has also been used for the manipulation of some graphics during the development of the interface for assisting knowledge querying procedures.

- A suite of development tools such as Adobe Flash 8 (Adobe Website, 2009) and scripting languages like ActionScript 2.0 have been utilised during the course of the development of this interface. These development methods are further identified in section 7.3.4.1 of this chapter.

7.3 Implementation of the Experimental System

An overview of the total implementation of the experimental system is next revealed. Figure 7-1 illustrates how all the levels of the SMIF are implemented in IODE and where relevant tools come into play. KFL files are present at the first three levels of the framework for encoding ontological content needed in the Foundation and Domain Ontology layers and to formalise semantic mapping concepts present in the Semantic Reconciliation Layer. SCL files are required in the Domain Ontology Layer to populate instances. Ontology files in KFL are loaded in separate Object Management Systems (OMS) in IODE as shown in Figure 7-1.

An OMS in IODE corresponds to a system that holds an ontology written in KFL with a linked KB for populating facts based on the ontology (if needed). In IODE, the Foundation Layer and domain models in the Domain Ontology Layer are developed in separate OMSs. In the event that a pair of domain models needs to be reconciled, each are merged in a single OMS (see OMS4 in Figure 7-1) and KFL files that hold the necessary semantic mapping

concepts are then loaded and saved in the merged OMS in order to complete the ontology mapping process.

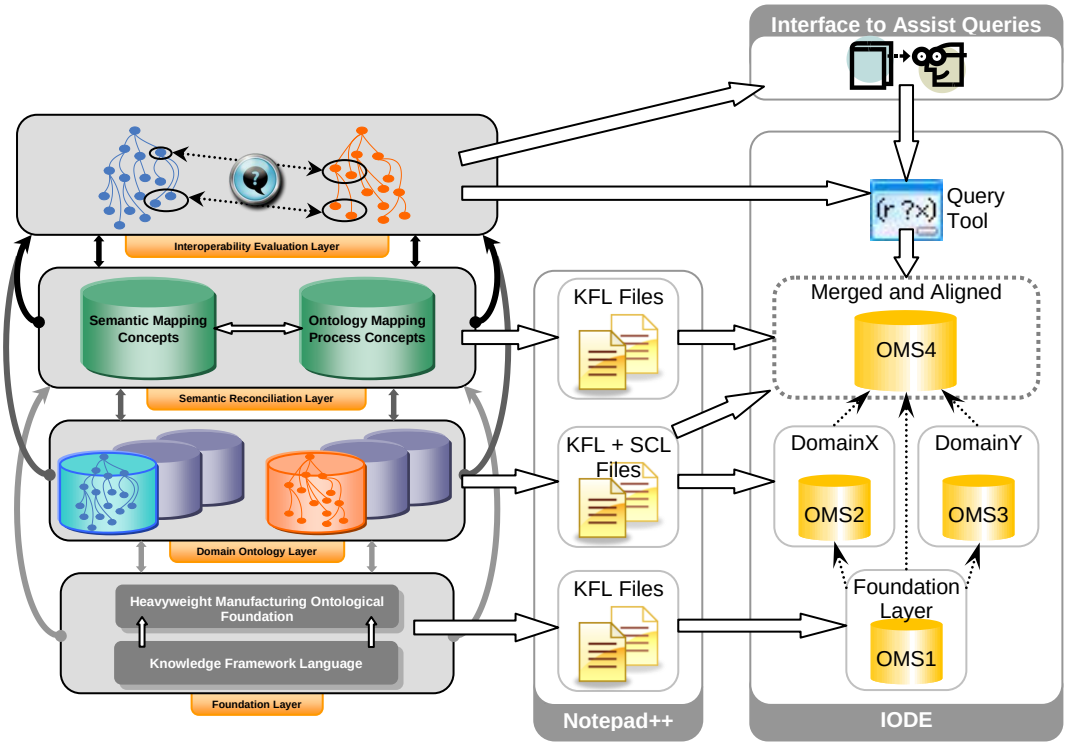


Figure 7-1 Overview of the Implementation of the Experimental System

To evaluate and verify cross-domain correspondences in the Interoperability Evaluation Layer, two main tools are employed. The interface, developed to assist querying procedures, is first used to retrieve the appropriate user-selected query. This query is then pasted in the query tool (an integral module of IODE) and run to process results. Results can then be analysed in IODE itself or saved for other external transactions.

7.3.1 Implementation of the Foundation Layer

The implementation of the Foundation Layer is at the base of the experimental system development process. All the concepts discussed in section 5.2 of Chapter 5 have been implemented in IODE. It is also to be noted that relevant concepts from the CPM and ISO 10303 AP224 have been formalised in KFL, following a careful study of their lightweight structures, natural language statements and informally-expressed axioms. Due to the different ontological components of the heavyweight manufacturing

ontological foundation, such as (1) process semantics based on PSL, (2) entity information semantics and (3) the participation semantics between entity and process concepts, KFL ontology files are developed for each component (1), (2) and (3). These files are then loaded together in a single OMS to hold the Foundation Layer of SMIF. Appendix C supports a full set of implemented semantic structures for the Foundation Layer.

The development of the heavyweight manufacturing ontological foundation is based on the knowledge engineering methodology prescribed by Noy and McGuinness (2001). Following this methodology, one major competency question has been identified for the implementation of the Foundation Layer:

- Can the Knowledge Framework Language (KFL) and IODE be used to formally capture and represent the heavyweight semantics required for a fully functioning Foundation Layer?

7.3.1.1 Implementation of PSL Core and PSL Outer-Core

The implementation of process semantics coming from PSL Core and Outer-Core theories forms part of one of the components of the Foundation Layer. To load and save PSL in a fresh OMS **(A)** (see Figure 7-2), the KFL file containing the PSL process semantics **(B)** is browsed and firstly parsed **(C)**. If no parsing errors are present, the loading process is initiated as shown in label **(D)**. If during the loading process no loading errors are detected, the loaded file is accepted and can be saved **(E)** to the OMS.

It is important to note that parsing errors occur as a consequence of missing parentheses in written axioms, wrongly specified KFL commands and the like. On the other hand, the loading action detects errors in the event that logical integrity conflicts are present within the KFL file during the loading process (i.e. the ontology as a logical theory does not prove to be consistent in its entirety). Note that the parsing, loading and saving processes in the OMS of the Foundation Layer also applies to KFL files that contain entity information semantics and participation semantics between entities and processes.

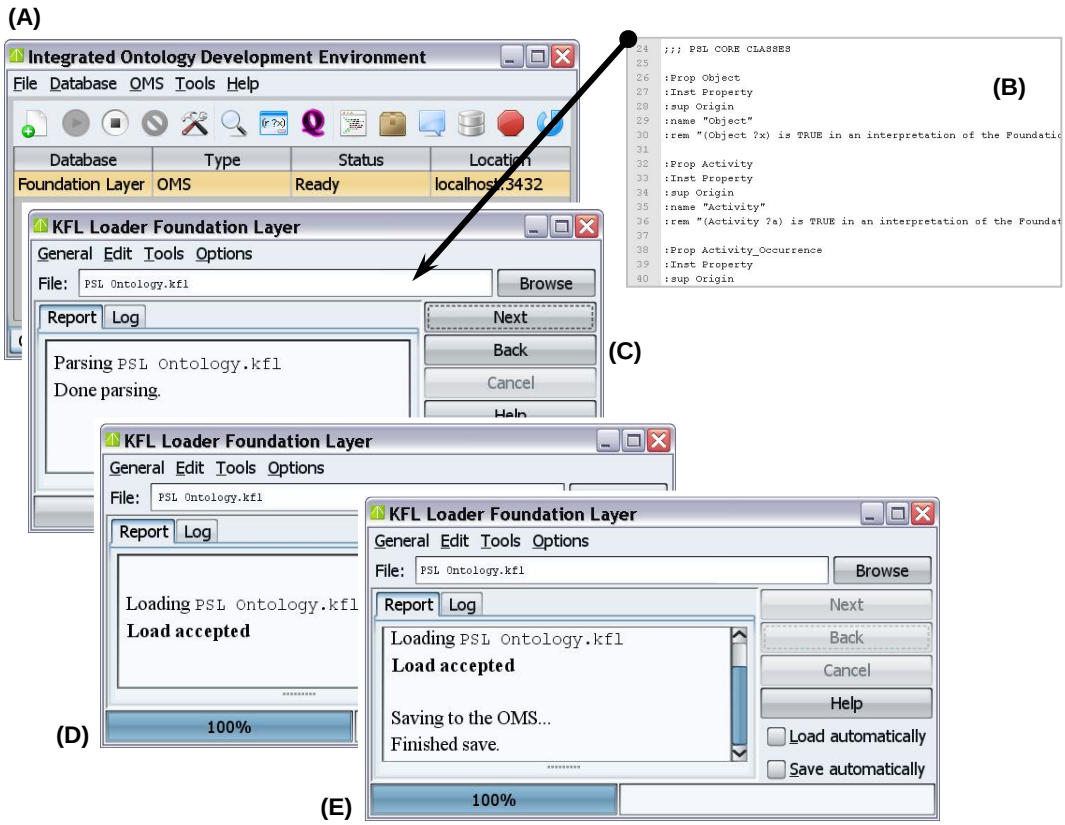


Figure 7-2 Parsing, Loading and Saving Process Semantics in the Foundation Layer OMS

7.3.1.2 Implementation Issues with PSL Process Semantics

During the implementation of PSL, from its Common Logic Interchange Format (CLIF) form to its KFL form in IODE, a few issues have been faced and resolved. One of the issues lies in the need to remove "forall" statements **(F)**, carefully disambiguating variables present in axioms **(G)** and appending the axiom with the necessary type of integrity constraint statement **(H)** as shown in Expression 7-1.

PSL Core axiom 13. An activity occurrence is associated with a unique activity.

CLIF Form

```
(forall (?occ ?a1 ?a2) (F)
(if (and (occurrence_of ?occ ?a1)
         (occurrence_of ?occ ?a2))
   (= ?a1 ?a2)))
```

KFL Form

```
(=> (and (Activity ?a1) ——— (G)
         (Activity ?a2) ——/  \
         (Activity_Occurrence ?occ)
         (occurrence_of ?occ ?a1)
         (occurrence_of ?occ ?a2))
   (= ?a1 ?a2)))
```

:IC hard "An activity occurrence is associated with a unique activity." **(H)**

Expression 7-1 Implementing a CLIF-Written PSL Axiom in KFL

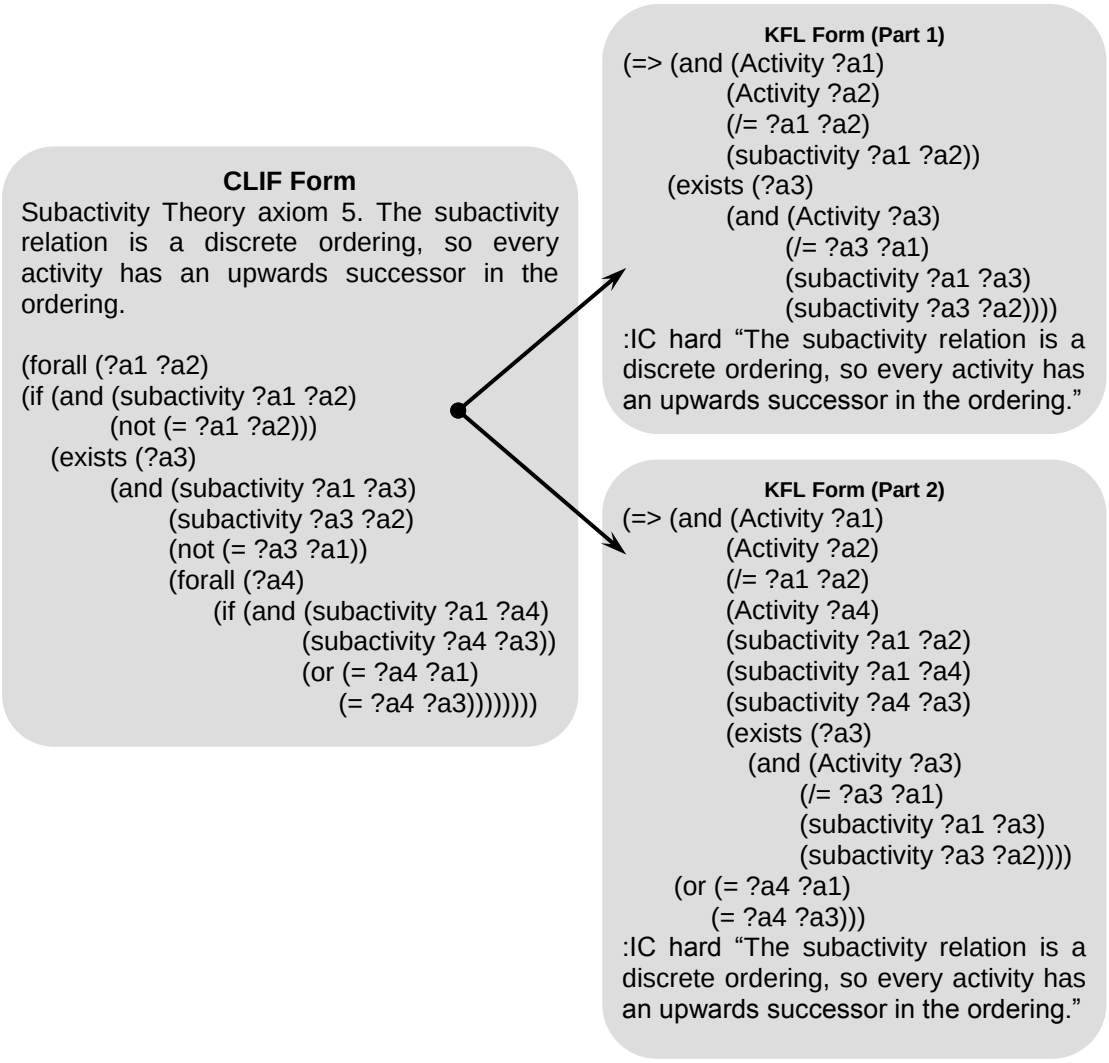
The example illustrates an original CLIF-written axiom from PSL (PSL Core axiom 13) versus its implemented form in KFL. Statements with “forall” are redundant and are, therefore, removed from logical statements because in KFL, written axioms already have an implicit universal quantification over them (i.e. although “forall” is not physically identified, it nevertheless is present in any axiom). The above-mentioned modifications to PSL axioms do not change their behaviour. In other words, original semantics are fully preserved.

Another obstacle faced with the implementation of CLIF-based PSL to PSL expressed in KFL is concerned with some of the very complex PSL axioms which have to be broken down into smaller axioms for better manageability in the IODE environment. One such example is captured in Expression 7-2 where in the CLIF form of the axiom, more than one “if-then” statement is nested into one another, which in IODE creates confusion. In the example, axiom 5 from PSL Outer-Core Theory of Subactivities has been split into two parts, the consequence of which is the same as expressing the more complex single axiom. Only few of these very complex axioms (five in all) arising in PSL Outer-Core Theory of Subactivities and Theory of Discrete States have been broken down.

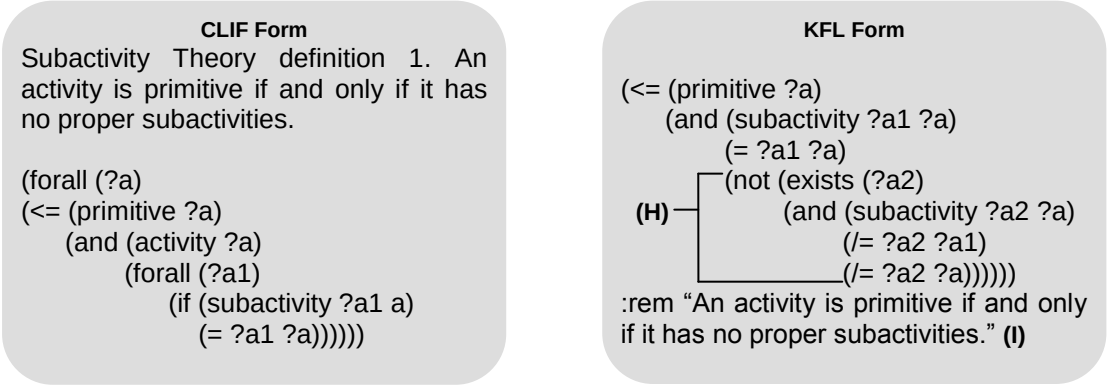
From an IODE implementation viewpoint, very few PSL rules need to be modified in order to enhance their interpretation. One such example is the logical condition in definition 1 from PSL Outer-Core Theory of Subactivities. A definition in PSL is analogous to an inference rule as opposed to an integrity constraint, hence explaining why a definition is appended with a remark line “:rem” **(I)** instead of “:IC” as identified in Expression 7-3.

The PSL definition is used for inferring instances of the class “Activity” as being “primitive” based on the fact that these instances do not have any proper subactivities. However, based on PSL semantics, any “Activity” instance is a subactivity of itself. If the definition were left as per its CLIF from in Expression 7-3, this would lead to the inference that even complex activities are “primitive” since complex activities, in addition to having proper subactivities, are also a subactivities of themselves. For this reason, the

definition of a “primitive” activity is extended in KFL **(H)** for not inferring complex activities as being “primitive”. During the implementation of PSL, only very few rules have been extended. However, this brings forward an improvement of PSL process semantics from an implementation perspective.



Expression 7-2 Splitting a PSL Axiom into Two Parts for Improving Manageability in IODE



Expression 7-3 Improving the Logical Interpretation of a PSL Definition

7.3.1.3 Exploring the Implemented Foundation Layer

After all the KFL files, containing the relevant ontological content for the heavyweight manufacturing ontological foundation have been parsed, loaded and saved in the OMS, it becomes possible to browse through the Foundation Layer. This is illustrated in Figure 7-3.

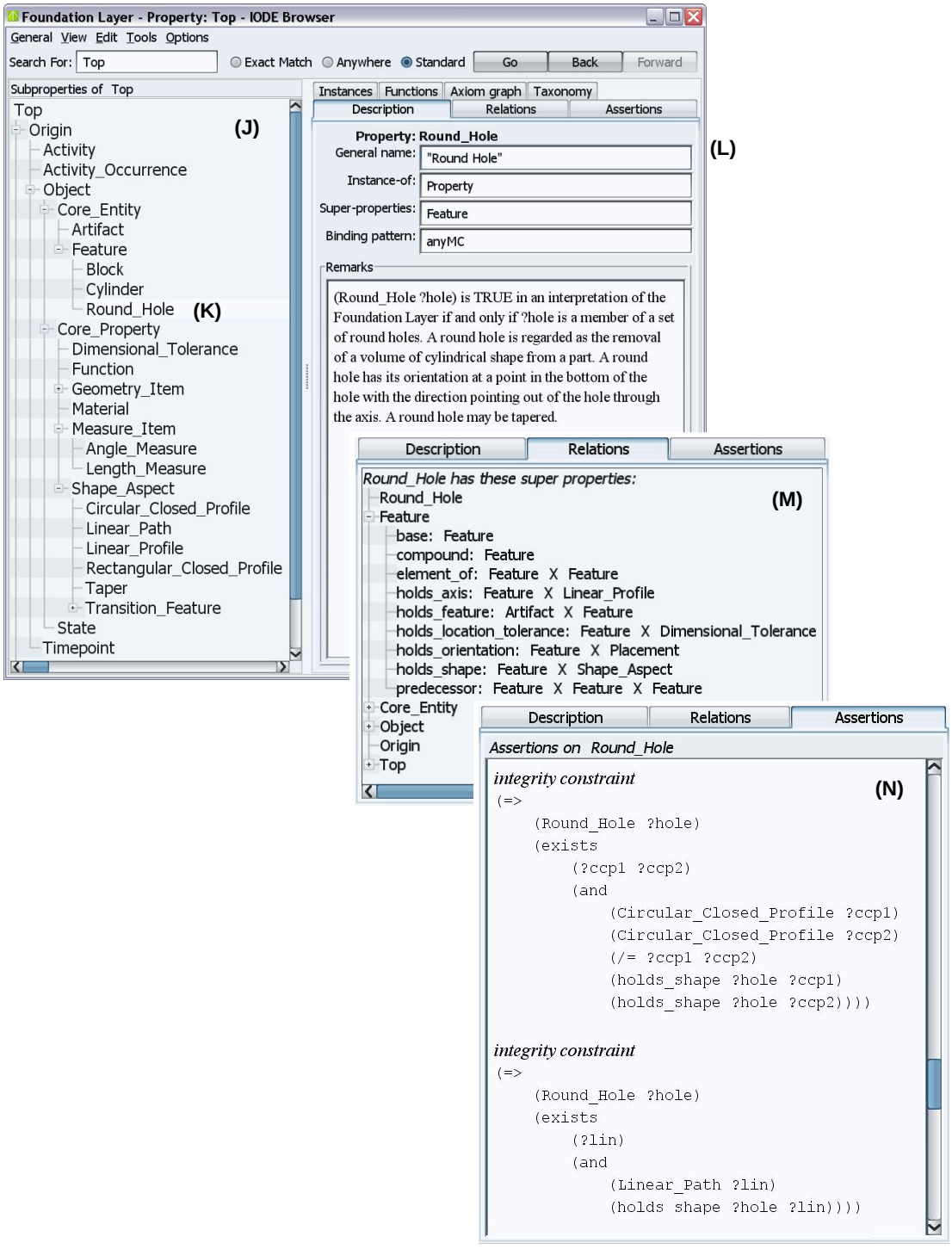


Figure 7-3 Browsing through the Implemented Foundation Layer

The screenshots indicate the possibility of browsing through the taxonomy of classes present in the Foundation Layer **(J)**. These classes form the backbone of the heavyweight manufacturing ontological foundation. In the figure, a majority of the developed classes are shown. By selecting a particular class like “Round_Hole” **(K)**, the user is able to view a number of aspects relevant to “Round_Hole”. For instance, the “Description” tab **(L)** allows the user to view general information about the class. This includes natural language remarks which informally describe the intuition behind “Round_Hole”. It is also possible to analyse the defined relations over “Round_Hole” by switching to the “Relations” tab as in **(M)**. Note that in this case no specific relation is defined explicitly over “Round_Hole”. This is because all relevant relations are inherited from its parent class “Feature”.

Axioms over classes, i.e. integrity constraints as well as definitions (inference rules), can be viewed by selecting the “Assertions” tab **(N)** in the IODE browser. In the example in Figure 7-3, two of the ICs governing two necessary conditions over “Round_Hole” are shown. It is important to emphasise at this point that once an ontology is saved in an OMS, it cannot be manipulated within the environment, i.e. modified in IODE itself. Any modification needs to be carried out in the relevant KFL file(s) prior to being re-saved to a new OMS.

7.3.2 Implementation of the Domain Ontology Layer

The implementation of the Domain Ontology Layer follows a similar approach to that of the Foundation Layer. The knowledge engineering methodology (Noy and McGuinness, 2001) is also applied for this purpose. Four different domain models are under consideration namely:

- A “Machining Hole Feature Ontology A” which treats the definition of different types of hole features based on the machining and process planning viewpoints.

- A “Machining Hole Feature Ontology B”, which focuses on the definition of a set of hole features based on the machining and process planning viewpoints.
- A “Design Hole Feature Ontology A” which entails an ontological model based on a design function viewpoint of the “Machining Hole Feature Ontology A”.
- An “ISO Tolerance Band Model” for round holes, based on ISO Tolerance Band and machining processes associated with ISO IT Tolerance Grade (ISO 286-2, 1988). This model serves as an external domain model to experiment with semantic mapping concepts based on external domains (refer to Chapter 6 section 6.2.2.3).

Figure 7-4 illustrates the creation of OMSs for the four domain ontologies under consideration. To create an OMS for any domain ontology that follows the SMIF approach, the Foundation Layer OMS is first cloned. The new OMS is renamed at convenience, and the KFL file for the required domain ontology is loaded in the new OMS. The implementation of domain models is not discussed further in this section as this constitutes an element of the case study in Chapter 8, where the implementation of integrity-driven domain models specialised from the Foundation Layer is debated in more detail. The content of the various domain models can be consulted in Appendix D.

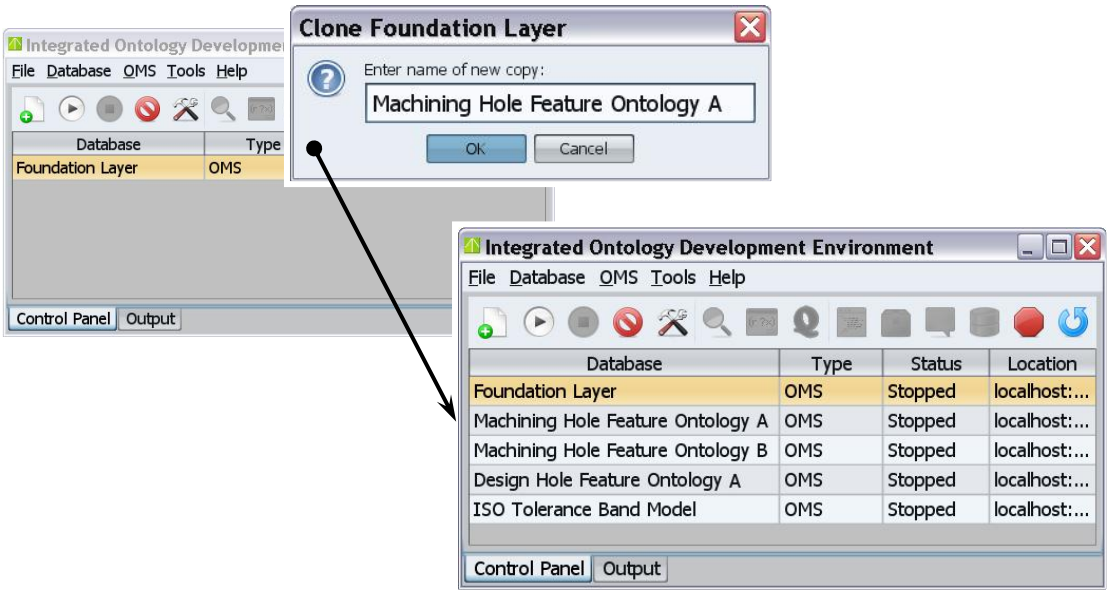


Figure 7-4 Creating OMSs for Four Domain Models under Investigation

7.3.3 Implementation of the Semantic Reconciliation Layer

The development process of semantic mapping concepts for semantic alignment follows the ontology alignment lifecycle (Euzenat *et al*, 2008; Shvaiko and Euzenat, 2008). This is because of the iterative process required during the creation, testing and modification (if needed) of these semantic mapping concepts for their optimised implementation. The meaning behind semantic mapping concepts (both formal and informal) required for the implementation of the Semantic Reconciliation Layer is captured in KFL files. Appendix E exposes a subset of the types of semantic mapping concepts used in the Semantic Reconciliation Layer.

If, for example, semantic mapping concepts based on foundation semantics are to be deployed, then the corresponding KFL file is loaded after the merging stage is completed for two domain models to be reconciled. Similarly, if semantic mapping concepts based on an external domain is required, then the KFL file containing the ontological content of the external domain has to be loaded after the merging stage is performed.

In this section, the implementation of semantic mapping concepts based on foundation semantics is explained. The case study in Chapter 8 further elaborates the other aspects of the Semantic Reconciliation Layer, for example, (1) how to reconcile cross-domain arguments based on known cross-domain correspondences and (2) how to reconcile cross-domain arguments based on external domains.

The KFL file containing all the semantic mapping concepts based on foundation semantics is loaded and saved in an OMS. The creation of this OMS follows the context adjustment and merging processes for two domain models to be reconciled. When the semantic mapping concepts are saved in the OMS, it then becomes possible to browse through them.

7.3.3.1 Semantic Mapping Concepts for Reconciling Classes

Figure 7-5 illustrates some of the semantic mapping relations for reconciling cross-domain classes **(O)**. Notice the "classMappingRelation_018" **(P)** previously explained in section 6.2.2.1 of Chapter 6. Further browsing into this semantic mapping relation provides options for viewing the logical sentence **(Q)**. This logical sentence provides the formal definition of the relation "classMappingRelation_018". By switching to the "Description" tab for the relation, the user is able to view the informal semantics in the form of written remarks **(R)**. Also notice within the window in **(R)** the presence of highlighted foundation concepts. These concepts are hyperlinked to their relevant locations in the Foundation Layer in case the user wishes to refer to these concepts too.

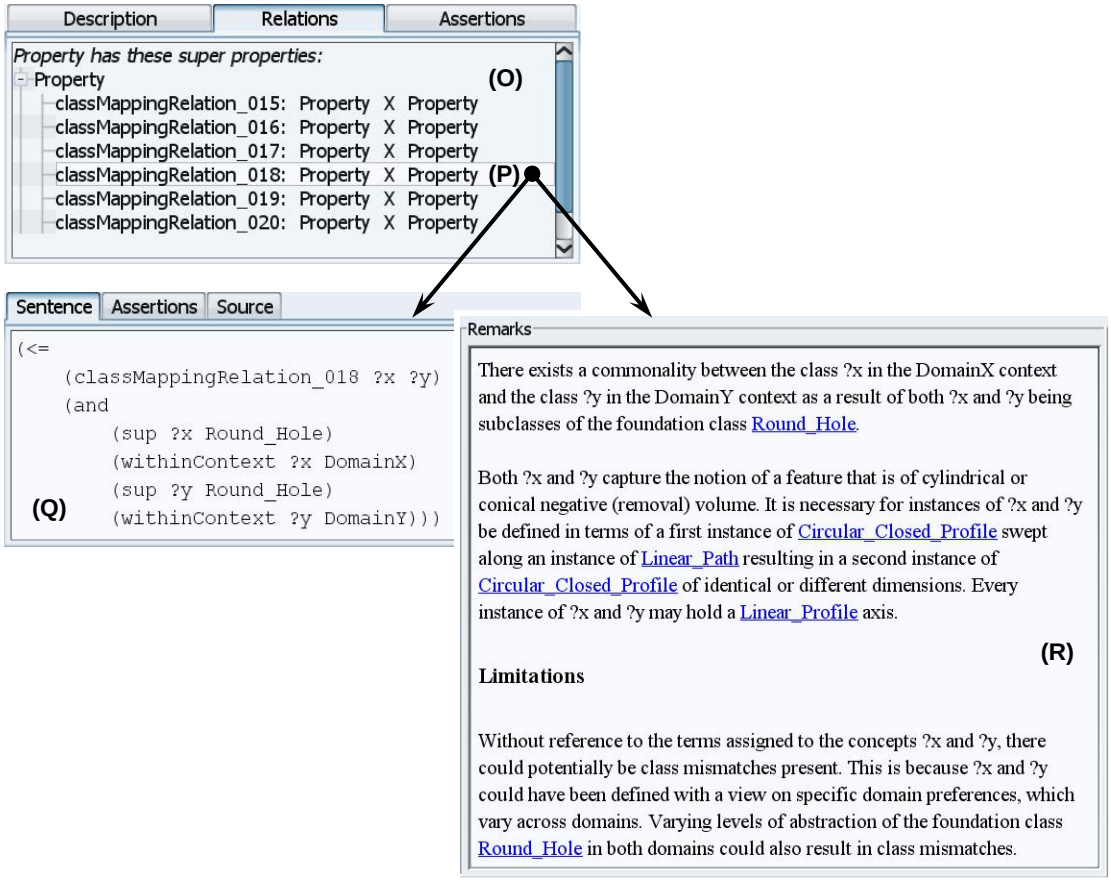


Figure 7-5 Implementation of Semantic Mapping Relations for Reconciling Cross-Domain Classes

7.3.3.2 Semantic Mapping Concepts for Reconciling Instances

Figure 7-6 depicts a portion of the semantic mapping concepts based on foundation semantics, explored for the reconciliation of cross-domain instances. The semantic mapping concept “instanceMappingRelation_041” **(S)**, also previously explained in section 6.2.2.1 of Chapter 6, is highlighted. As with all other semantic mapping concepts based on foundation semantics, “instanceMappingRelation_041” is also accompanied by its formal definition and the adequate tagged remarks.

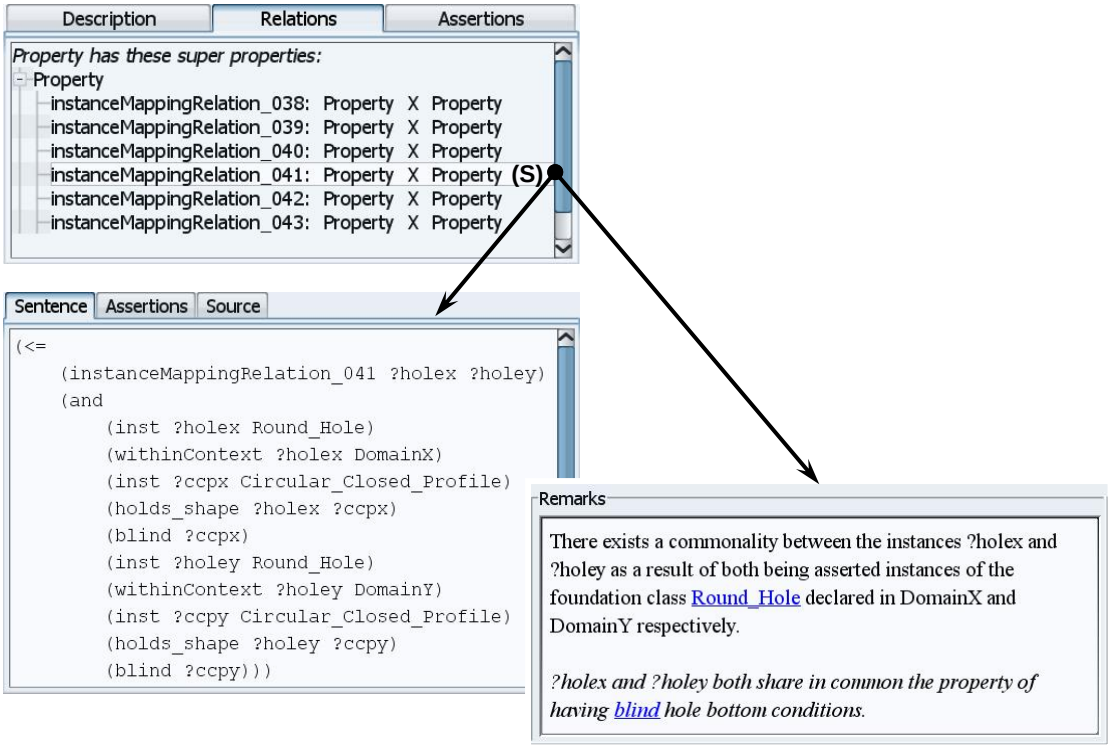


Figure 7-6 Implementation of Semantic Mapping Concepts for Reconciling Cross-Domain Instances

7.3.3.3 Semantic Mapping Concepts for Reconciling Ontological Functions

The implementation of semantic mapping concepts based on foundation semantics also involves reconciling at the ontological function level of domain ontologies. Figure 7-7 identifies four such mapping relations where the logical definition as well as the informal semantics of the relation

"functionMappingRelation_003" **(T)** are illustrated. This semantic mapping concept infers correspondences between cross-domain units of measurement functions for denoting instances of the foundation class "Length_Measure" **(U)**.

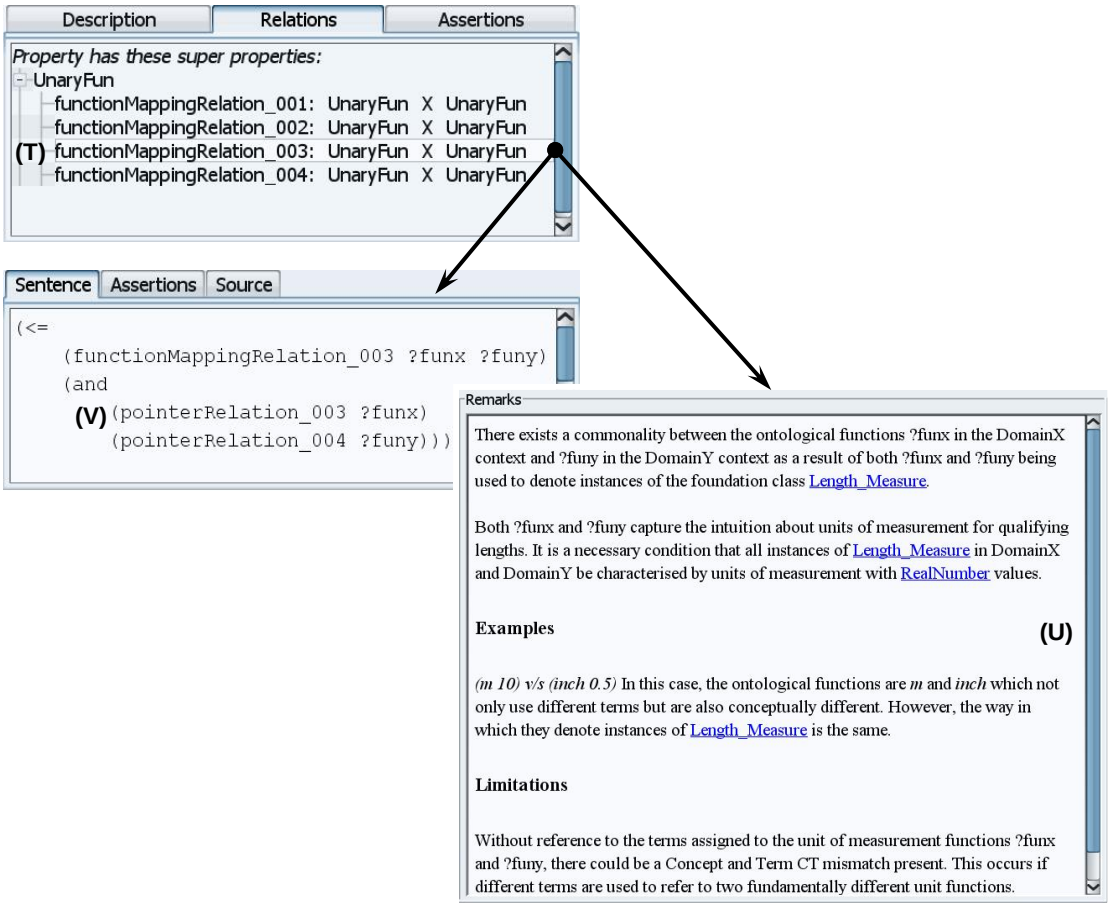


Figure 7-7 Implementation of Semantic Mapping Concepts for Reconciling Cross-Domain Ontological Functions

The logical conditions that define "functionMappingRelation_003" **(T)** are relatively complex and for this reason, the logical statement has to be split for better manageability. During implementation, "pointer relations" such as "pointerRelation_003" **(V)** are defined to provide a better facility to infer over complex logic.

7.3.4 Implementation of the Interoperability Evaluation Layer

There are two main components used in the Interoperability Evaluation Layer for the discovery of correspondences. The first is a graphical Web-based user interface, developed in this work, called the Interoperability Evaluation Assistant. This user interface applies the matrix configuration identified in section 6.3.2 of Chapter 6 in order to improve the management of queries and user interaction, before these queries can be executed. Appropriate queries are accessed from the Interoperability Evaluation Assistant and run in the second component in the Interoperability Evaluation Layer. This second component is the query tool facility provided in IODE.

7.3.4.1 Interoperability Evaluation Assistant

During the development of the Interoperability Evaluation Assistant, a number of software tools and programming languages have been harnessed. These include:

- Microsoft Office FrontPage 2003 (Microsoft Office FrontPage Homepage, 2009). This application allows the development of Web-based interfaces and has, therefore, been exploited towards the development of the Interoperability Evaluation Assistant. Where necessary, the scripting language JavaScript has been used to control user inputs in text fields and for outputting relevant messages.
- Adobe Flash 8 (Adobe Website, 2009). This application allows more complex Web-based interfaces to be realised, with the advantage of nesting several possible user actions within one page instead of requiring multiple pages. The scripting language ActionScript 2.0 has been utilised in Adobe Flash 8 for enabling the coordination of dynamic content present within the Interoperability Evaluation Assistant.
- Adobe Photoshop CS (Adobe Website, 2009). The manipulation of graphical content to go on the interface has been performed through the application of this image editing software.

Two main aspects have been taken into consideration for selecting a Web-based approach towards the realisation of the user interface. Firstly, a Web-based approach has been chosen because of the recognised information sharing benefits of Web-based architectures for collaborative product development (Rodriguez and Al-Ashaab, 2005). This implies that a Web-based interface is a useful way of supporting an interoperability-enabled environment. Secondly, a Web-based interface is relatively straightforward to implement, when viewed from the author's experience. Appendix F highlights the sitemap and sample codes used in the development of the interface.

Figure 7-8 identifies the main panel of the interface. Two ways of building queries are supported namely by (1) allowing the user to look for specific semantic mapping concepts to query, using the matrix configuration **(W)** and (2) allowing the user to build queries to retrieve all semantic mapping concepts that hold between two known cross-domain arguments **(X)**.

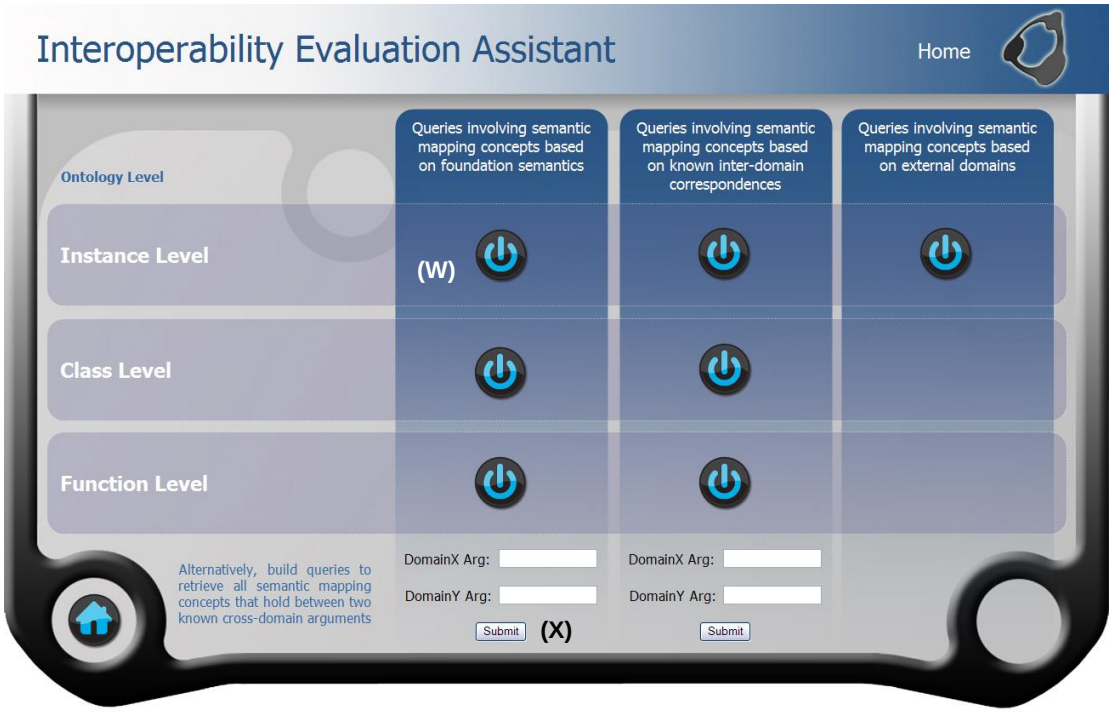


Figure 7-8 Main Panel of the Interoperability Evaluation Assistant

Figure 7-9 identifies how the Interoperability Evaluation Assistant helps the user to browse through specific semantic mapping concepts to be queried (also see Appendix F if needed) In the first place, the user switches on the

relevant cell in the matrix, for example, “Queries involving semantic mapping concepts based on foundation semantics” against the “Instance Level” **(Y)**. Using the taxonomical breakdown of foundation classes **(Z)**, the user goes to the relevant concept in question, in this case “Round_Hole” **(A1)**. From the set of possible semantic mapping concepts that may exist, the user then selects the intended query and clicks on the download button **(B1)** to retrieve the logical query **(C1)**. The query can then be copied and pasted in the query tool provided in IODE for processing. Note that the storage of these queries has been done using a simple folder-based method. For even better manageability, this method would preferably be a database storage facility.

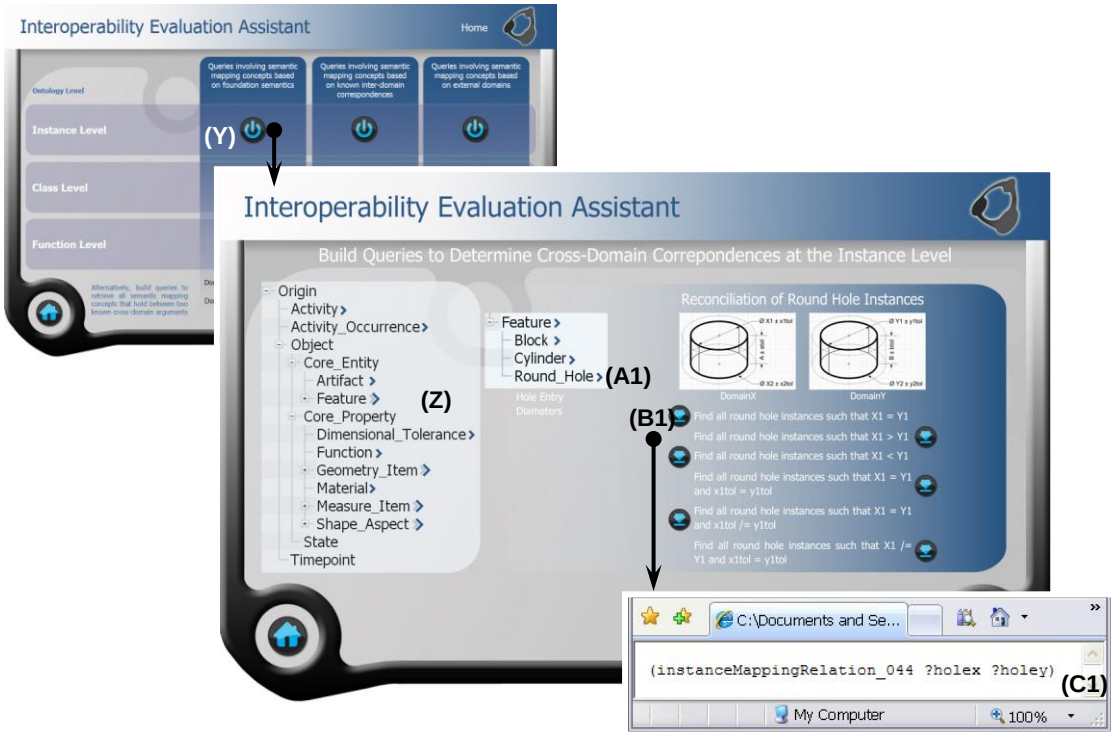


Figure 7-9 Retrieving a Specific Interoperable Knowledge Query

Figure 7-10 depicts the other way in which the interface can be used, i.e. to build queries for retrieving all semantic mapping concepts that hold between two known cross-domain arguments. For so doing, the additional JavaScript-supported facility (refer to label **(X)** in Figure 7-8) is utilised. Suppose the user, by browsing through an IODE OMS containing two merged domain models, comes across two classes named “Primary_Hole” in the “DomainX” context and “Drilled_Hole” in the “DomainY” context. The user, at this point, wishes to

build a query to find out all the semantic mapping concepts based on foundation semantics, that hold between these two classes.

The query procedure consists of opening the main panel of the Interoperability Evaluation Assistant and using the JavaScript-supported facility, the names of the two arguments are typed in the provided text fields **(D1)**. On clicking the submit button, a new window opens **(E1)** where the more complex query can be retrieved. The query is selected and copied **(F1)** prior to being pasted into IODE's query tool for being processed.

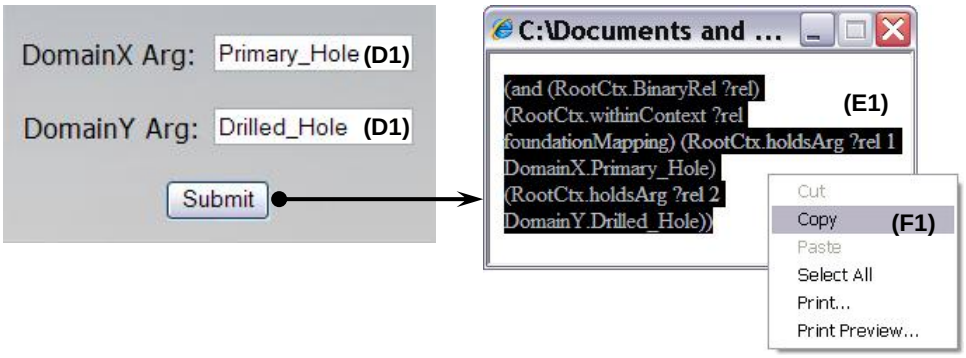


Figure 7-10 Building a Query to Retrieve All Semantic Concepts that Hold between Two Known Cross-Domain Arguments

7.3.4.2 The Query Tool in IODE

After the appropriate query is accessed from the Interoperability Evaluation Assistant (refer to Figure 7-10), the query is pasted into the query editing window of the query tool in IODE (see label **(G1)** on Figure 7-11). On clicking the "Run query" button, all the results of the query can be viewed as a table of results. Notice the presence of "classMappingRelation_018" **(H1)**, which is one of the correspondences that hold between the class "Primary_Hole" in "DomainX" and "Drilled_Hole" in "DomainY". The user is able to further browse into the details of the query result by selecting it and viewing its tagged remarks **(I1)**.

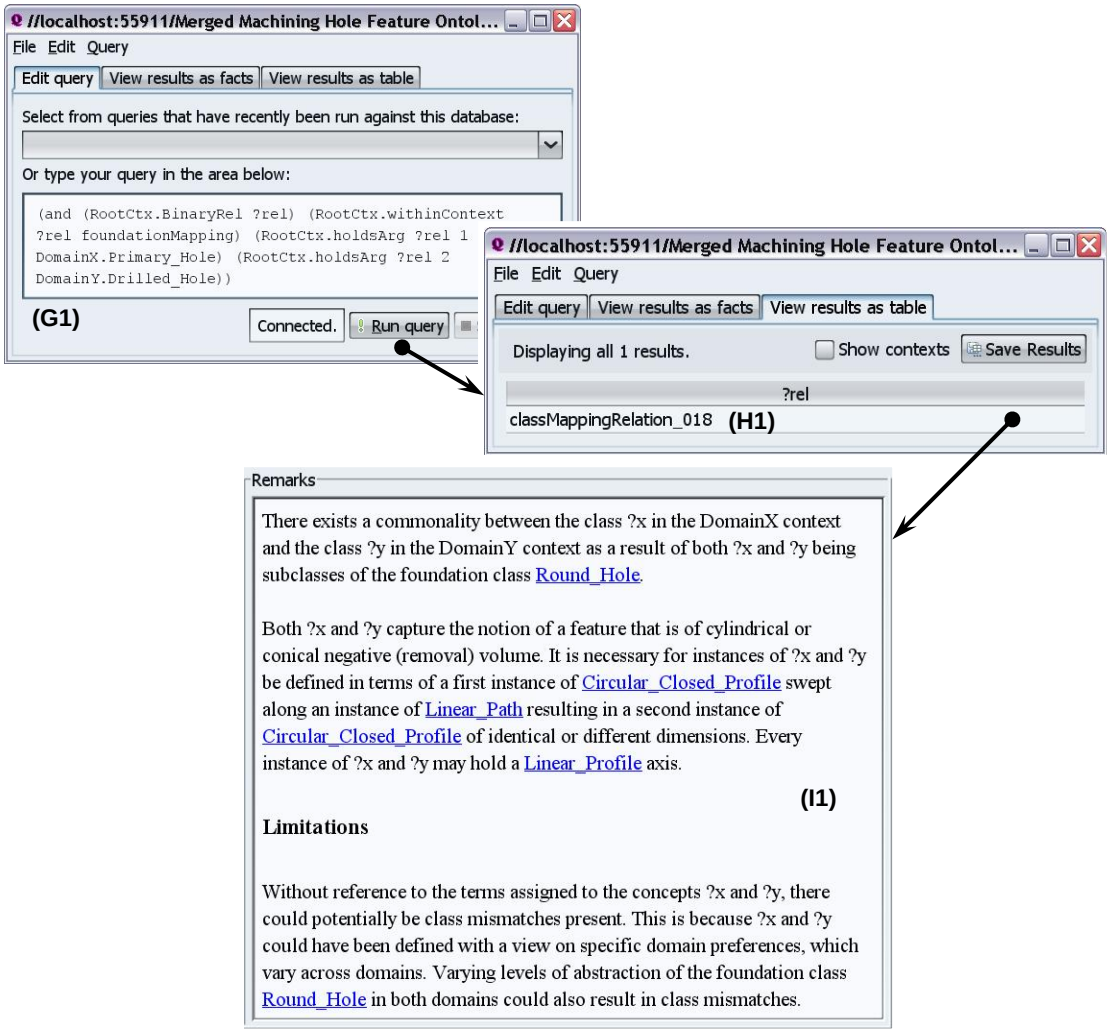


Figure 7-11 Executing an Interoperable Knowledge Query and Viewing Its Results

7.3.4.3 Logically Verifying Query Results

Results obtained from running a query can then be verified through logical proof for enhancing the user's awareness of why the query results portray certain semantic mapping concepts. The verification process also utilises IODE's query tool. By switching to the "View results as facts" window (see label **(J1)** on Figure 7-12) an option for launching the proof procedure for each query result becomes available. On clicking this link, the proof structure for a specific query result is made visible **(K1)**. A proof structure is accompanied by both an informal interpretation **(K1)** as well as a formal one expressed in logic form (not shown on Figure 7-12).

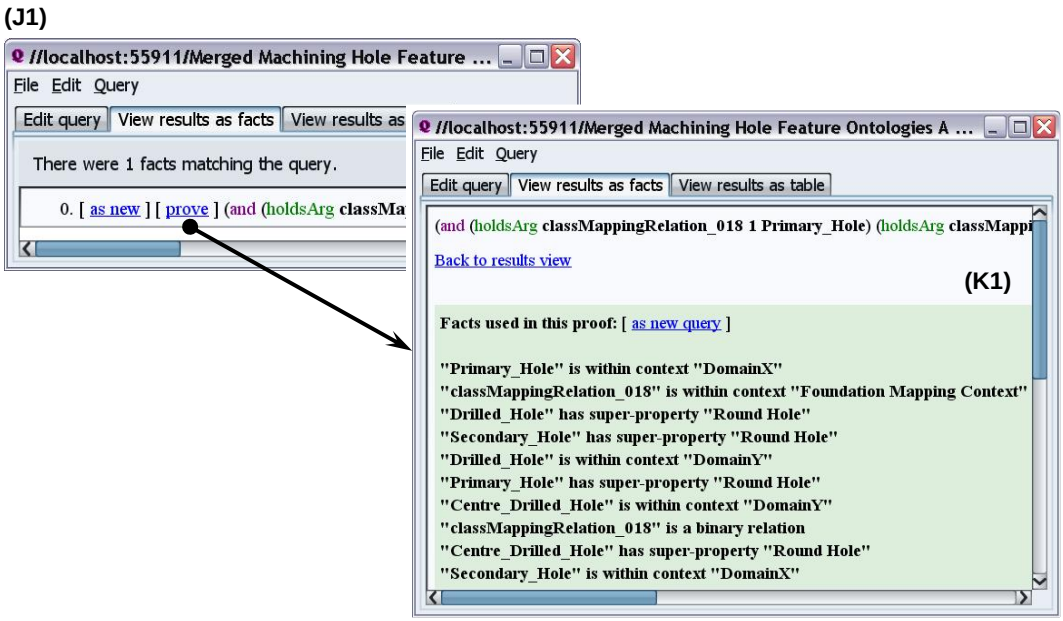


Figure 7-12 The Verification of an Evaluated Query Result using Logical Proof

7.4 Summary

This chapter has described the core details involved in the development and deployment of an experimental system for testing the SMIF approach. This consequently meets the fourth objective of this research (see Chapter 1 section 1.3.1). The implementation has been decomposed into a number of stages that hold for each specific level of SMIF, where the primary implementation environment exploited being IODE, the latter supporting the development of ontologies expressed in KFL. IDEF5 schematics used for the exploration of concepts in the heavyweight manufacturing ontological foundation (see Appendix C), have helped implementation into KFL and deployment in IODE. The competency question to be answered is as follows (see section 7.3.1):

- Can the Knowledge Framework Language (KFL) and IODE be used to formally capture and represent the heavyweight semantics required for a fully functioning Foundation Layer?

It is evident from the implementation that the full semantic capability required for establishing the Foundation Layer is acquired, although, for example, some modifications related to PSL axioms and definitions need to be performed during implementation. However, it is seen that these modifications do not affect semantic integrity (i.e. there is no actual loss of meaning in computational form).

In the Domain Ontology Layer, various domains employ the implemented Foundation Layer to build, in an integrity-driven way, their own tailored domain models. The case study in Chapter 8 analyses this aspect in more detail, following the success in the implementation of the Foundation Layer. In the event that a pair of domain models need to be reconciled with the intention of identifying the correspondences that hold between the two, the Semantic Reconciliation and Interoperability Evaluation layers are deployed.

The pair of domain models to be reconciled undergo the simple merging procedure under the explored ontology mapping process concepts from Chapter 6. The domain models are merged in a distinct Object Management System (OMS) where, based on the intention of the user, the relevant set of semantic mapping concepts (available as KFL files) are loaded in the OMS in question. Section 7.3.3 has illustrated that semantic mapping concepts based on foundation semantics can be made relevant to different levels of domain models namely the (1) class level, (2) instance level and (3) function level.

Mapping discovery and the interpretation of cross-domain correspondences between domain models is carried out at the fourth level of the framework. Two mechanisms are applied for this purpose. A Web-based user interface, the Interoperability Evaluation Assistant developed in this work, facilitates the retrieval of the correct query to be processed. After the query is obtained, the latter is simply copied and pasted in the query tool provided in IODE. The results obtained from a query action are tabulated. These results can be browsed or proved in order to support the verification of evaluated cross-domain correspondences.

8 Case Study

8.1 Introduction

This chapter explores a number of test cases as part of a complete case study in order to provide a proof of concept for the overall deployment of the Semantic Manufacturing Interoperability Framework (SMIF), whose underlying understanding has been discussed in the previous chapters. The case study has been set in order to test the research hypothesis identified in Chapter 1. A re-statement of the research hypothesis is given below:

- The specification of a heavyweight manufacturing ontological foundation can provide a basis for the integrity-driven specialisation of domain models, while supporting the capability to evaluate and verify the correspondences between pairs of domain models that have been specialised from the foundation.

The different test cases are thus oriented on the research hypothesis, where the appropriate aims and objectives have been identified for each test case. The results gathered from the case study are presented and necessary discussions and validation of results are exposed at the end of each test case. Section 8.2 provides a global picture of the intended test cases as well as the identification of relevant case study boundaries and assumptions. Four test cases are then analysed in sections 8.3, 8.4, 8.5 and 8.6. Finally, section 8.7 provides a summary of the main findings from the test case implementations.

8.2 Overview of Test Cases

8.2.1 The Arrangement of Test Cases in the Case Study

The various test cases are aimed at specific levels of the SMIF notably the Domain Ontology, Semantic Reconciliation and Interoperability Evaluation layers. Note that at this point, a fully functioning and validated Foundation Layer has already been implemented (see section 7.3.1 in Chapter 7 and

Appendix C) and this implementation is, therefore, not further documented in the case study.

Figure 8-1 visually illustrates how the test cases are arranged. The test cases involve the development of three domain models in the first place, namely a "Machining Hole Feature Ontology A" **(A)** within a "System Domain A", a "Machining Hole Feature Ontology B" **(B)** within a "System Domain B" and a "Design Hole Feature Ontology A" **(C)** pertaining to the "System Domain A". Test Case 1 firstly analyses the integrity-driven development of the "Machining Hole Feature Ontology A" facilitated through the heavyweight semantics residing in the Foundation Layer.

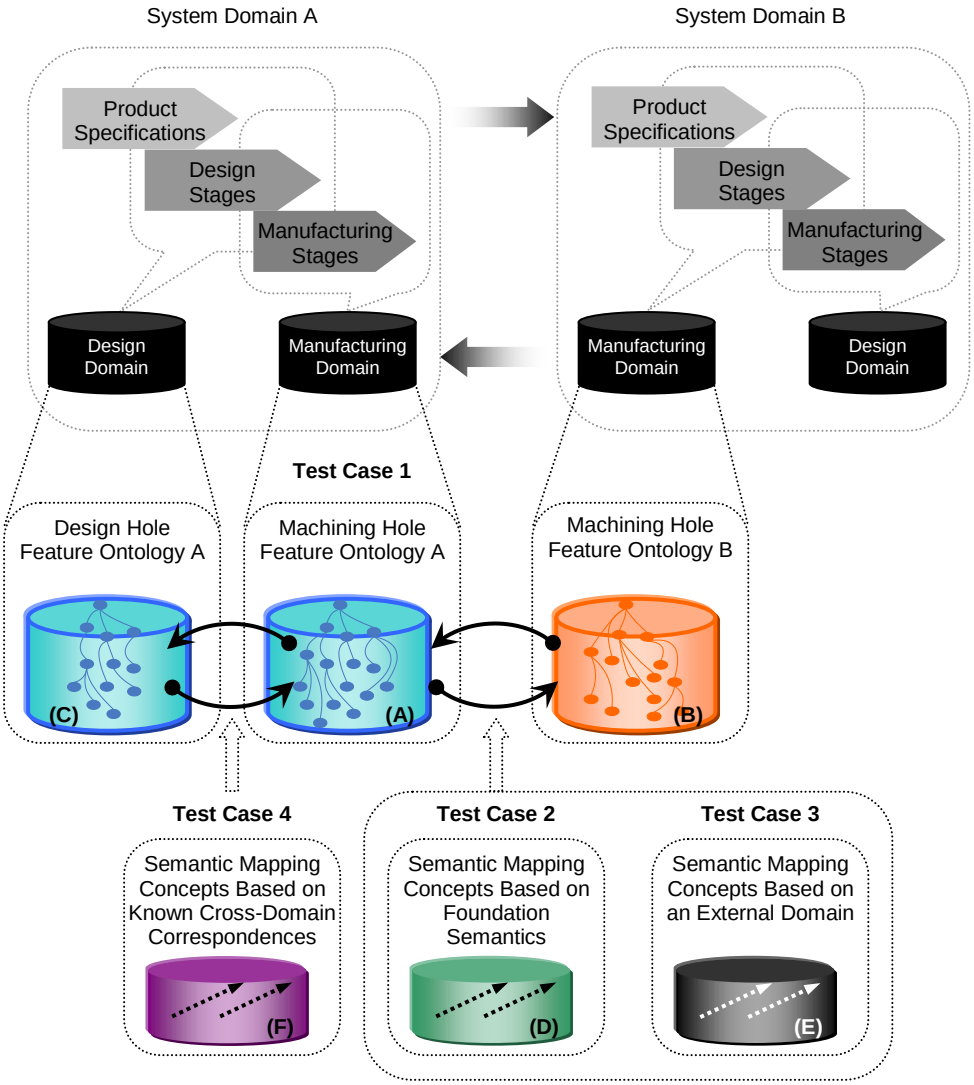


Figure 8-1 The Arrangement of Test Cases in the Case Study

Test Case 2 focuses on the reconciliation of two domain models, “Machining Hole Feature Ontology A” **(A)** and “Machining Hole Feature Ontology B” **(B)** developed within the system domains A and B respectively. In Test Case 2, reconciliation is established using semantic mapping concepts based on foundation semantics **(D)**. Test Case 3, on the other hand, also aims at reconciling “Machining Hole Feature Ontology A” and “Machining Hole Feature Ontology B”, but instead, the reconciliation is driven through semantic mapping concepts based on an external domain ontology **(E)**. Both test cases 2 and 3 are targeted at inter-system interoperability.

Test Case 4 considers intra-system interoperability. Another domain model pertaining to the “System Domain A”, identified as “Design Hole Feature Ontology A” **(C)**, is developed for this purpose. The ontology captures the concepts from the “Machining Hole Feature Ontology A” **(A)** but aligned to a functional design viewpoint. The “Design Hole Feature Ontology A” and “Machining Hole Feature Ontology A” are then reconciled utilising semantic mapping concepts based on known cross-domain correspondences **(F)**. Test cases 2, 3 and 4 altogether explore the different modes in which semantic mapping concepts occur.

All domain models explored in the test cases have been developed following the Knowledge Engineering Methodology (Noy and McGuinness, 2001). The types of hole feature concepts defined in the various test cases have been partly inspired from (1) hole feature terminologies obtained from company sources, (2) sources such as Canciglieri (1999) and NX (Siemens PLM Software Website, 2009) terms for holes and (3) the author’s preferences and experience of the research scope.

8.2.2 Case Study Boundaries and Assumptions

The main boundary set is concerned with a restriction to the scope of the research, which is centred around the formal representation of hole features in design and manufacture and the representation of hole making process sequences. Furthermore it is assumed that all the domain models being

developed within the framework follow the controlled specialisation approach and use IODE as a common implementation platform. It is also assumed that the formalised domain integrity constraints accurately capture the intended informal meaning. Other boundaries and assumptions previously identified in section 4.7 of Chapter 4 also apply to the case study.

8.3 Test Case 1: Integrity-Driven Specialisation of a Machining Hole Feature Ontology

8.3.1 Aim and Objectives

The aim of Test Case 1 is to prove the ways in which the Foundation Layer facilitates the specialisation of a “Machining Hole Feature Ontology A”, such that a semantically rich and accurate representation of the ontology-based model is obtained. In this first test case, the following competency questions have been formulated:

- Can the ontological mechanisms that allow specialisation to occur be used during the development of “Machining Hole Feature Ontology A”?
- Can the specification of domain-defined integrity constraints be achieved in a flexible way while not violating foundation semantics?
- Is it possible to accurately represent discrete knowledge through instantiation, based on the semantics captured in the “Machining Hole Feature Ontology A” and foundation semantics?

There are two main objectives involved namely (1) the deployment of the Foundation and Domain Ontology layers of the Semantic Manufacturing Interoperability Framework in order to analyse the test case and (2) the use of the relevant set of tools and ontological formalism depicted in Chapter 7 notably IODE (Ontology Works Inc., 2009) as ontology development environment.

8.3.2 Machining Hole Feature Ontology A

The diagram in Figure 8-2 provides a view on the type of part family being investigated, where in this scenario, a “Housing_Part_Family” is considered. It is to be pointed out that the diagram does not represent a concrete state of the domain ontology, i.e. instances of the concepts from the ontology, but in fact reflect some of entity information classes being developed in the “Machining Hole Feature Ontology A”. Additionally, although the classes “Turned_Flange” and “Turned_Boss” are referenced in the domain ontology, these are primarily present to provide a context for the existence of the hole features held by the “Housing_Part_Family”.

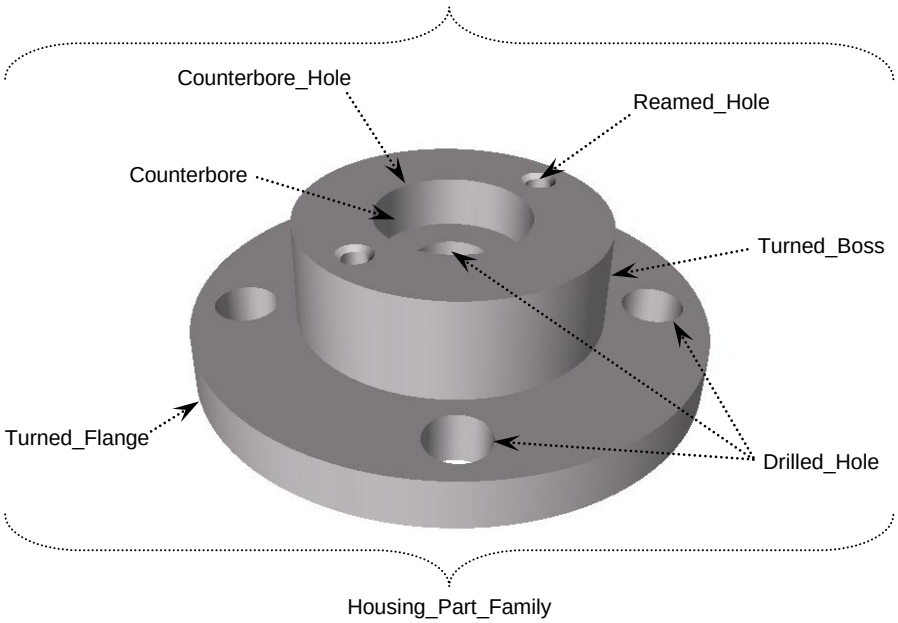


Figure 8-2 Examples of Classes Developed in the “Machining Hole Feature Ontology A”

8.3.2.1 Entity Information Semantics

The taxonomy of entity information classes for the “Machining Hole Feature Ontology A” is shown in Figure 8-3 **(G)**. Some of the classes present are “Housing_Part_Family” **(H)** specified as a sub-class of the foundation class “Artifact”, “Counterbore_Hole” **(I)** as a sub-class of “Feature” and “Drilled_Hole” **(J)** as a sub-class of “Round_Hole”. Consider the domain-defined class “Counterbore_Hole”. The latter is defined as a sub-class of the

foundation class “Feature” using the super-class/sub-class directive “:sup” **(K)**, in the KFL file of the domain ontology.

This is because “Counterbore_Hole” is a class of compound feature that aggregates the “Round_Hole” sub-classes “Drilled_Hole” **(J)** and “Counterbore” **(L)**, which themselves have their definitions based on domain-defined dimensional parameters. For example, “Drilled_Hole” **(J)** consists of “Drilled_Hole_Diameter” **(M)** and “Drilled_Hole_Depth” **(N)**, which are identified as sub-classes of the foundation class “Length_Measure”. The “Description” tab **(O)** views the remarks defined for “Counterbore_Hole”, while the “Assertions” tab **(P)** depicts two of the ICs defined for that class.

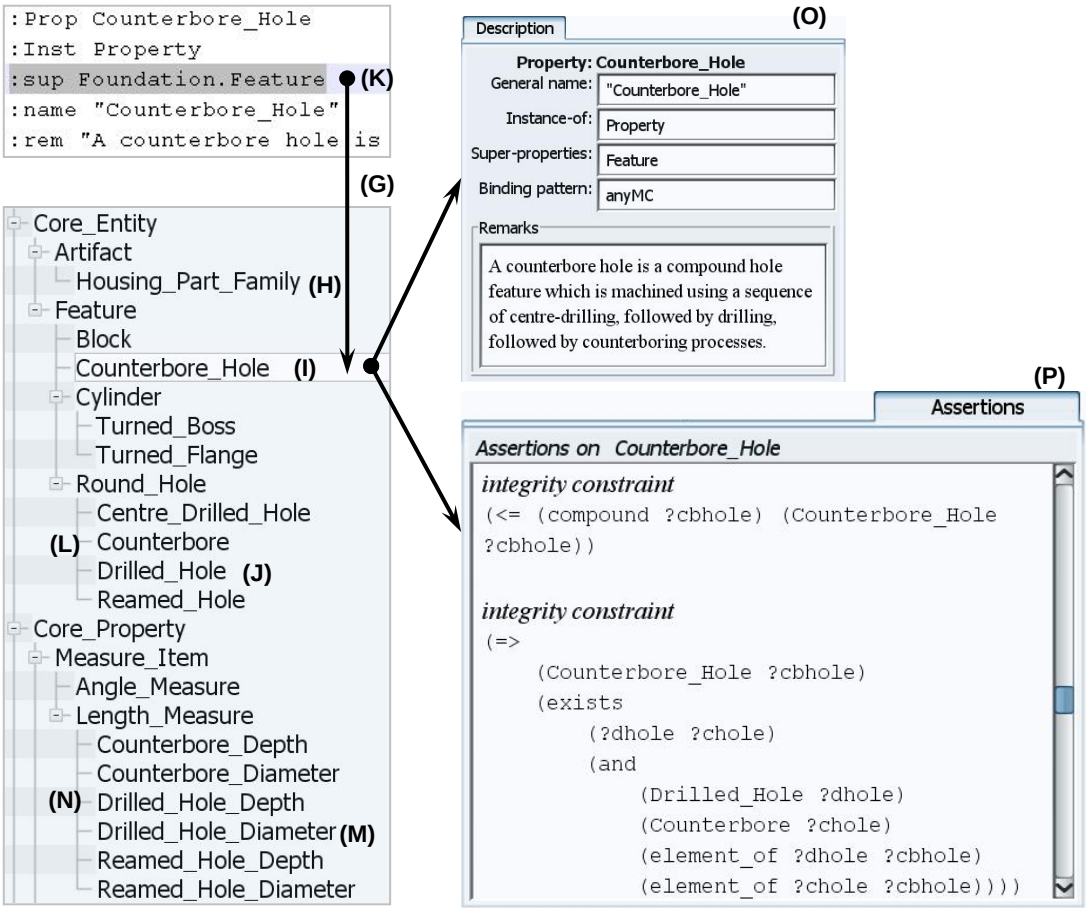
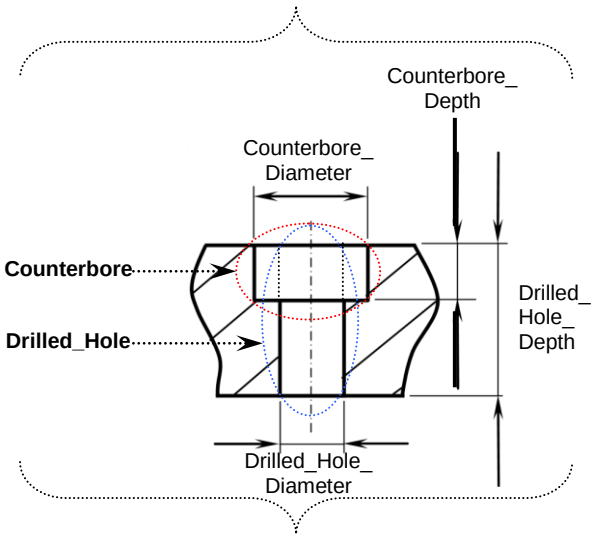


Figure 8-3 The Specialisation of Entity Information Classes in the “Machining Hole Feature Ontology A”

The diagram in Figure 8-4 illustrates the intuitions adopted in the “Machining Hole Feature Ontology A” for capturing the domain-defined axioms for the class “Counterbore_Hole” in terms of the “Round_Hole” classes that it

aggregates. A list of the several informal ICs is provided in Figure 8-4. Similar axioms have been formalised (refer to Appendix D.1 if needed) in order to have a semantically enriched model which is (1) consistent to the practices and preferences within “Machining Hole Feature Ontology A” as well as (2) consistent with foundation 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

Drilled_Hole

- Every drilled hole holds exactly two circular closed profiles of identical drilled hole diameter
- Every drilled hole holds exactly one linear path of drilled hole depth

Counterbore

- Every counterbore holds exactly two circular closed profiles of identical counterbore diameter
- Every counterbore holds exactly one linear path of counterbore depth

Figure 8-4 Entity Information Semantics for the class “Counterbore_Hole”

```
(=> (Counterbore_Hole ?cbhole)
    (Foundation.compound ?cbhole)) (Q)
:IC hard "A counterbore hole is a compound feature."
```

```
(=> (and (Counterbore_Hole ?cbhole)
         (Drilled_Hole ?dhole)
         (Foundation.element_of ?dhole ?cbhole))
    (Foundation.base ?dhole)) (S)
:IC hard "The drilled hole of a counterbore hole is the base feature of the counterbore hole."
```

```
(=> (Counterbore_Hole ?cbhole)
    (exists (?dhole ?chole)
        (and (Drilled_Hole ?dhole)
             (Counterbore ?chole)
(Foundation.element_of ?dhole ?cbhole) (R)
(Foundation.element_of ?chole ?cbhole))))
:IC hard "Every counterbore hole involves a drilled hole and a counterbore which are elements of the counterbore hole."
```

Expression 8-1 Example of ICs for the Class “Counterbore_Hole”

The formal logical statements written in KFL for the first three ICs of the class “Counterbore_Hole” are listed in Expression 8-1. These expressions capture

domain-defined ICs. It is important to notice how the specification of these axioms are based on the reuse of appropriate foundation semantics such as in line **(Q)** where the foundation unary relation “compound” is used, in line **(R)** where the “element_of” binary relation is used and in line **(S)** where the unary relation “base” is used.

8.3.2.2 Machining Process Semantics and Relationships to Entities

The taxonomy of machining process classes for the “Machining Hole Feature Ontology A” is illustrated in Figure 8-5. In the figure, a number of sub-classes of the foundation class “Activity” is present such as “Reamed_Hole_Making” **(T)**. The “:sup” directive has also been exploited for the purpose of creating the sub-classes of “Activity”. Similar to the previously explained example of the “Counterbore_Hole”, “Activity” sub-classes also carry informal semantics as captured in the “Description” tab **(U)** and formal ICs for semantic enrichment as shown in the “Assertions” tab **(V)** in Figure 8-5.

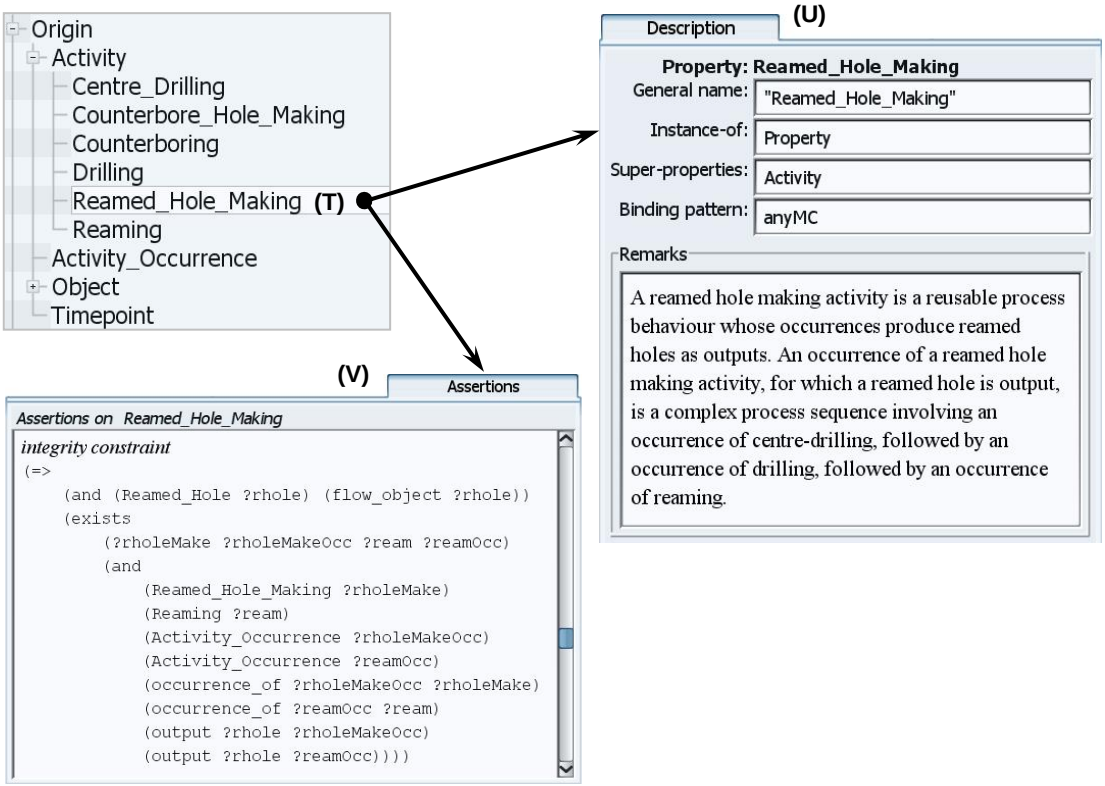
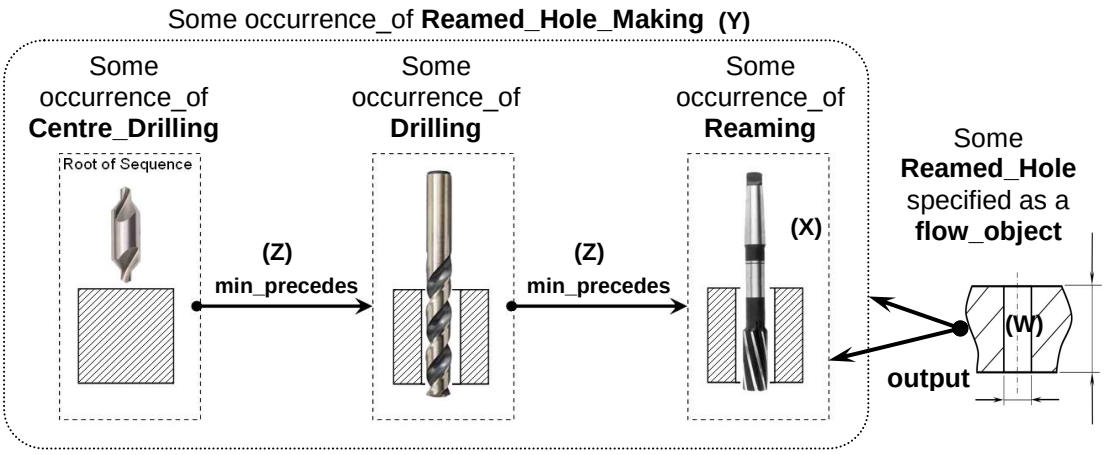


Figure 8-5 The Specialisation of Machining Process Classes in the “Machining Hole Feature Ontology A”

The IC exposed in Figure 8-5 **(V)** is an example of an axiom that governs the participation semantics between the individuals of the class “Reamed_Hole” and the corresponding activity occurrences. Informally, this soft IC states that “every reamed hole that is a flow object is both an output from a potential occurrence of a complex reamed hole making activity and an output from a potential occurrence of an atomic reaming activity." This understanding is captured in Figure 8-6, where it may be required that some instance of the class “Reamed_Hole” **(W)**, carrying the “flow_object” semantics, be specified as being an “output” of some “occurrence_of” the “Activity” class “Reaming” **(X)** and an “output” of some “occurrence_of” the “Reamed_Hole_Making” class **(Y)**.



Reamed_Hole_Making

- An occurrence of centre drilling must precede an occurrence of drilling under a complex occurrence of reamed hole making. Other behaviours under the complex reamed hole making activity may occur in between
- An occurrence of drilling must precede an occurrence of reaming under a complex occurrence of reamed hole making. Other behaviours under the complex reamed hole making activity may occur in between
- 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 8-6 Process Semantics for the class “Reamed_Hole_Making” and its Relationships to the Entity Class “Reamed_Hole”

The informal ICs defined for capturing rigorous semantics for the class “Reamed_Hole_Making” are also listed in Figure 8-6. The Expression 8-2 then reveals the formalised IC for the first informal axiom in the list from Figure 8-6. It is important to notice the use of the “min_precedes” relation (see Figure 8-6 **(Z)** and line **(Z)** in Expression 8-2) defined in PSL-based process

semantics coming from the Foundation Layer and reused in the domain ontology in order to capture the semantics of the process sequence under occurrences of “Reamed_Hole_Making”. A full list of the explored process-based ICs for the “Machining Hole Feature Ontology A” is also provided in Appendix D.1.

```
(=> (and (Reamed_Hole_Making ?rholeMake)
            (Foundation.Activity_Occurrence ?rholeMakeOcc)
            (Foundation.occurrence_of ?rholeMakeOcc ?rholeMake))
     (exists (?cDrill ?drill ?cDrillOcc ?drillOcc)
          (and (Centre_Drilling ?cDrill)
                (Drilling ?drill)
                (Foundation.Activity_Occurrence ?cDrillOcc)
                (Foundation.Activity_Occurrence ?drillOcc)
                (Foundation.occurrence_of ?cDrillOcc ?cDrill)
                (Foundation.occurrence_of ?drillOcc ?drill)
                (Foundation.min_precedes ?cDrillOcc ?drillOcc ?rholeMake)))) (Z)
:IC hard "An occurrence of centre drilling must precede an occurrence of drilling
under a complex occurrence of reamed hole making. Other behaviours under
the complex reamed hole making activity may occur in between."
```

Expression 8-2 Example of an IC for the class “Reamed_Hole_Making”

8.3.2.3 Warnings and Errors in Loading the Machining Hole Feature Ontology A

During the development of the “Machining Hole Feature Ontology A”, few warnings and errors were flagged while loading the KFL file containing the domain ontology in the corresponding Object Management System (OMS). These occurred during the parsing phase of the KFL file. Figure 8-7 depicts (1) warnings as a result of confusing variables declared in some axioms and (2) an error which occurred due to an incorrect use of the foundation unary relation “base”. These warnings and errors have prompted the necessary rectifications prior to a successful loading and saving of the KFL file for “Machining Hole Feature Ontology A”.

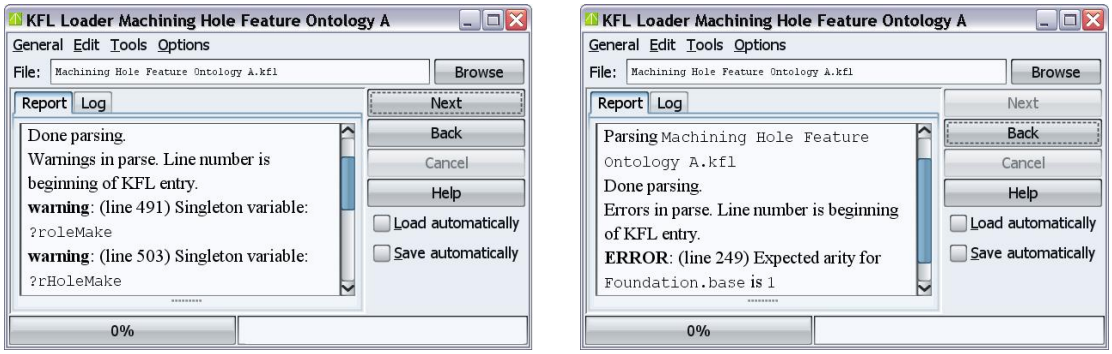


Figure 8-7 Warnings and Errors during the Loading Process of "Machining Hole Feature Ontology A" in its OMS

8.3.2.4 Instantiating Entity Information Concepts

The "Machining Hole Feature Ontology A" provides a domain model that allows the semantic representation of discrete knowledge through instantiation. Instances based on the domain ontology are populated according to the KB schema defined within the "Machining Hole Feature Ontology A". In this test case a concrete state of the entity information concepts of the ontology has been captured as shown in Figure 8-8.

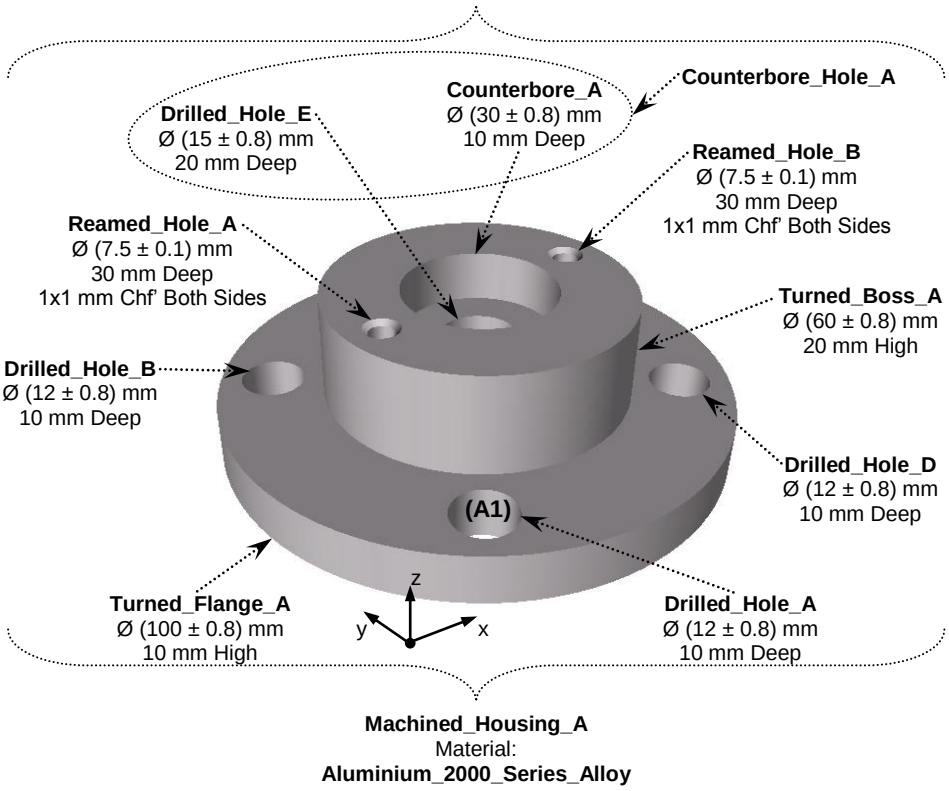


Figure 8-8 Populated Entity Information Instances for Discrete Knowledge Representation

Individuals of the various classes of features are identified and these carry geometry and dimensional semantics as shown in the figure, for example, “Drilled_Hole_A” **(A1)** is an instance of the class “Drilled_Hole” (see Figure 8-3 **(J)**) and has a diameter that measures 12 mm, a diameter tolerance of +/- 0.8 mm and has a depth that measures 10 mm. Note that another instance “Drilled_Hole_C” of “Drilled_Hole” has also been defined but has not been shown in the diagram as it is hidden. The “Drilled_Hole_C” follows the same dimensional parameters as “Drilled_Hole_A” but has a different placement. All the specified instances that pertain to this domain ontology have been defined within the “machiningHoleFeatureOntologyA” context, which is the created namespace for the “Machining Hole Feature Ontology A”.

8.3.2.5 Identifying Incorrect and Missing Entity Information Knowledge

Figure 8-9 illustrates the process of loading facts and fact sentences into the KB linked to “Machining Hole Feature Ontology A” in IODE. The “Asserter” button **(B1)** is used to invoke the “Asserter” pane **(C1)**. Required facts and fact sentences are copied and pasted from the appropriate SCL file containing the instances into the load window **(D1)** of the “Asserter”.

It is of extreme importance to check the “Check ICs?” field **(E1)** prior to loading the SCL file, as this process is detrimental to saving instances in such a way that these follow the consistency of the ICs from the Foundation Layer and the “Machining Hole Feature Ontology A”. In this way any violated ICs are reported, thereby prompting the knowledge engineer to perform the necessary modifications to rectify incorrect and/or missing semantics in the SCL file containing the facts. In the first attempt to load and save entity information knowledge, two hard IC violations and three soft IC violations have been reported as shown in Figure 8-9 **(F1)**. The source of the infringements appear at the end of each listed violated IC (not shown in Figure 8-9 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” **(G1)**, the first loading attempt is rejected. On consulting the appropriate SCL file, it is discovered that missing information is in fact present in the specification of the “Counterbore_Hole_A” instance (see Figure 8-8), where “Drilled_Hole_E” and “Counterbore_A” have not been aggregated under the compound feature “Counterbore_Hole_A”. Note also from Figure 8-9 the soft IC violation “Every core entity holds some function” **(H1)**, 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. The consequence of the soft IC is not detrimental to the integrity of facts being populated under “Machining Hole Feature Ontology A”.

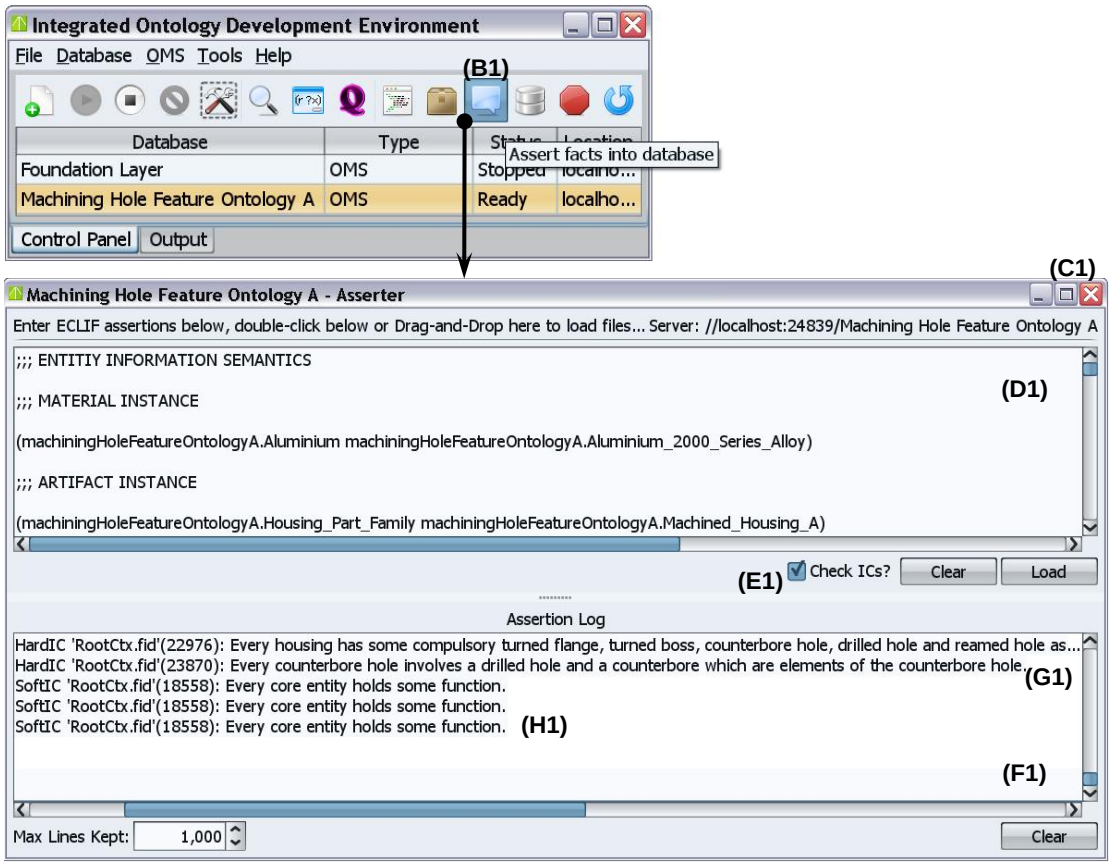


Figure 8-9 Loading Entity Information Instances in the KB of "Machining Hole Feature Ontology A" Using the Asserter Tool in IODE

Facts with hard IC violations are rectified accordingly, reloaded and checked for IC violations again, and saved in the KB. Figure 8-10 shows the “Counterbore_Hole_A” instance **(I1)** that has been successfully created and can be browsed from the “Instances” tab **(J1)** for the class “Counterbore_Hole” **(I)**. In the test case, all entity information instances have

been successfully created, with consideration made to the list of IC violations reported in Figure 8-9 **(F1)**.

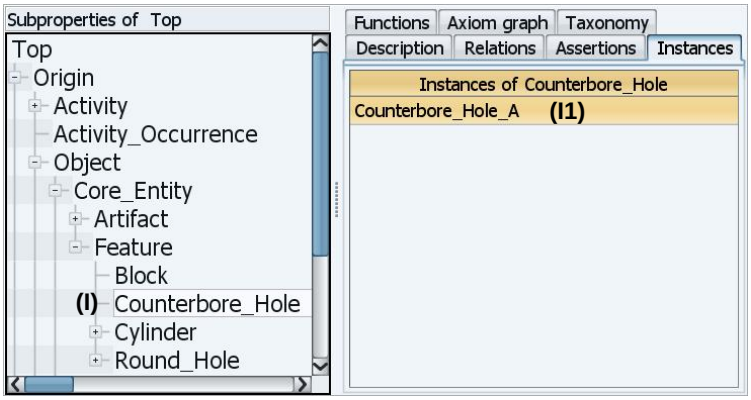


Figure 8-10 Example of a Successfully Created Instance of the Class "Counterbore_Hole"

8.3.2.6 Instantiating Machining Process Concepts

Based on the “Machining Hole Feature Ontology A”, a concrete state of the machining process concepts, and the relationships between process and entity information instances, have also been captured as shown in Figure 8-11. The figure depicts a complex instance of the foundation class “Activity Occurrence”, “occ_Make_Counterbore_Hole_A” **(K1)**, from which the compound feature “Counterbore_Hole_A” is output **(O1)**. Notice that the occurrence “occA_Hole_Centre_Drilling” **(M1)** is not only at the root of the machining process sequence “occ_Make_Counterbore_Hole_A” **(K1)**, but is also positioned as the initial activity occurrence in the occurrence tree, and therefore precedes all other occurrences in the tree. The dotted arrows **(N1)** in the diagram capture the linear ordering semantics over the various occurrences. These linear ordering semantics are built based on the PSL Core Theory, the Theory of Subactivities, Occurrence Trees, Complex Activities and Activity Occurrences coming from the PSL Outer-Core.

A complex occurrence “occ_Make_Reamed_Holes_AB” **(L1)** has also been specified from which “Reamed_Hole_A” and “Reamed_Hole_B” are output **(O1)**. Recall from Figure 8-6 the linear ordering semantics involved in the specification of occurrences of the “Activity” class “Reamed_Hole_Making”

from which “occ_Make_Reamed_Holes_AB” **(L1)** is an occurrence. Following these linear ordering semantics, the occurrences appearing under “occ_Make_Reamed_Holes_AB”, shown in Figure 8-11, are completely legal and allowable. This is because, for example, as long as all occurrences of “Centre_Drilling” are happening before occurrences of “Drilling”, which in turn are happening before all occurrences of “Reaming”, then the complex occurrence “occ_Make_Reamed_Holes_AB” can take place.

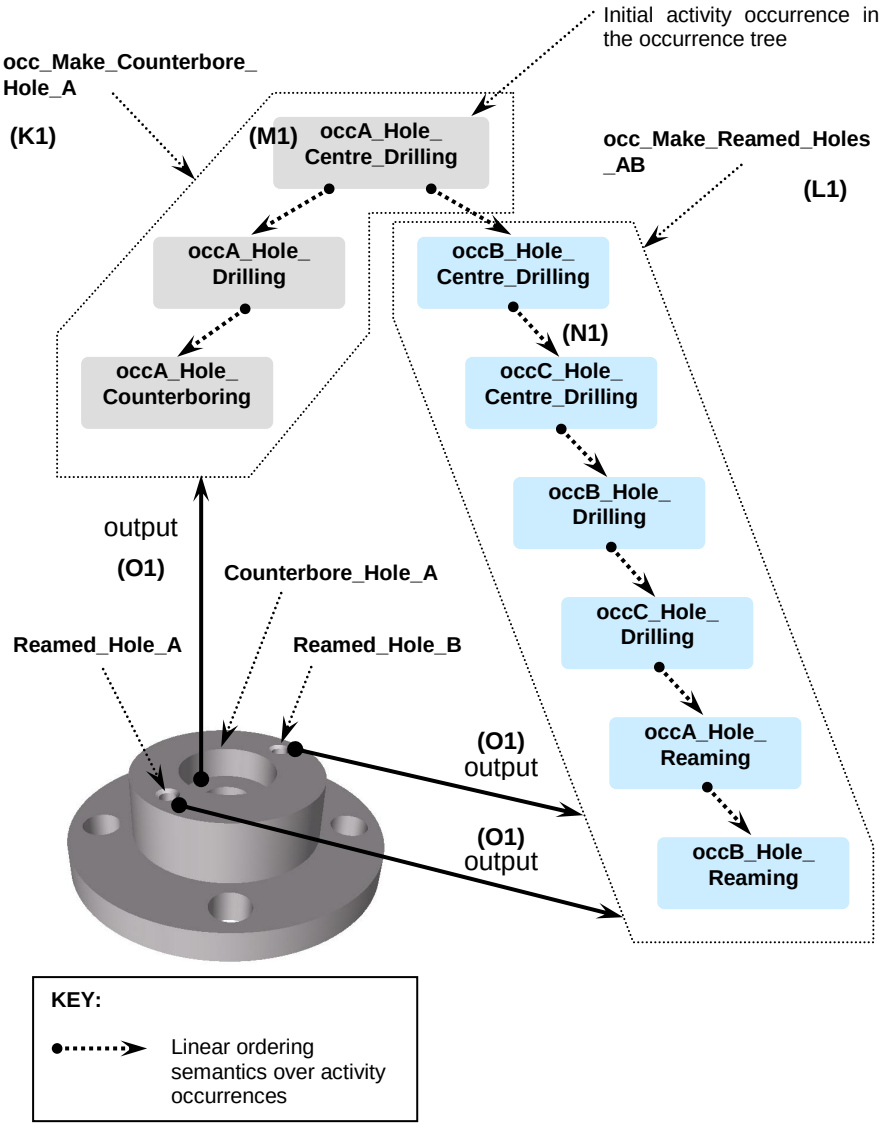


Figure 8-11 Populating Process Instances and Creating Relationships between Entity Information and Process Instances for Discrete Knowledge Representation

8.3.2.7 Identifying Incorrect and Missing Process Knowledge

Facts and fact sentences that contain the semantics expressed in Figure 8-11 are loaded into the KB of “Machining Hole Feature Ontology A”. During the loading process, a number of violations of ICs have been reported. This is illustrated in Figure 8-12. Five soft ICs **(P1)** are present and, therefore, do not constitute a problem to committing the loaded instances into the KB. However, the soft ICs being flagged 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 add extra semantics to the represented discrete knowledge.

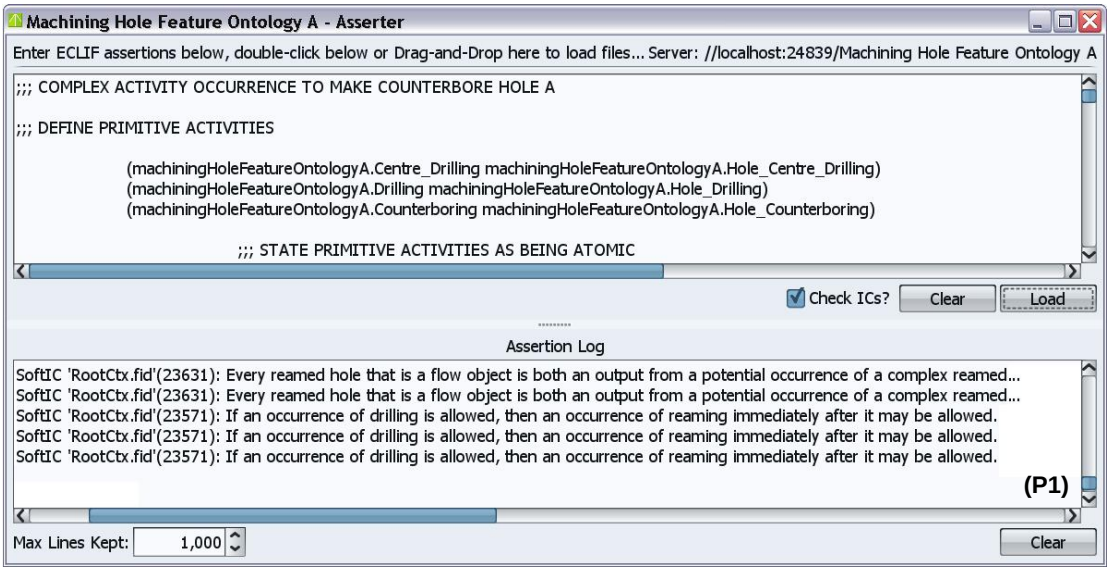


Figure 8-12 Loading Process Instances in the KB of “Machining Hole Feature A” Using the Asserter Tool in IODE

Figure 8-13 then portrays all the successfully created instances of the class “Activity_Occurrence” **(Q1)** relevant to the “Machining Hole Feature Ontology A”. The complex occurrence “occ_Make_Reamed_Holes_AB” **(R1)** has been highlighted. Reviewing Figure 8-11 with Figure 8-13 reveals that all the subactivity occurrences of complex occurrences have also been asserted.

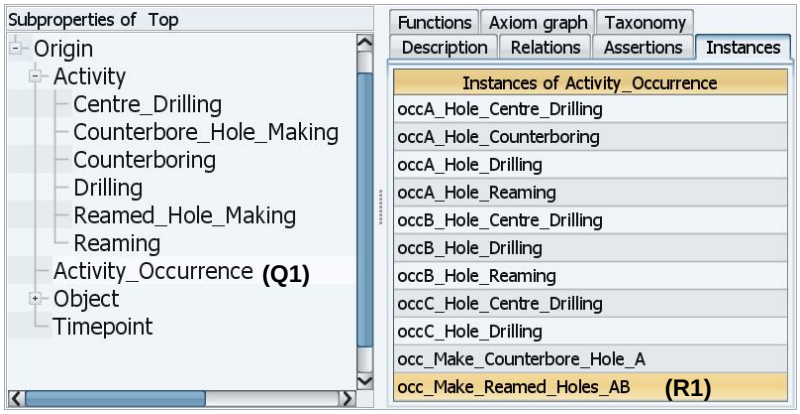


Figure 8-13 Example of Successfully Created Instances of the class "Activity_Occurrence"

8.3.3 Discussions and Validation of Results

Test Case 1 has demonstrated how the integrity-driven specialisation of a domain model, “Machining Hole Feature Ontology A”, can be achieved using the supported semantic structures present in the Foundation Layer of SMIF. The competency questions featured in section 8.3.1 are next reviewed.

- Can the ontological mechanisms that allow specialisation to occur, be used during the development of “Machining Hole Feature Ontology A”?

The simplest of the ontological mechanisms that allows specialisation to occur is concerned with the statement of a context (namespace) for the “Machining Hole Feature Ontology A”. In the test case, this context has been named “machiningHoleFeatureOntologyA”. Secondly, the primary type of ontological relationship used to specialise the concepts from the Foundation Layer to appropriate concepts in the domain ontology has been specified through super/sub-class relationships. Instance-of relationships have been employed whenever facts have been populated.

Since the controlled specialisation approach is under consideration, this indicates that the domain ontology has not defined new relations but instead reused the relations already present in the heavyweight manufacturing ontological foundation. Overall, Test Case 1 provides a solid confirmation that

the ontological mechanisms that allow specialisation to take place can fully be exploited during domain model specialisation.

- Can the specification of domain-defined integrity constraints be achieved in a flexible way while not violating foundation semantics?

A number of integrity constraints of varied formal meaning, relevant to the domain ontology, has been documented in this section. These ICs have been defined in order to capture the constraints pertinent to the semantics of “Machining Hole Feature Ontology A”. It is evident that the approach fosters the desired level of flexibility in the specialisation of domain-defined ICs. Perceivable mistakes and inconsistencies in the logical theory of the domain ontology have been reported during the KFL file loading phase. However, it is worth pointing out that IODE is an ontological environment as opposed to a theorem prover, and therefore the underlying computational principle of IODE differs slightly from theorem provers.

- Is it possible to accurately represent discrete knowledge through instantiation, based on the semantics captured in the “Machining Hole Feature Ontology A” and foundation semantics?

Instance files written in SCL contain the necessary semantics for the representation of discrete knowledge to be loaded in the KB of a domain model. In Test Case 1, a number of facts and fact sentences have been populated in the KB of “Machining Hole Feature Ontology A”. During the assertion process of these instances, a number of IC violations have been reported and rectified. Therefore it is possible, through the specification of correctly structured and rectified SCL instance files, based on domain and foundation semantics, to support the accurate representation of discrete knowledge.

8.4 Test Case 2: Reconciliation Using Semantic Mapping Concepts Based on Foundation Semantics

8.4.1 Aim and Objectives

The aim of Test Case 2 is to provide a proof of concept for the reconciliation between two inter-system domains. The mode in which reconciliation is to take place is through the use of semantic mapping concepts based on foundation semantics. Two domain models are under consideration namely (1) “Machining Hole Feature Ontology A” and (2) another hole feature ontology identified as “Machining Hole Feature Ontology B”. The following competency questions apply to Test Case 2:

- Is it possible to exploit the semantic mapping concepts based on foundation semantics to evaluate and verify the correspondences at the class level between “Machining Hole Feature Ontology A” and “Machining Hole Feature Ontology B”?
- Is it possible to exploit the semantic mapping concepts based on foundation semantics to evaluate and verify the correspondences at the function level between “Machining Hole Feature Ontology A” and “Machining Hole Feature Ontology B”?
- Is it possible to exploit the semantic mapping concepts based on foundation semantics to evaluate and verify the correspondences at the instance level between “Machining Hole Feature Ontology A” and “Machining Hole Feature Ontology B”?

There are three main objectives involved namely (1) the deployment of the Semantic Reconciliation and Interoperability Evaluation layers for reconciling between the two domain models, (2) the deployment of semantic mapping concepts based on foundation semantics and (3) the use of the relevant set of tools identified in Chapter 7 notably IODE, the Interoperability Evaluation Assistant and the query tool in IODE for evaluating and verifying cross-domain correspondences.

8.4.2 Machining Hole Feature Ontology B

The ontology development process for “Machining Hole Feature Ontology B” follows a similar approach to that of “Machining Hole Feature Ontology A”. All the concepts and ICs developed in this domain ontology can be browsed from Appendix D.3. The diagram in Figure 8-14 illustrates the types of entity information and process concepts being investigated in “Machining Hole Feature Ontology B” pertaining to a “Crank_Pulley_Part_Family”. A number of classes of features form the basis for the representation of entity information semantics, for example, the class “Pulley_Core_Feature” **(S1)** identifies a category of feature of compound property expressed in terms of “Bored_Hole” **(T1)**, “Large_Bored_Hole” **(U1)** and the foundation class “Cylinder”.

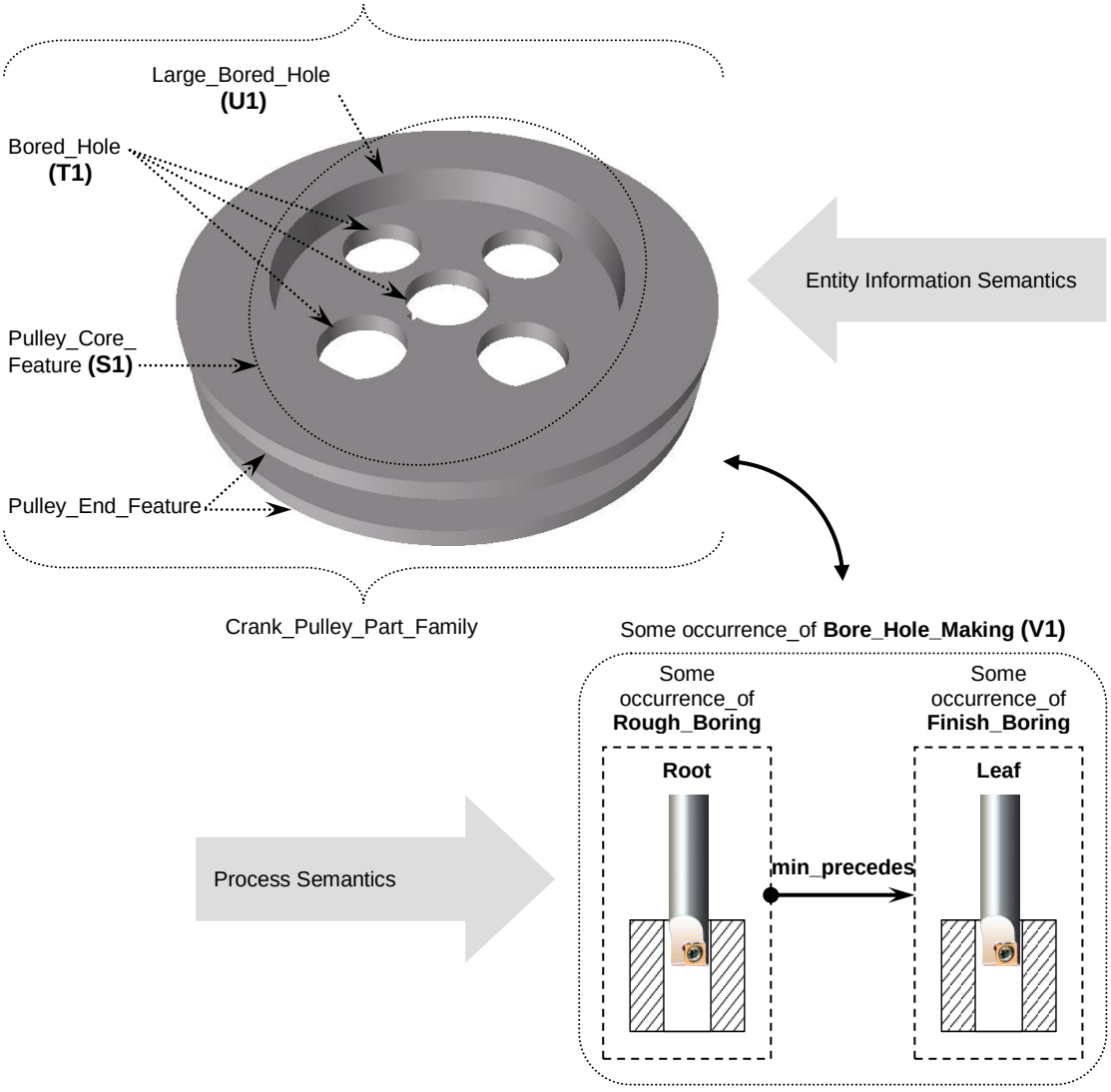


Figure 8-14 Example of Entity Information and Process Semantics Developed in "Machining Hole Feature Ontology B"

Process concepts are also present in order to further capture the machining process planning viewpoint of the domain ontology. Figure 8-14 illustrates an example featuring “Bore_Hole_Making” **(V1)**, a sub-class of the foundation class “Activity”, which captures the semantics of hole boring operations for the types of hole features represented in the ontology.

Based on the entity information and process concepts explored in “Machining Hole Feature Ontology B”, a concrete state of the ontology is then presented in Figure 8-15. The figure identifies the instantiation of concepts from Figure 8-14 used for representing and populating discrete knowledge in the KB of “Machining Hole Feature Ontology B”. Entity information instances have been specified such as “Crank_Pulley_Series_01” **(W1)** and “Pulley_Core_A” **(X1)**. Note that the instances “Cylinder_A” **(Y1)** and “Large_Bored_Hole_A” **(Z1)** are hidden features which cannot be directly labelled in the figure but have been defined in order to obtain a full representation of “Crank_Pulley_Series_01”.

Figure 8-15 also depicts a defined branch of the occurrence tree containing the linear ordering semantics over atomic “Activity_Occurrence” instances, used for capturing the ordering semantics within an execution of the complex occurrence “occ_Make_Bored_Holes_ABCDE” **(A2)**, from which the hole features “Bored_Hole_A” to “Bored_Hole_E” are output. Under the formalised version of the process semantics expressed in Figure 8-14 (see Appendix D.3), such a process execution sequence as shown in Figure 8-15 is completely legal according to the domain-defined process semantics.

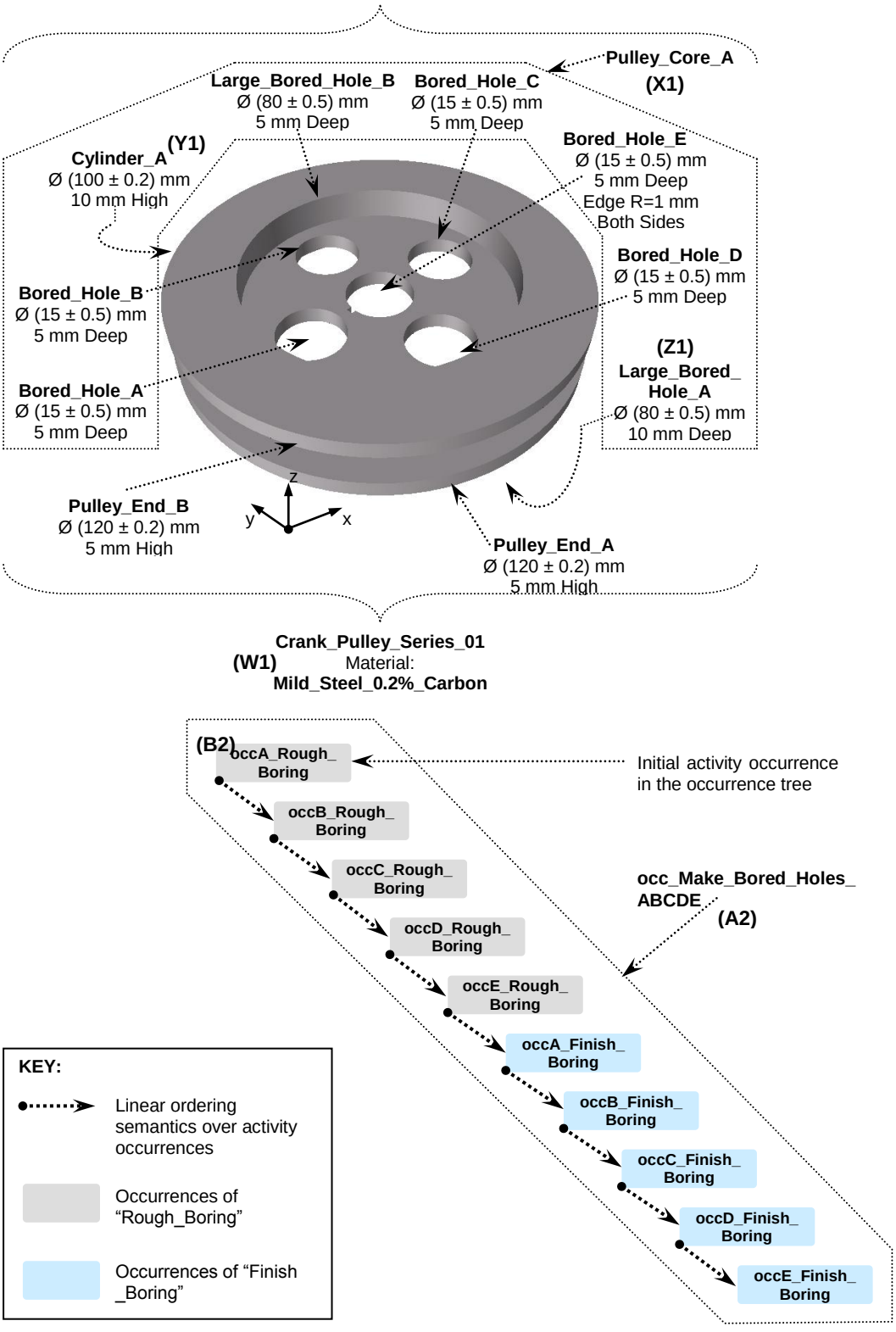


Figure 8-15 Populated Entity Information and Process Instances for Discrete Knowledge Representation in the KB of “Machining Hole Feature Ontology B”

8.4.3 Reconciliation Scenarios

A number of reconciliation scenarios are to be proved in this test case. These scenarios are based on the reconciliation of “Machining Hole Feature Ontology A” with “Machining Hole Feature Ontology B”. The class, function and instance levels of both models are under consideration, and where appropriate, reconciliation is to be shown for entity information as well as process semantics between both domain models.

8.4.3.1 Reconciliation at the Class Level

Figure 8-16 illustrates the concepts to be reconciled at the class level. These involve the discovery of correspondences between the entity information classes “Reamed_Hole” and “Bored_Hole” **(B2)**, and the process classes “Reaming” and “Finish_Boring” **(C2)**.

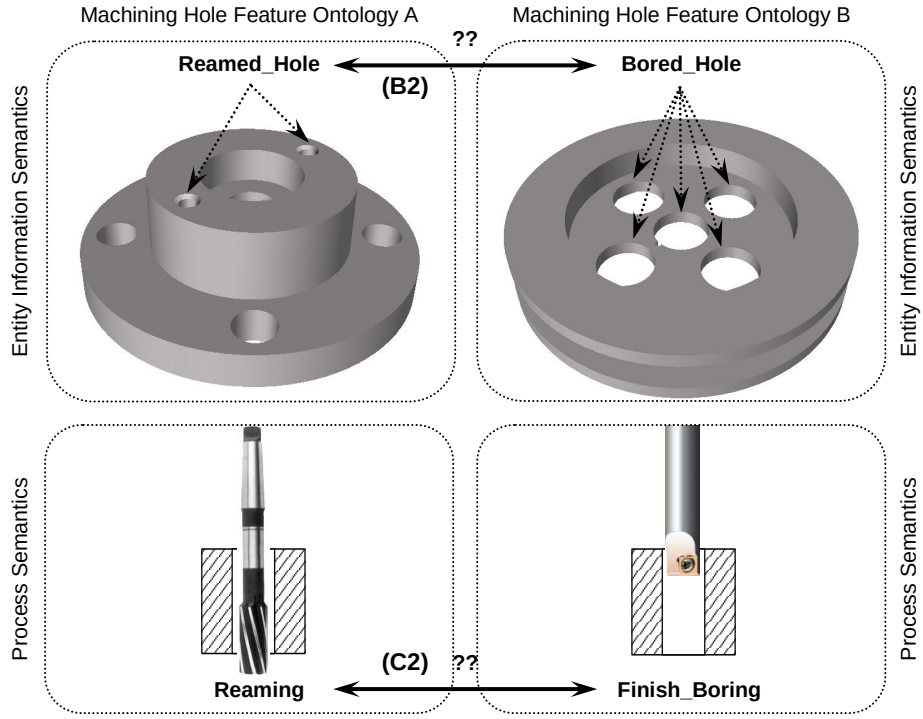


Figure 8-16 Reconciliation Scenario at the Class Level

8.4.3.2 Reconciliation at the Function Level

Two unary domain-defined ontological functions are to be reconciled as shown in Figure 8-17. The correspondences that hold between the functions “inch” and “inches” **(D2)** are to be discovered.

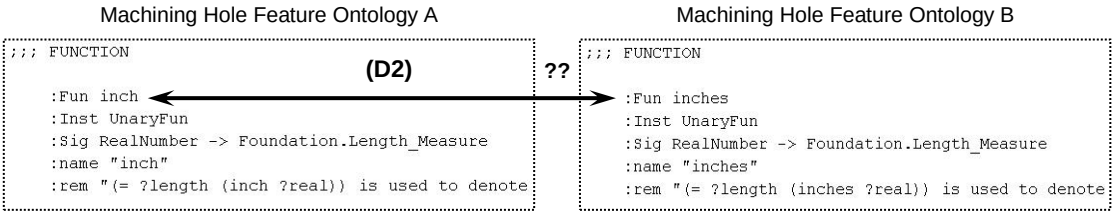


Figure 8-17 Reconciliation Scenario at the Function Level

8.4.3.3 Reconciliation at the Instance Level

Figure 8-18 depicts the individuals to be reconciled at the instance level. Correspondences need to be discovered between the entity information instances “Reamed_Hole_A” and “Bored_Hole_E” **(E2)**, and the compound features “Counterbore_Hole_A” and “Pulley_Core_A” **(F2)**.

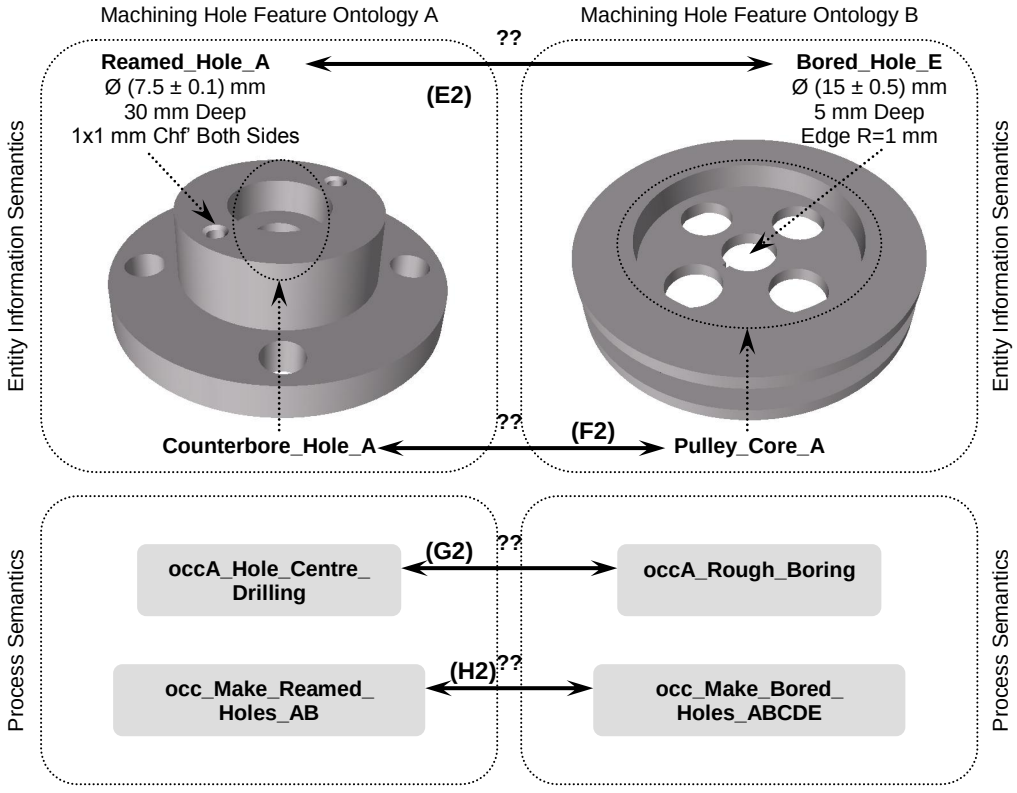


Figure 8-18 Reconciliation Scenario at the Instance Level

Other correspondences are to be discovered between the primitive activity occurrences “occA_Hole_Centre_Drilling” and “occA_Rough_Boring” **(G2)**, and the complex activity occurrences “occ_Make_Reamed_Holes_AB” and “occ_Make_Bored_Holes_ABCDE” **(H2)**.

8.4.4 Ontology Mapping Process

The deployment of the Semantic Reconciliation Layer necessitates the application of the ontology mapping process concepts explained in Chapter 6. Figure 8-19 demonstrates the order in which the ontology mapping process concepts are executed, while picturing the result of each stage during implementation. The context “machiningHoleFeatureOntologyA” defined for the “Machining Hole Feature Ontology A” is first adjusted to “DomainX” **(I2)** in the KFL file for the domain ontology. The same step is carried out for the “Machining Hole Feature Ontology B” whose context is adjusted to “DomainY”. Instance files pertaining to both domain models also have their contexts adjusted to “DomainX” and “DomainY” respectively.

The OMS for the “Foundation Layer” **(J2)** is cloned and renamed accordingly **(K2)**. The context-adjusted “Machining Hole Feature Ontology A” is loaded and saved **(L2)** into the new OMS named “Test Case 2”. “Machining Hole Feature Ontology B” is loaded and saved in the same OMS resulting in the merged ontologies. Instance files are merged into the KB of “Test Case 2”. Figure 8-19 **(M2)** identifies the taxonomy of merged domain-defined classes pertaining to both domain ontologies and carrying their adjusted contexts, for example, “DomainX.Reamed_Hole” and “DomainY.Bored_Hole”. Following the simple merging stage, the KFL file for semantic mapping concepts based on foundation semantics (see Appendix E.1 if needed) is loaded and saved **(N2)** in “Test Case 2”.

During the latter process, semantic alignments are automatically assigned to the appropriate cross-domain arguments, but remain invisible to the user. The discovery of applicable correspondences between cross-domain arguments is performed by deploying the Interoperability Evaluation Layer.

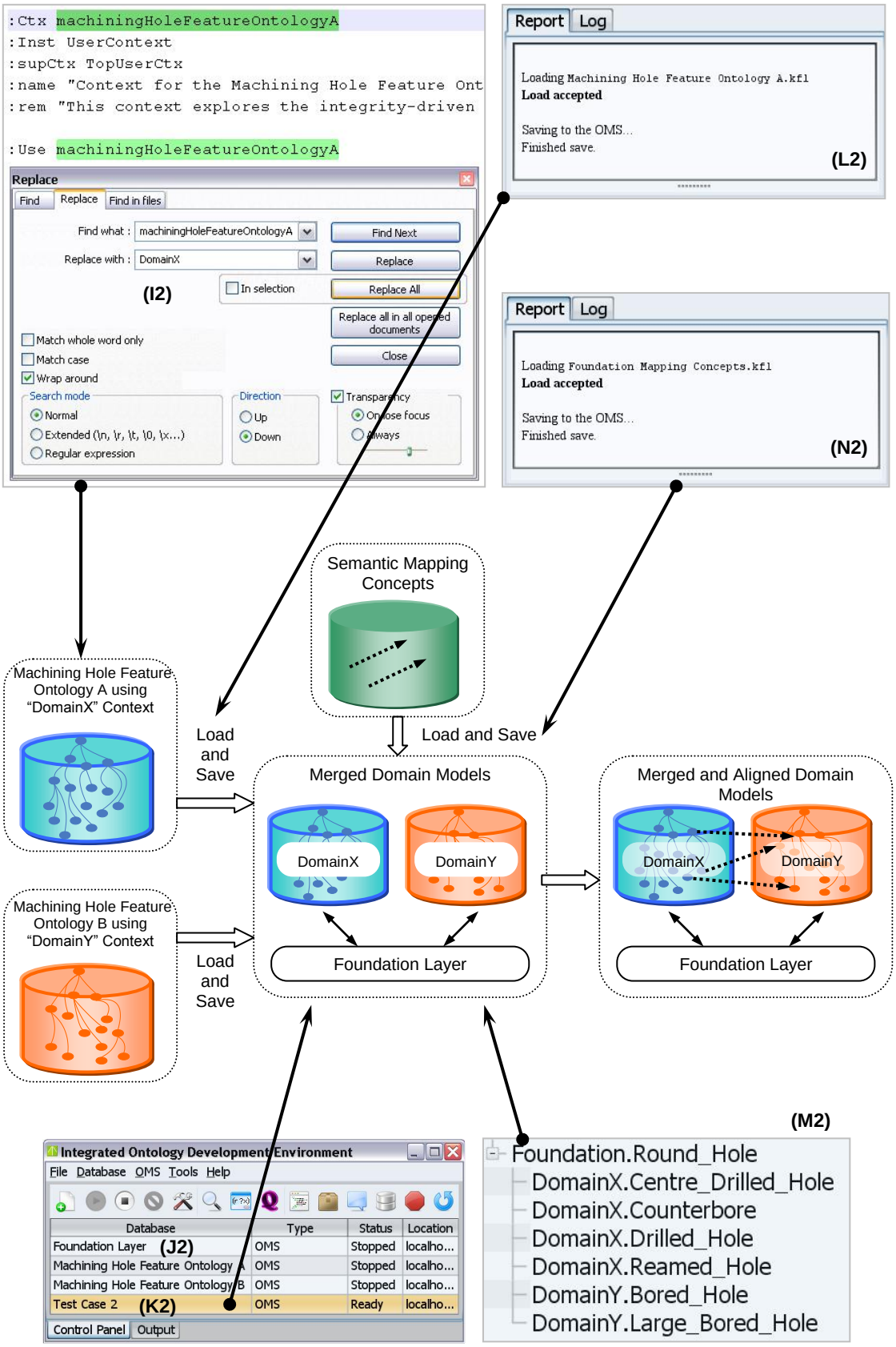


Figure 8-19 Deploying Ontology Mapping Concepts in the Semantic Reconciliation Layer

8.4.5 Interoperability Evaluation and Verification

In the Interoperability Evaluation Layer, two main tools are deployed namely the web-based Interoperability Evaluation Assistant interface and the query tool in IODE. Note that in this test case only the Java-based modules of the Interoperability Evaluation Assistant have been used in order to build queries for retrieving all semantic mapping concepts between known cross-domain arguments in a single transaction (see section 7.3.4.1 in Chapter 7). This is used particularly for a quicker approach to analysing query results.

8.4.5.1 Discovery of Semantic Mapping Concepts at the Class Level

"Reamed_Hole" v/s "Bored_Hole"

The home page for the Interoperability Evaluation Assistant is invoked as shown in Figure 8-20. Using the Java-based module for building queries involving semantic mapping concepts based on foundation semantics, the names of the classes are typed in the relevant text field **(O2)**. On clicking the submit button, the query is retrieved **(P2)**, copied and pasted in the query tool in IODE as illustrated in Figure 8-21 **(Q2)**.

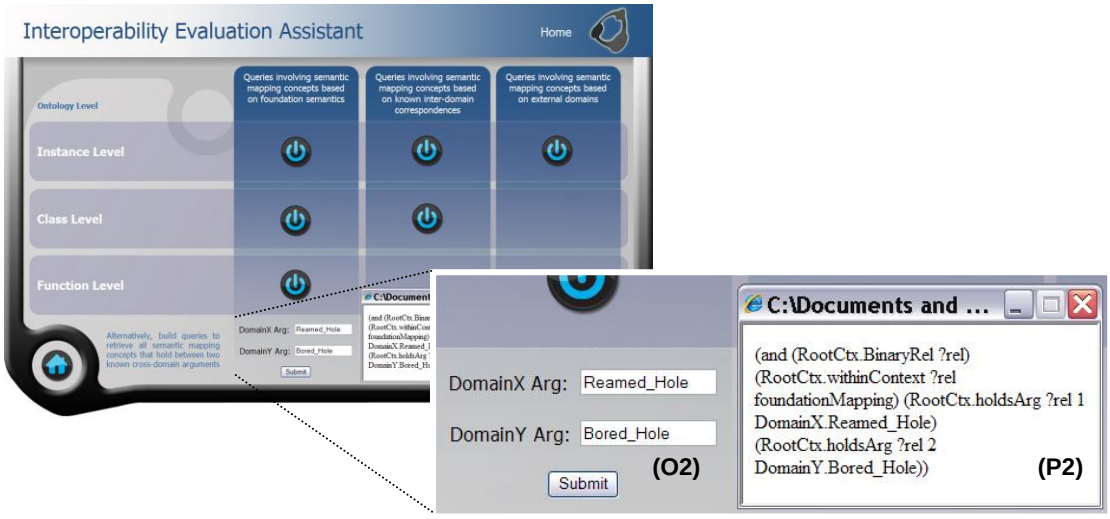


Figure 8-20 Using the Interoperability Evaluation Assistant to Build a Query

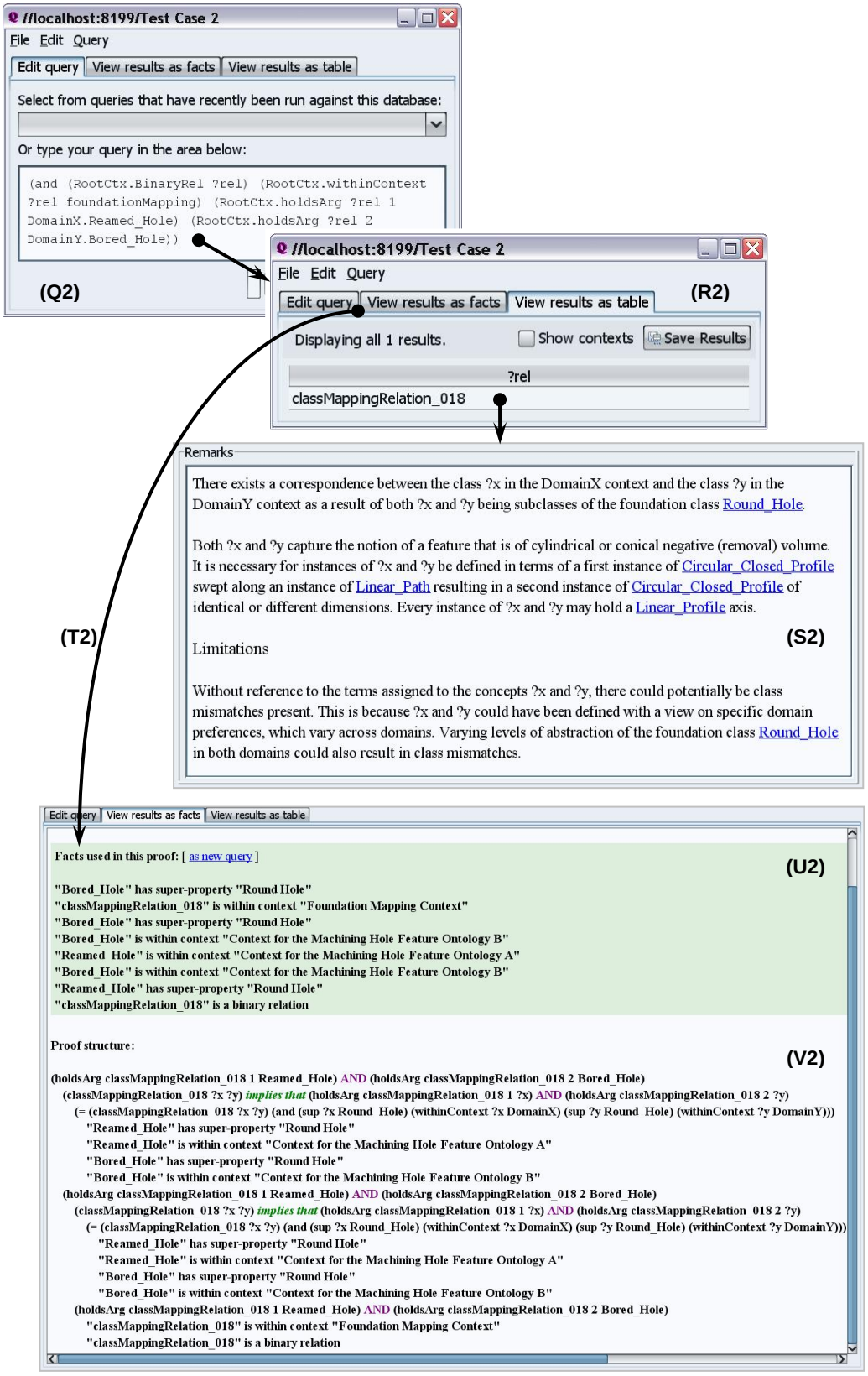


Figure 8-21 Running a Query, Viewing Results and Verifying Semantic Mapping Concepts between the Classes “Reamed_Hole” and “Bored_Hole”

The query is run and appropriate results are tabulated **(R2)**. In this case, there is only one correspondence “classMappingRelation_018” that has been aligned between the class “Reamed_Hole” from “Machining Hole Feature Ontology A” and “Bored_Hole” from “Machining Hole Feature Ontology B”. The semantic mapping concept can be browsed to view the nature of the interoperability **(S2)**. The remarks available during browsing provide a view on the commonalities as well as the possible uncertainties that may exist between the two classes. The result can further be viewed as a fact and proved **(T2)** using logic. During this verification stage, an informal interpretation of the verification is provided **(U2)** alongside the more formal proof structure **(V2)**.

“Reaming” v/s “Finish_Boring”

A similar procedure is applied for finding the semantic mapping concepts between the classes “Reaming” and “Finish_Boring” (see Figure 8-16). Figure 8-22 shows a single result, “classMappingRelation_022”, obtained. Browsing the semantic mapping concept reveals the nature of the commonalities and possible differences between both cross-domain classes. The verification stage resembles the one shown previously and has not been indicated here.

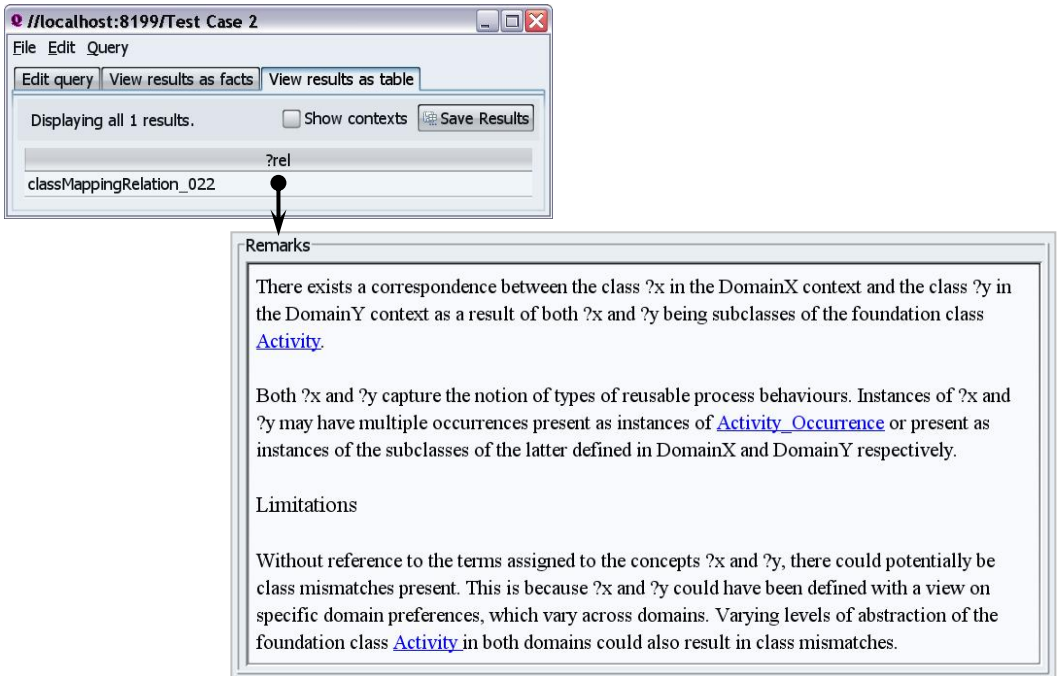


Figure 8-22 Browsing the Semantic Mapping Concept between the Classes "Reaming" and "Finish_Boring"

8.4.5.2 Discovery of Semantic Mapping Concepts at the Function Level

“inch” v/s “inches”

The names of the ontological functions to be reconciled are typed into the required text fields from the Interoperability Evaluation Assistant. The built query is pasted into the query tool in IODE and executed as shown in Figure 8-23.

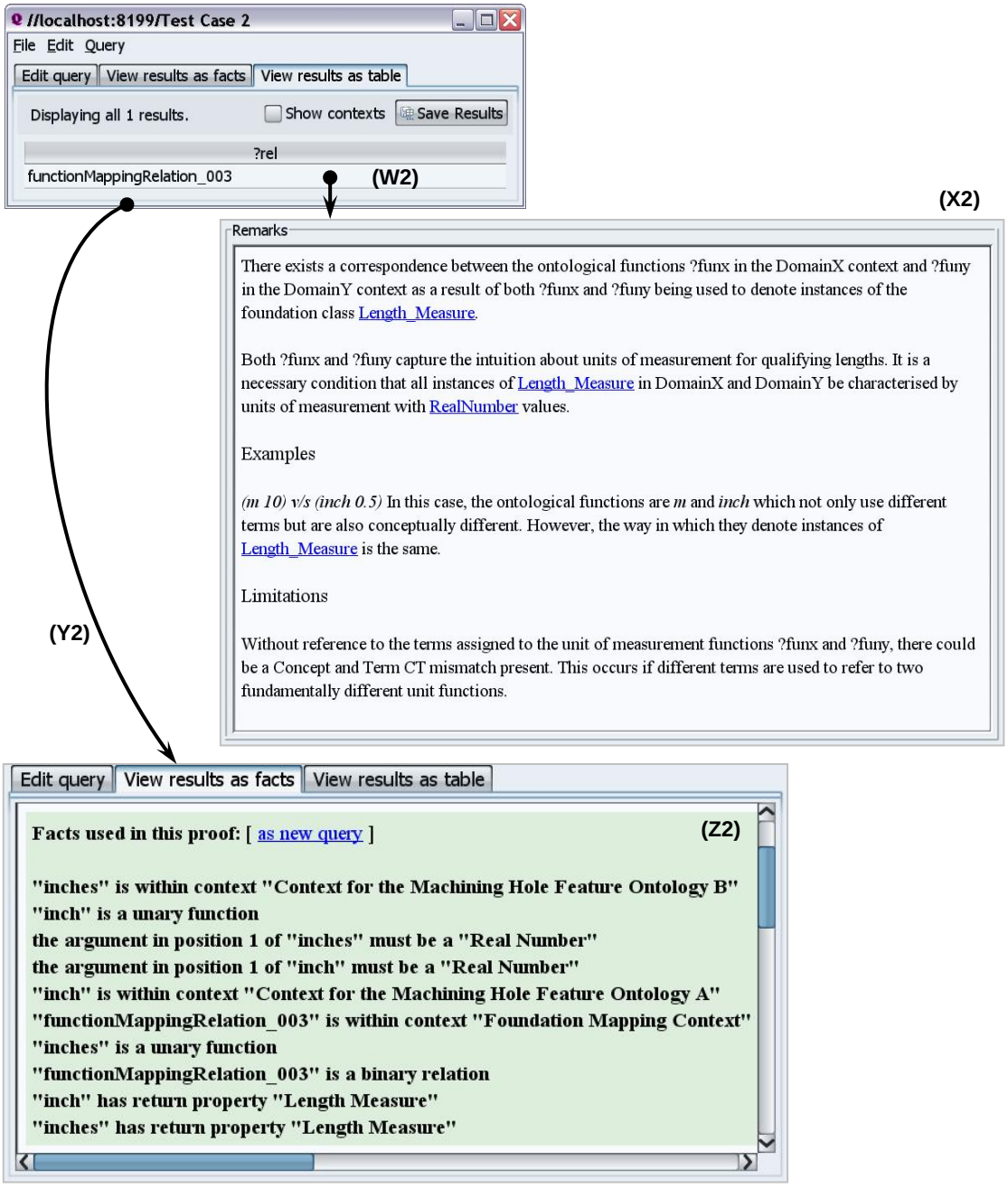


Figure 8-23 Browsing and Verifying the Semantic Mapping Concept between the Ontological Functions "inch" and "inches"

On running the query, one result called “functionMappingRelation_003” **(W2)** is retained. The browsed result **(X2)** informally identifies the correspondences between “inch” in “Machining Hole Feature Ontology A” and “inches” in “Machining Hole Feature Ontology B”. The result can also be viewed as a fact and proved **(Y2)** in order to verify the reason behind the alignment of the semantic mapping concept “functionMappingRelation_003” to “inch” and “inches”. Part of the proof, which is informally expressed, **(Z2)** is also identified in Figure 8-23.

8.4.5.3 Discovery of Semantic Mapping Concepts at the Instance Level

“Reamed_Hole_A” v/s “Bored_Hole_E”

A number of query results are obtained while interrogating the semantic mapping concepts that hold between the two entity information instances “Reamed_Hole_A” and “Bored_Hole_E” (see Figure 8-24).

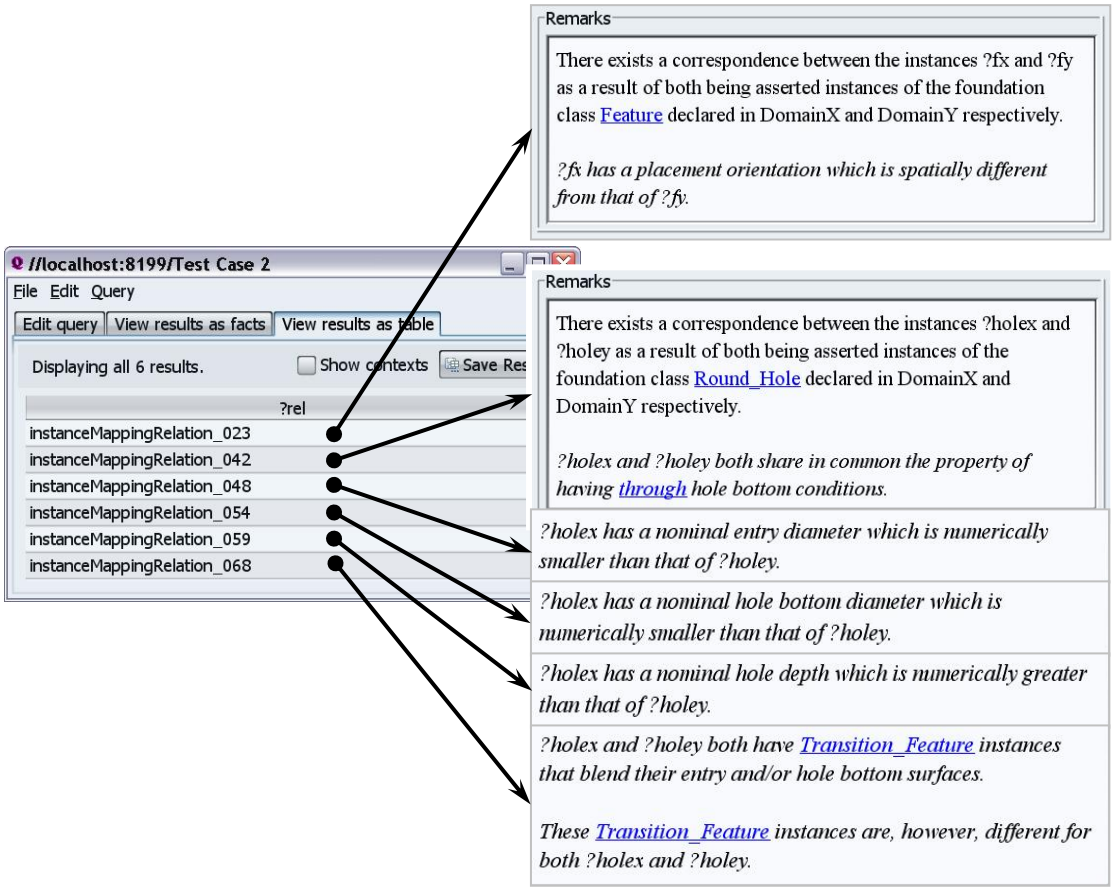


Figure 8-24 Viewing and Browsing Semantic Mapping Concepts between the Instances "Reamed_Hole_A" and "Bored_Hole_E"

These results represent the commonalities and differences that occur between the geometric and dimensional semantics carried by these instances. The verification and proof structure stage has not been shown here due to the lengthy proof structures present. However, in any situation where semantic mapping concepts are discovered to the held between two cross-domain arguments, it is always possible to verify them.

“Counterbore_Hole_A” v/s “Pulley_Core_A”

Two feature instances of compound property namely “Counterbore_Hole_A” from “Machining Hole Feature Ontology A” and “Pulley_Core_A” from “Machining Hole Feature Ontology B” have been compared. Figure 8-25 illustrates the established semantic mapping concepts.

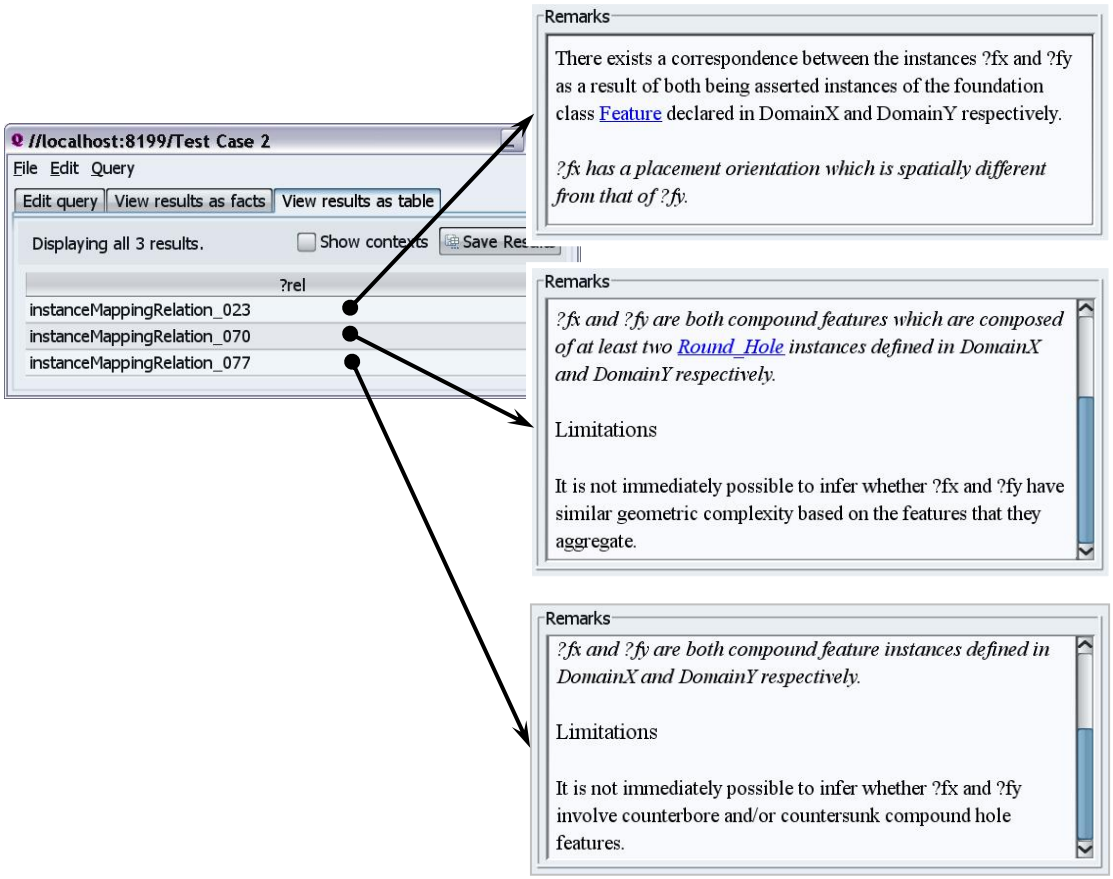


Figure 8-25 Viewing and Browsing Semantic Mapping Concepts between the Instances “Counterbore_Hole_A” and “Pulley_Core_A”

“occA_Hole_Centre_Drilling” v/s “occA_Rough_Boring”

Figure 8-26 depicts the established semantic mapping concepts between two atomic activity occurrences which form part of distinct branches of the occurrence tree in both domains.

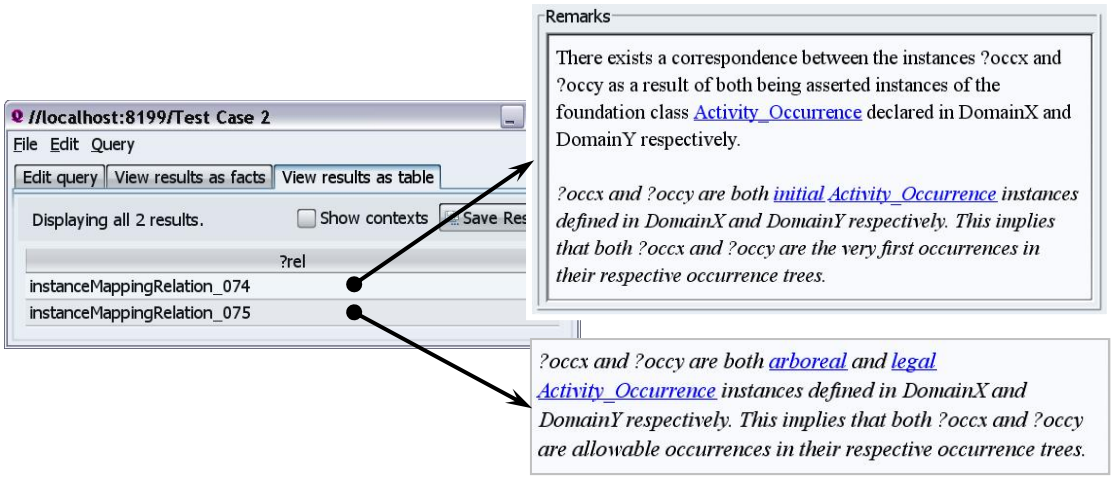


Figure 8-26 Viewing and Browsing Semantic Mapping Concepts between the Instances "occA_Hole_Centre_Drilling" and "occA_Rough_Boring"

“occ_Make_Reamed_Holes_AB” v/s “occ_Make_Bored_Holes_ABCDE”

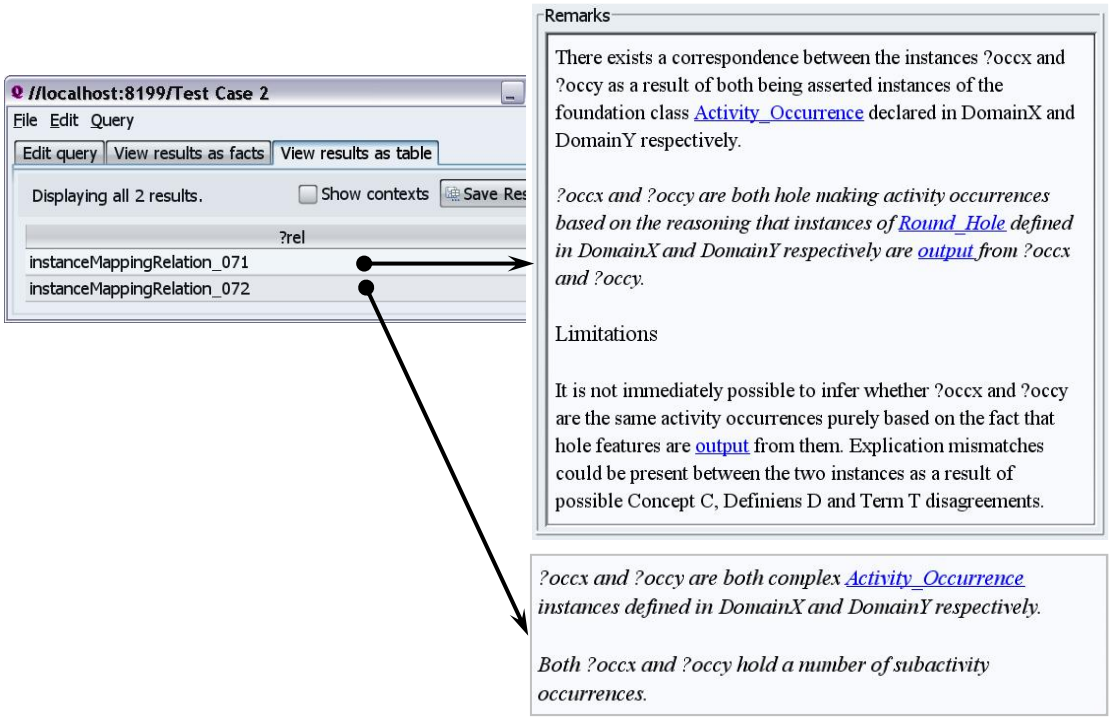


Figure 8-27 Viewing and Browsing Semantic Mapping Concepts between the Instances “occ_Make_Reamed_Holes_AB” and “occ_Make_Bored_Holes_ ABCDE”

Figure 8-27 illustrates the results of querying for all the semantic mapping concepts that hold between the instances “occ_Make_Reamed_Holes_AB” and “occ_Make_Bored_Holes_ABCDE”.

8.4.6 Discussions and Validation of Results

Test Case 2 has demonstrated that it is possible, following the SMIF approach, to exploit semantic mapping concepts based on foundation semantics. These semantic mapping concepts have been used to evaluate and verify the correspondences that hold between cross-domain arguments that could be present at the class, function and instance levels of domain models. Based on the test case results, the competency questions identified in section 8.4.1 have been answered positively.

Carefully selected pairs of cross-domain arguments have been chosen in order to illustrate the applicability of semantic mapping concepts for the reconciliation of inter-domain semantics. However, in reality, any suitable pairs of cross-domain arguments could be identified and queried for semantic mapping concepts. Results would be obtained in the event that semantic mapping concepts can be logically verified between the pairs of arguments. It is to be noted that the verification stage is heavily logic-dependent and necessitates a good user knowledge of Common Logic and KFL in order to interpret the proof structures. However, it is seen that the verification stage is more appropriate to the system as it is through this process that the results for queries can be interpreted.

8.5 Test Case 3: Reconciliation Using Semantic Mapping Concepts Based on an External Domain

8.5.1 Aim and Objectives

The aim of Test Case 3 is to provide a proof of concept for the reconciliation between two inter-system domains. The mode in which reconciliation is to take place is through the use of semantic mapping concepts based on an external domain model (see section 6.2.2.3 in Chapter 6), which has been specialised from the Foundation Layer. In this test case, “Machining Hole Feature Ontology A” and “Machining Hole Feature Ontology B” are again under consideration. The competency question relevant to this test case is listed next:

- Can the knowledge structures contained in an external domain model be employed as semantic mapping concepts in order to evaluate and verify the correspondences between cross-domain arguments?

The objectives of this test case include (1) the deployment of the Semantic Reconciliation and Interoperability Evaluation layers, (2) the deployment of semantic mapping concepts based on the “ISO Tolerance Band Model” (adapted from ISO 286-2, 1988) and (3) the use of IODE, the Interoperability Evaluation Assistant and the query tool in IODE for evaluating and verifying cross-domain correspondences.

8.5.2 ISO Tolerance Band Model as External Domain

The understanding behind the formalisation of the “ISO Tolerance Band Model”, as external domain ontology under construction, has previously been explained in section 6.2.2.3 from Chapter 6. Therefore, this section concentrates on providing a global picture on the domain model and how a logic-based approach has been devised for allowing inferences to be made based on the formalised knowledge contained within the domain model.

The “ISO Tolerance Band Model” is based on foundation semantics. The formalised semantic structures for the model can be accessed in Appendix D.4. The “ISO Tolerance Band Model” can firstly help establish the possible hole making processes that could be used to produce known domain-defined “Round_Hole” instances, based on their nominal diameters and diameter tolerances. Tables 8-1 and 8-2, adapted from the documentation in ISO 286-2, serve as the source of knowledge formalised in the external domain ontology. Nominal hole diameters of up to 50 mm (see Table 8-1) and six common hole machining processes (see Table 8-2) have been considered in the domain model.

Table 8-1 ISO Tolerance Band Table for Nominal Hole Diameters up to 50 mm

	Nominal Diameter of Round Hole (mm)					
over	1	3	6	10	18	30
inc.	3	6	10	18	30	50
IT Grade	**Diameter Tolerance in Microns**					
3	2	2.5	2.5	3	4	4
4	3	4	4	5	6	7
5	4	5	6	8	9	11
6	6	8	9	11	13	16
7	10	12	15	18	21	25
8	14	18	22	27	33	39
9	25	30	36	43	52	62
10	40	48	58	70	84	100
11	60	75	90	110	130	160
12	100	120	150	180	210	250
13	140	180	220	270	330	390
14	250	300	360	430	520	620

Table 8-2 ISO Tolerance Band Process Chart for Six Common Hole Machining Processes

IT Grade	3	4	5	6	7	8	9	10	11	12	13	14
Honing												
Internal grinding												
Internal Broaching												
Reaming												
Boring												
Drilling												

Secondly, using the same knowledge it becomes possible to compare domain-defined hole making processes to standard hole machining processes from the “ISO Tolerance Band Model”. For example, it is possible to infer from the “ISO Tolerance Band Model” whether a certain domain-defined instance of “Round_Hole” satisfies the reaming criteria. If such is the case, and if the “Round_Hole” instance is an output from some “Activity_Occurrence” instance, then this could potentially imply that the “Activity_Occurrence”

instance in question would match the tolerance range capability of a reaming process under the “ISO Tolerance Band Model”, regardless of the name of the occurrence. The screen shot in Figure 8-28 identifies all the inference relations present and the remarks associated to two inference relations.

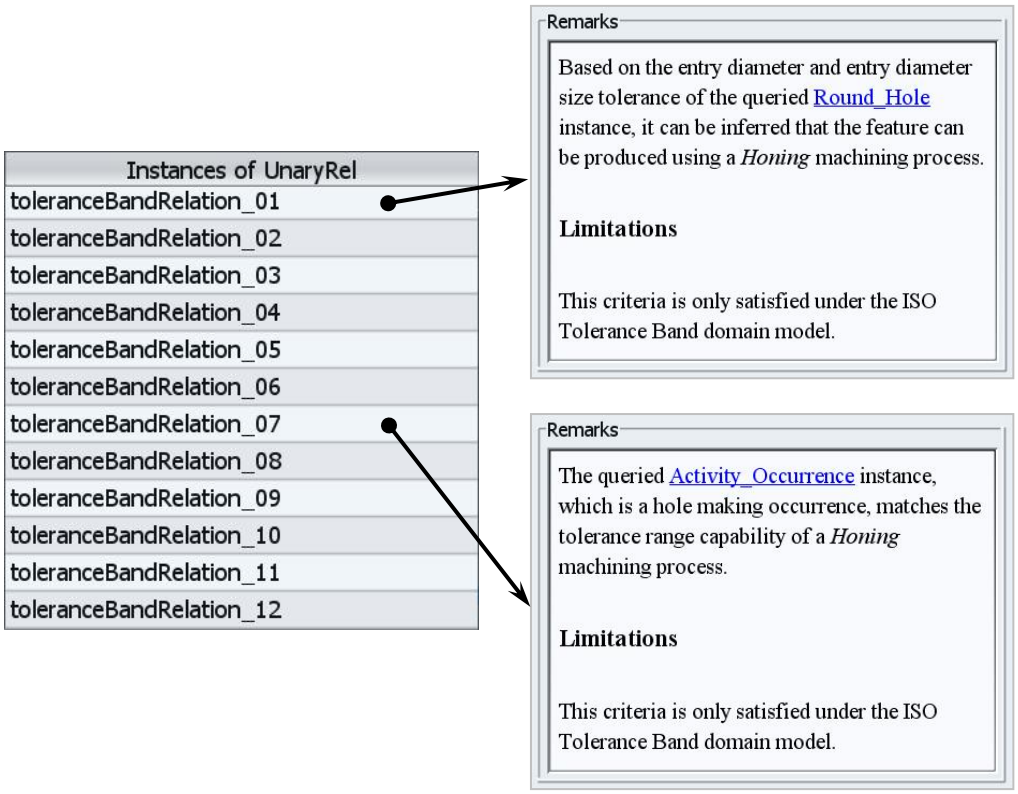


Figure 8-28 Inference Relations Defined in the ISO Tolerance Band Model

8.5.3 Reconciliation Scenario

The reconciliation scenario in this test case is to take place at the instance level, due to the nature of the knowledge captured in the external domain model, which interacts only with discretely-represented knowledge. The formalised inference relations in the “ISO Tolerance Band Model” are to be used as semantic mapping concepts in order to help establish cross-domain correspondences. Figure 8-29 depicts three pairs of instances to be reconciled: “Reamed_Hole_A” and “Bored_Hole_E” **(A3)**, “Drilled_Hole_A” and “Bored_Hole_E” **(B3)** and “occ_Make_Reamed_Holes_AB” and “occ_Make_Bored_Holes_ABCDE” **(C3)**.

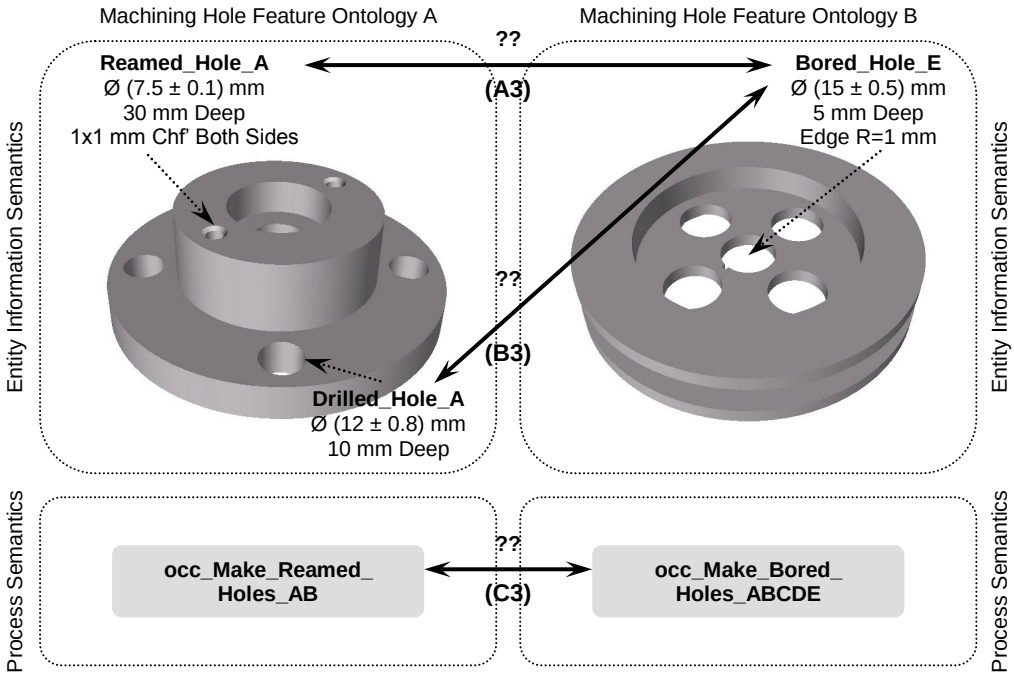


Figure 8-29 Reconciliation Scenario at the Instance Level

It is to be noted that the ontology mapping process used in Test Case 3 follows a similar implementation to the one explained in section 8.4.4 (see Figure 8-19). The only difference is that instead of loading semantic mapping concepts based on foundation semantics, the KFL file for the "ISO Tolerance Band Model" is loaded and saved. The various actions involved in this ontology mapping process use a new OMS named "Test Case 3".

8.5.4 Interoperability Evaluation and Verification

"Reamed_Hole_A" v/s "Bored_Hole_E"

To build the query for reconciling "Reamed_Hole_A" and "Bored_Hole_E", the appropriate cell in the matrix configuration on the main panel of the Interoperability Evaluation Assistant is activated (see Figure 8-30 label **(D3)**). A new window is opened and the required text fields **(E3)** are used to input the names of the two cross-domain instances of "Round_Hole". The "Submit" button is pressed and the relevant query is retrieved **(F3)**. The query is copied and pasted in the query tool in IODE and run. The results of the query are illustrated in Figure 8-31.

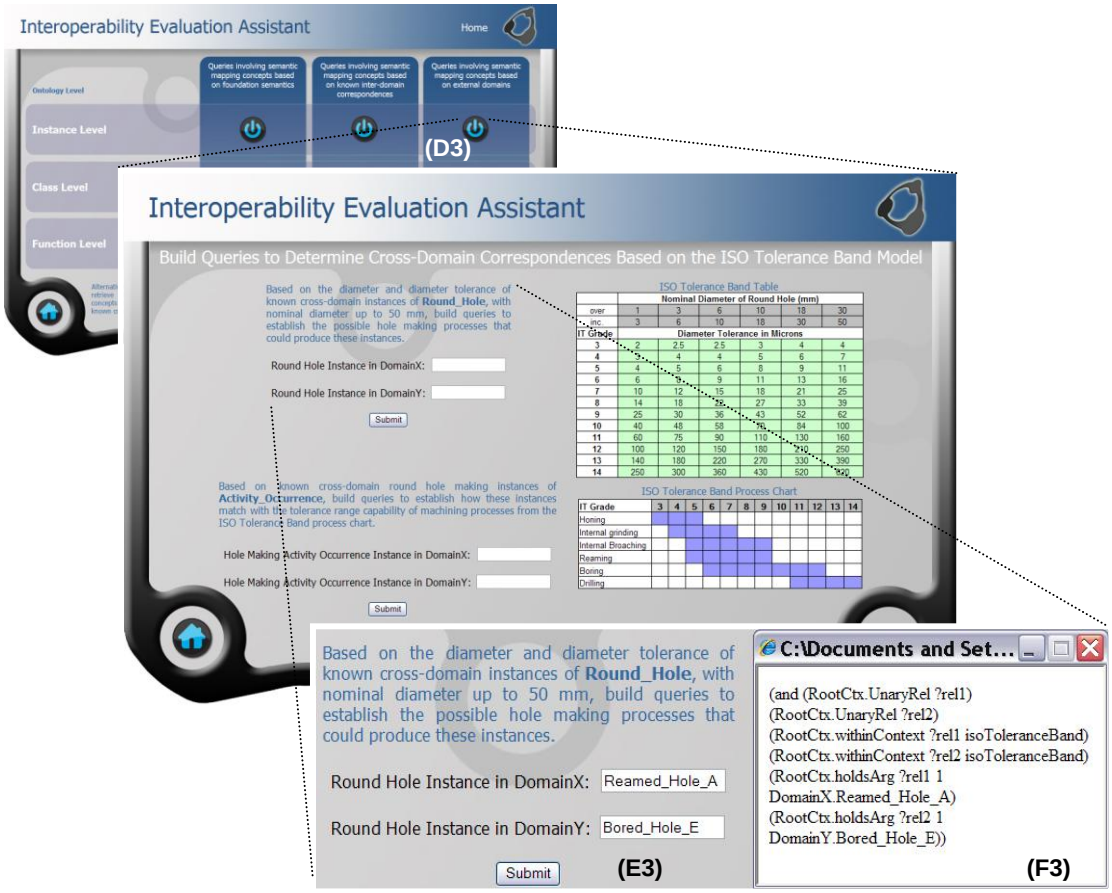


Figure 8-30 Using the Interoperability Evaluation Assistant to Build a Query for Reconciling Two "Round_Hole" Instances

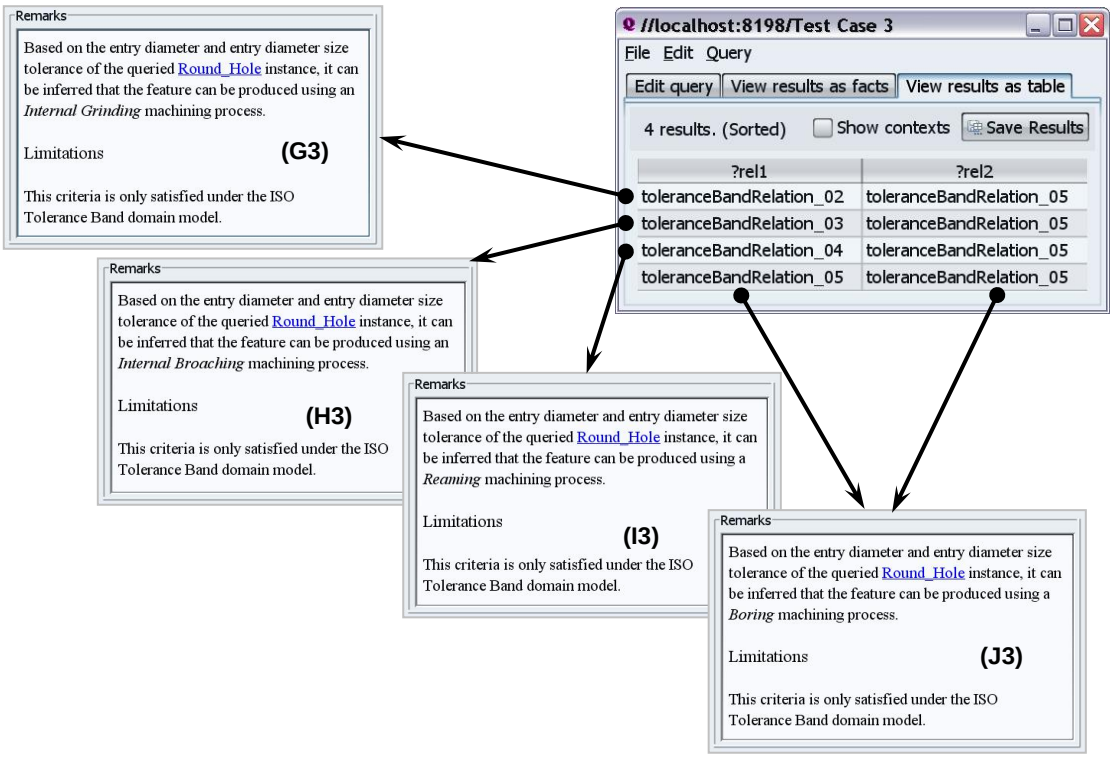


Figure 8-31 Viewing and Browsing Semantic Mapping Concepts between "Reamed_Hole_A" and "Bored_Hole_E"

Figure 8-31 shows that using the semantics from the “ISO Tolerance Band Model” it is possible to understand the machining processes that could be associated to “Round_Hole” instances of given diameters and diameter tolerances. In this case, the “Reamed_Hole_A” from “Machining Hole Feature Ontology A” has been inferred, via the external domain model, as being a feature that could be produced using internal grinding **(G3)**, internal broaching **(H3)**, reaming **(I3)** as well as boring **(J3)**. “Bored_Hole_E” from “Machining Hole Feature Ontology B”, on the other hand, has been inferred as a suitable candidate which can be produced using some boring machining process **(J3)**. These results have been logically formulated based on the nominal diameter and diameter tolerances carried by the various domain-defined hole features, and articulated through the “ISO Tolerance Band Model” semantics.

“Drilled_Hole_A” v/s “Bored_Hole_E”

Figure 8-32 depicts the results of processing a query to determine cross-domain correspondences, based on the “ISO Tolerance Band Model”, between the “Round_Hole” instances “Drilled_Hole_A” and “Bored_Hole_E”. It is seen that both instances are suitable candidates which could be produced using boring as machining process.

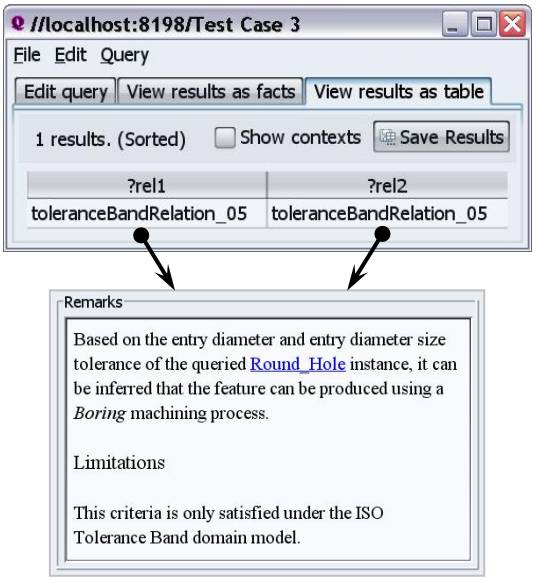


Figure 8-32 Viewing and Browsing Semantic Mapping Concepts between “Drilled_Hole_A” and “Bored_Hole_E”

“occ_Make_Reamed_Holes_AB” v/s “occ_Maked_Bored_Holes_ABCDE”

To determine the correspondences that hold between “occ_Make_Reamed_ Holes_AB” and “occ_Make_Bored_Holes_ABCDE” a query is built as shown in Figure 8-33.

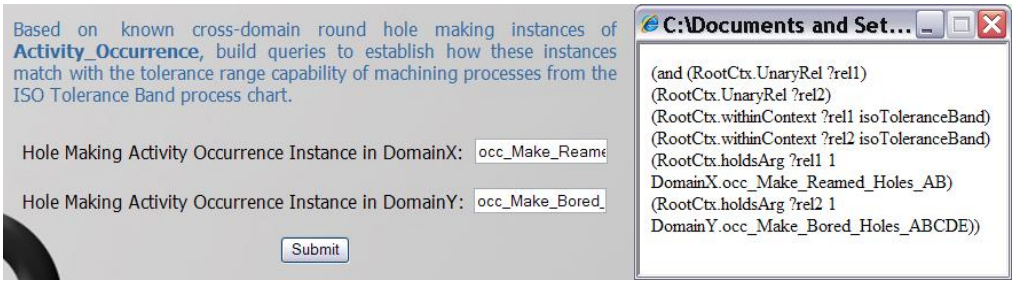


Figure 8-33 Using the Interoperability Evaluation Assistant to Build a Query for Reconciling Two “Activity_Occurrence” Instances

The query is copied and pasted in IODE’s query tool and executed. Figure 8-34 portrays the results of processing the query where the deductions indicate that the occurrence “occ_Make_Reamed_Holes_AB” could potentially be matched to the tolerance range capability of internal grinding **(K3)**, internal broaching **(L3)**, reaming **(M3)** and boring processes **(N3)**. The occurrence “occ_Make_Bored_Holes_ABCDE” conforms to the tolerance range of boring machining processes **(N3)**.

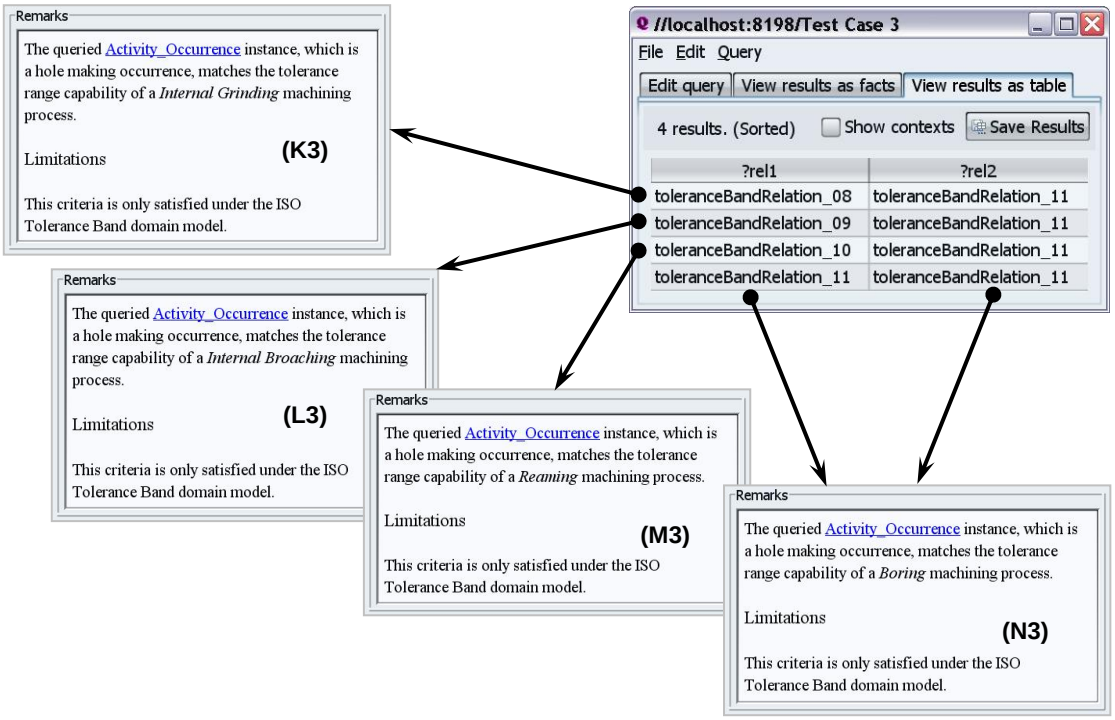


Figure 8-34 Viewing and Browsing Semantic Mapping Concepts between "occ_Make_ Reamed_Hole_AB" and "occ_Make_Bored_Holes_ABCDE"

8.5.5 Discussions and Validation of Results

This test case has shown that the knowledge structures contained in an external domain ontology, notably the inference relations present within the “ISO Tolerance Band Model”, can be employed as semantic mapping concepts in order to evaluate and verify the correspondences between cross-domain arguments. The external domain model, based on the Foundation Layer, carries valuable semantics which help articulate cross-domain arguments. It is to be noted that in this test case, emphasis has not been laid on logical proof (verification), since the evaluation of queries retaining results already points to their verification at computational level.

The type of reconciliation mechanism under consideration in Test Case 3 has proved its benefits at the instance level. This is because the formalised knowledge in the “ISO Tolerance Band Model” facilitates interactions with instances carrying discrete knowledge. For extending the reconciliation capabilities using external domains, it is possible to further exploit other standards-based models such as ISO limits and fits for holes and shafts. Similar domain models would require specialisation from the Foundation Layer.

8.6 Test Case 4: Reconciliation Using Semantic Mapping Concepts Based on Known Cross Domain Correspondences

8.6.1 Aim and Objectives

The aim of Test Case 4 is to provide a proof of concept for the reconciliation between two intra-system domains. The mode in which reconciliation is to take place is through the use of semantic mapping concepts based on known cross-domain correspondences. Two domain models are under scrutiny namely (1) “Machining Hole Feature Ontology A” (machining process viewpoint) and (2) “Design Hole Feature Ontology A” (functional design viewpoint). The following competency question applies to Test Case 4:

- Is it possible to exploit the semantic mapping concepts based on known cross-domain correspondences to evaluate and verify correspondences between “Machining Hole Feature Ontology A” and “Design Hole Feature Ontology A”?

There are three objectives to this test case namely (1) the deployment of the Semantic Reconciliation and Interoperability Evaluation layers for reconciling between the two domain models, which are in the same system domain, (2) the deployment of semantic mapping concepts based on known cross-domain correspondences and (3) the use of IODE, the Interoperability Evaluation Assistant and the query tool in IODE for evaluating and verifying intra-domain correspondences.

8.6.2 Design Hole Feature Ontology A

The “Design Hole Feature Ontology A” captures entity information semantics from viewpoints including GD & T and design function. Figure 8-35 identifies the classes of concepts that are being explored in this domain ontology. These classes adopt terminologies and semantic definitions that are relevant to the field of design. In particular, the dimensional parameters carried by the

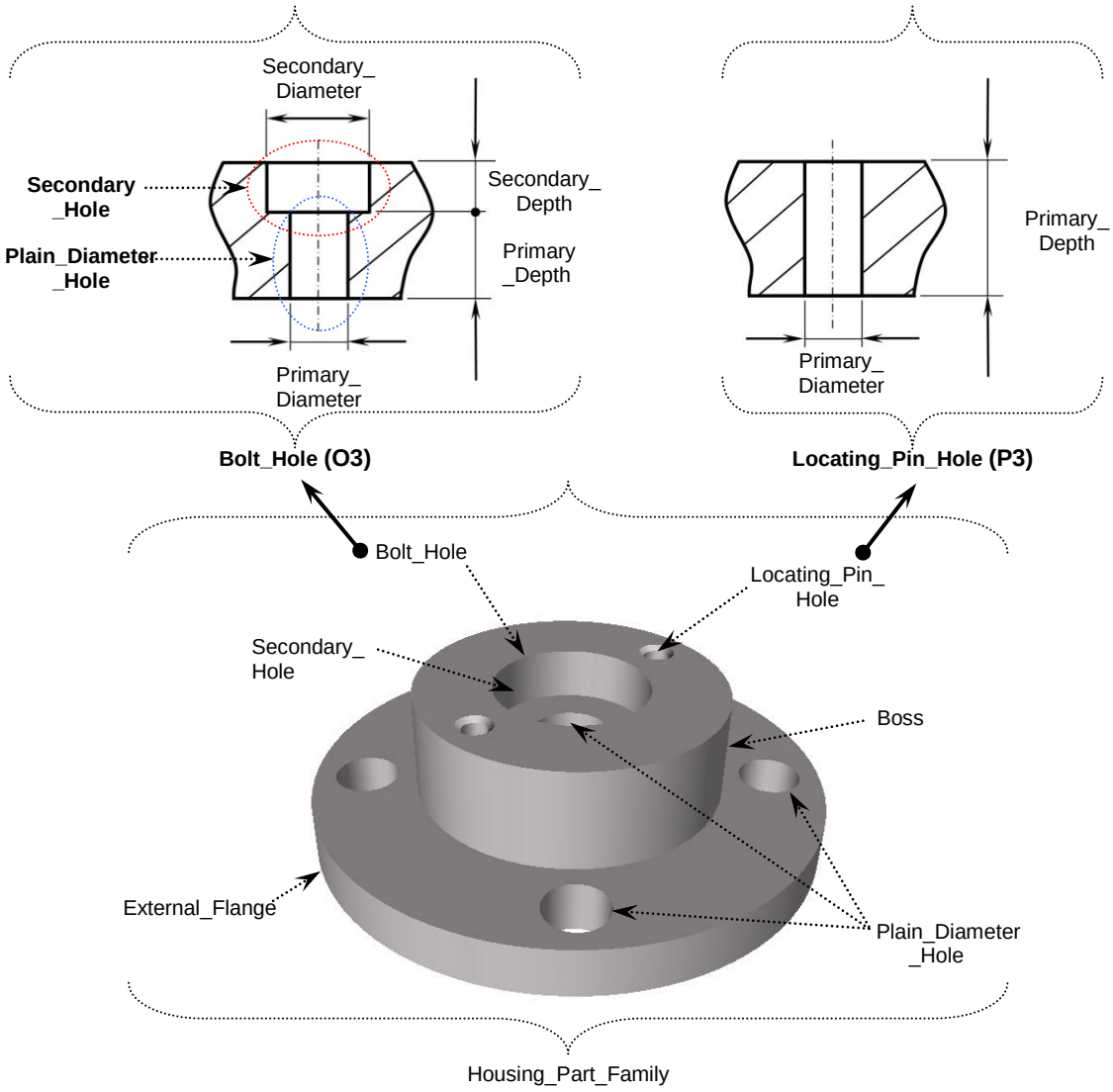
various classes of features have been specifically chosen in order to reflect the design perspective on the features pertaining to “Housing_Part_Family”.

Bolt_Hole

- A bolt hole is a compound feature
- Every bolt hole involves a plain diameter hole and a secondary hole which are elements of the bolt hole
- The plain diameter hole of a bolt hole is the base feature of the bolt hole
- The secondary hole element of a bolt hole has a diameter value which is always greater than that of the plain diameter hole element of the same bolt hole.

Locating_Pin_Hole

- Every locating pin hole holds exactly two circular closed profiles of identical primary diameter.
- Every locating pin hole holds exactly one linear path of primary depth.
- Every locating pin hole that has a through hole bottom condition needs to be chamfered in order to facilitate easy insertion.



Miscellaneous (Q3)

- Every instance of feature and artifact in the Design Hole Feature Ontology A holds some design function.
- Every housing has some compulsory external flange, boss, bolt hole, plain diameter hole and locating pin hole as features present on the housing.
- Every housing is made up of some aluminium material.

Figure 8-35 Examples of Classes and Informal ICs captured in the "Design Hole Feature Ontology A"

Two example features have been elaborated in Figure 8-35. The entity information class “Bolt_Hole” **(O3)** of compound feature property has been rigorously defined using a number of ICs. The same understanding applies to the class “Locating_Pin_Hole” **(P3)** in terms of the dimensional parameters that define the latter and other necessary conditions such as “every locating pin hole that has a through hole bottom condition needs to be chamfered in order to facilitate easy insertion”. This captures a necessary design aspect that needs to be fulfilled during the population of instances in the KB for the “Design Hole Feature Ontology A”. Figure 8-35 further depicts other forms of ICs **(Q3)** relevant to the functional design viewpoint in the domain ontology. Appendix D.2 can be consulted for a formalised interpretation of all the concepts and ICs explored for this domain ontology.

A concrete state of the “Design Hole Feature Ontology A” has also been captured by defining the semantic representation of discrete knowledge through instantiation. Figure 8-36 portrays this concrete state of design entity information.

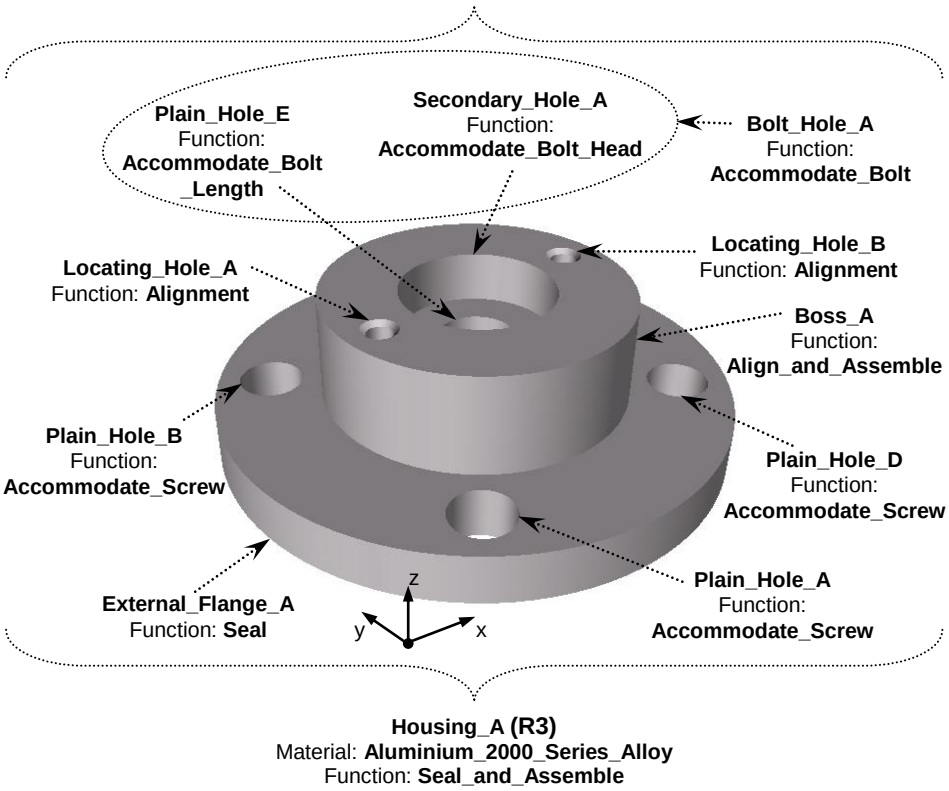


Figure 8-36 Populated Entity Information Semantics for Capturing a Concrete State of "Design Hole Feature Ontology A"

The various features present on “Housing_A” **(R3)** carry exact dimensional semantics to that of “Machined_Housing_A” (see Figure 8-8 section 8.3.2.4), but the terminologies and semantics of their defining structures are different since, for example, in the “Design Hole Feature Ontology A”, functional information of features is a prerequisite, which is not the case in its machining viewpoint counterpart. Note that the instance “Plain_Hole_C” has not been shown on the diagram because it is hidden. Figure 8-37 then identifies part of the implemented taxonomy for the “Design Hole Feature Ontology A” and sample instances that satisfy both foundation and domain-defined ICs.

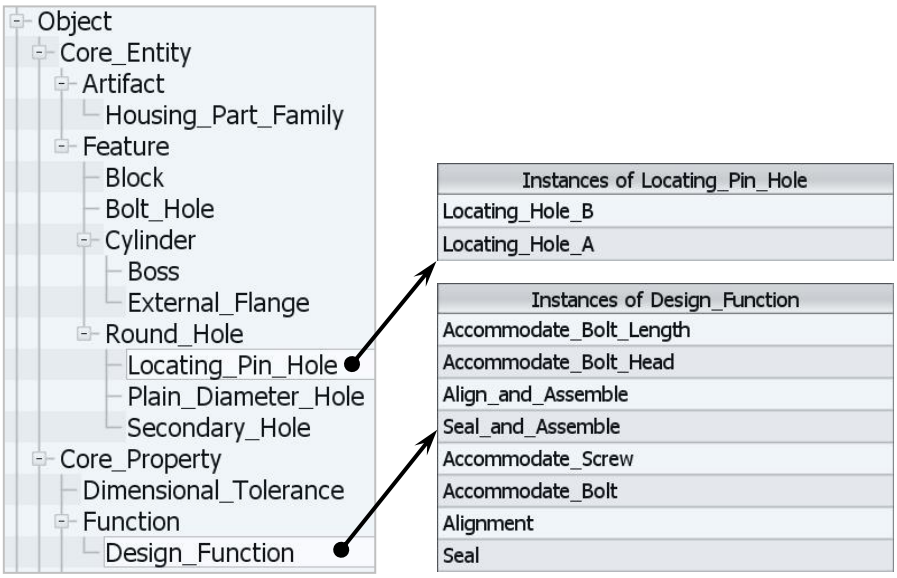


Figure 8-37 Example of Classes and Instances Defined in the "Design Hole Feature Ontology A"

8.6.3 Reconciliation Scenarios

The ontology mapping process used in Test Case 4 follows a similar implementation approach to the one previously explained in section 8.4.4 (see Figure 8-19). However, instead of loading semantic mapping concepts based on foundation semantics, the KFL file for the semantic mapping concepts based on known cross-domain correspondences between “Design Hole Feature Ontology A” and “Machining Hole Feature Ontology A” (see Appendix E.2 if needed) is loaded and saved. The various actions involved in this ontology mapping process use a new OMS named “Test Case 4”. In the

process, the context for “Design Hole Feature Ontology A” has been renamed to “DomainX” and that of “Machining Hole Feature Ontology A” to “DomainY”.

8.6.3.1 Reconciliation at the Class Level

Figure 8-38 illustrates the concepts to be reconciled at the class level. These involve the discovery of correspondences between three pairs of entity information classes namely “Boss” and “Turned_Boss” **(S3)**, “Bolt_Hole” and “Counterbore_Hole” **(T3)** and “Primary_Depth” and “Drilled_Hole_Depth” **(U3)**.

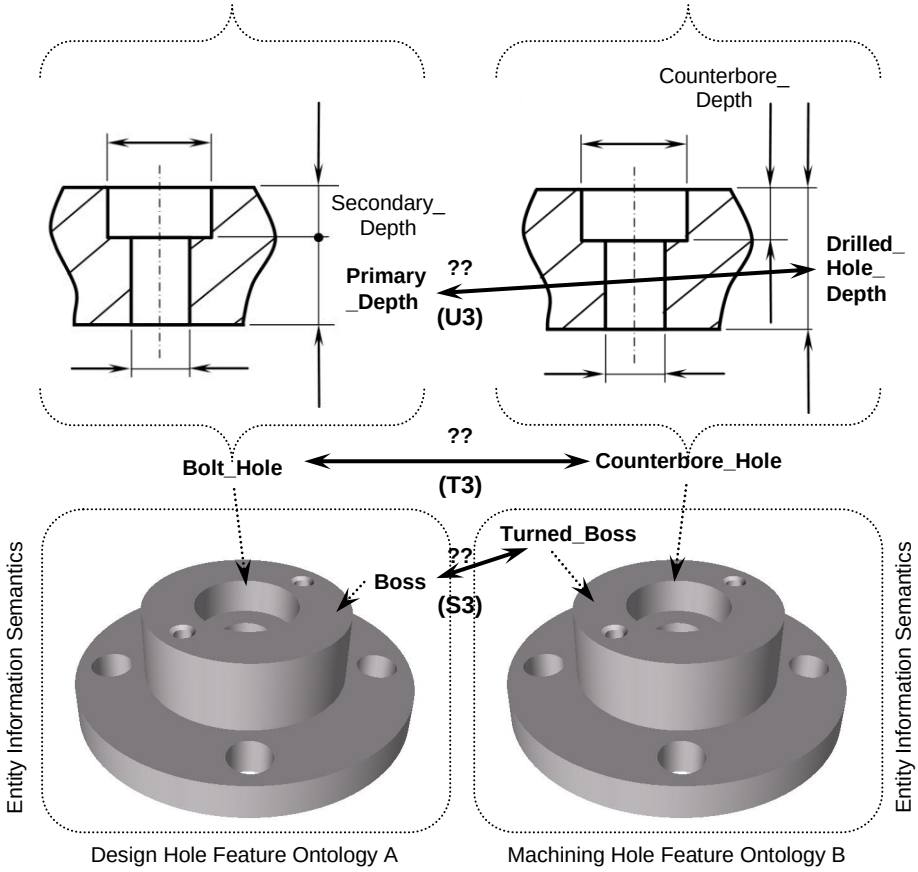


Figure 8-38 Reconciliation Scenario at the Class Level

8.6.3.2 Reconciliation at the Function Level

Two unary domain-defined ontological functions are to be reconciled as shown in Figure 8-39. The correspondences that hold between the functions “inch” in “Design Hole Feature Ontology A” and “inch” in “Machining Hole Feature Ontology A” **(V3)** are to be discovered.

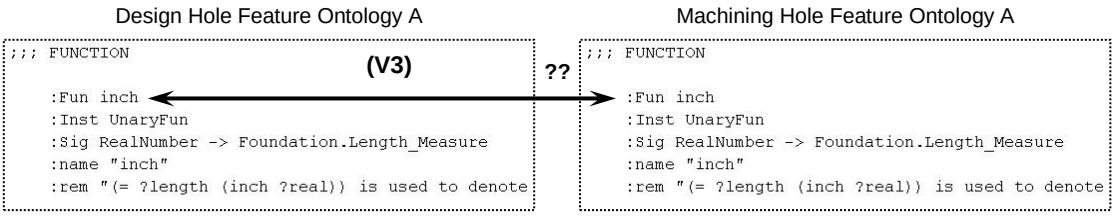


Figure 8-39 Reconciliation Scenario at the Function Level

8.6.3.3 Reconciliation at the Instance Level

Figure 8-40 depicts two pairs of individuals to be reconciled at the instance level. Correspondences need to be discovered between the entity information instances “External_Flange_A” and “Turned_Flange_A” **(W3)**, and the “Plain_Hole_A” and “Drilled_Hole_D” **(F2)**.

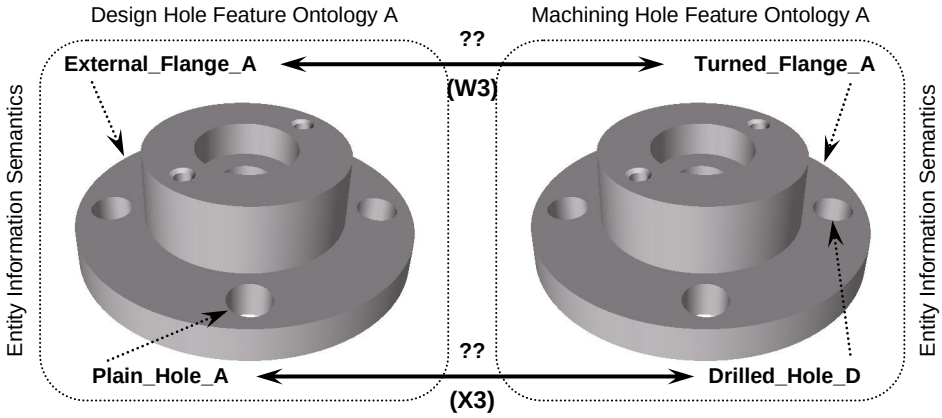


Figure 8-40 Reconciliation Scenario at the Instance Level

8.6.4 Interoperability Evaluation and Verification

8.6.4.1 Discovery of Semantic Mapping Concepts at the Class Level

Boss v/s Turned_Boss

The Interoperability Evaluation Assistant is first invoked as shown in Figure 8-41. The Java-based module for building queries involving semantic mapping concepts based on known cross-domain correspondences is used, where the names of the classes are typed in the relevant text field **(Y3)**. On clicking the

submit button, the query is retrieved **(Z3)** and is copied and pasted in the query tool in IODE.

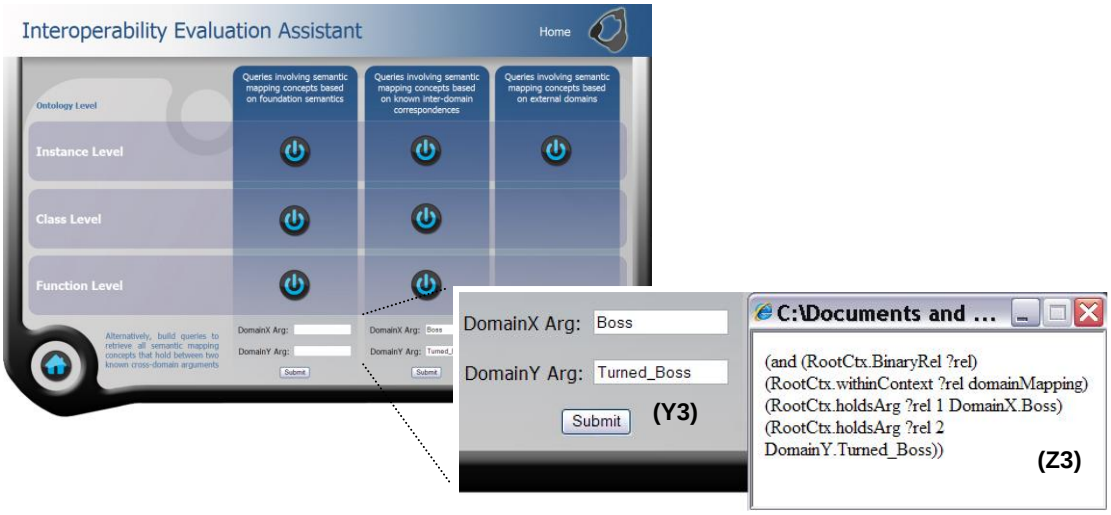


Figure 8-41 Using the Interoperability Evaluation Assistant to Build a Query for Reconciling the Classes "Boss" and "Turned_Boss"

The results of executing the above query is illustrated in Figure 8-42. Two mapping results have been discovered and the nature of the interoperability between the classes "Boss" and "Turned_Boss" has been captured in the informal remarks associated to each semantic mapping concept. Note that the results could easily be verified through the logic-based proof structure. In all cases, the verification is confirmed through the presence of query results.

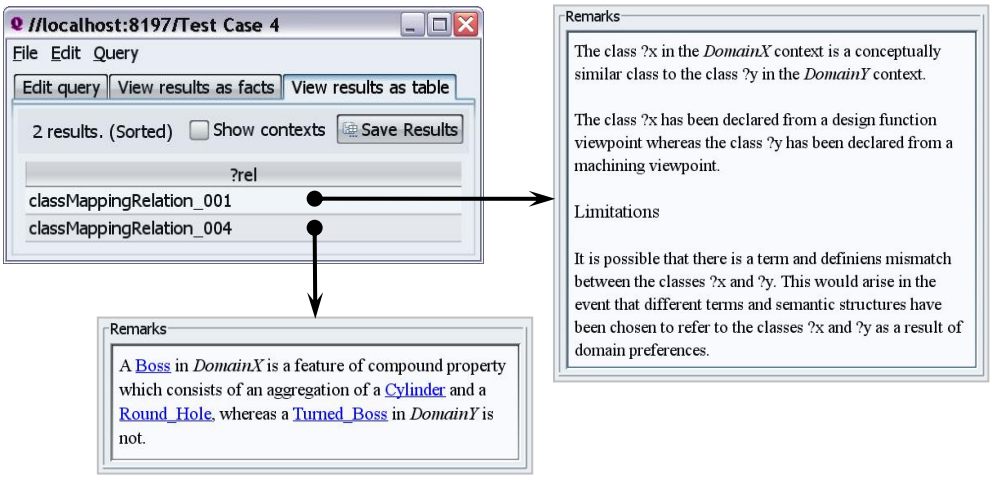


Figure 8-42 Viewing and Browsing Semantic Mapping Concepts between the Classes "Boss" and "Turned_Boss"

Bolt_Hole v/s Counterbore_Hole

The query to be processed to discover the semantic mapping concepts between the classes “Bolt_Hole” and “Counterbore_Hole” is retrieved in a similar way as documented previously. Figure 8-43 depicts the results of processing the relevant query. Note that the semantic mapping concept “classMappingRelation_001” is the same as the one shown in Figure 8-42. The other alignment “classMappingRelation_002”, when browsed, provides an enhanced view of the differences between the two classes.

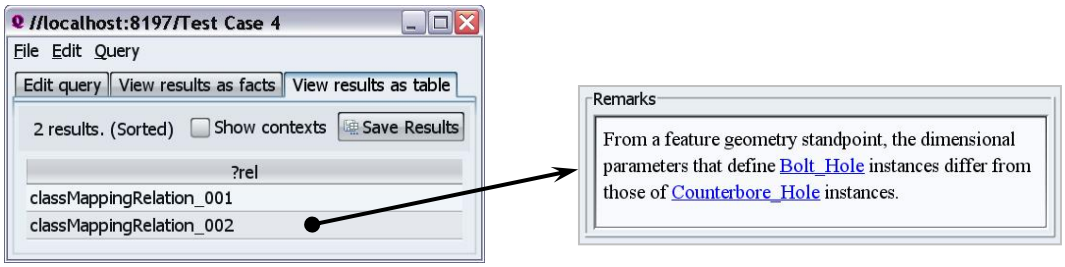


Figure 8-43 Viewing and Browsing Semantic Mapping Concepts between the Classes "Bolt_Hole" and "Counterbore_Hole"

Primary_Depth v/s Drilled_Hole_Depth

A query is also formulated in order to process the semantic mapping concepts between the classes “Primary_Depth” and “Drilled_Hole_Depth”. The results between the two classes are shown in Figure 8-44. In addition to the semantic mapping concept “classMappingRelation_001”, another mapping concept identified as “classMappingRelation_003” is also present. Browsing the remarks clearly depicts some clear differences between the two classes.

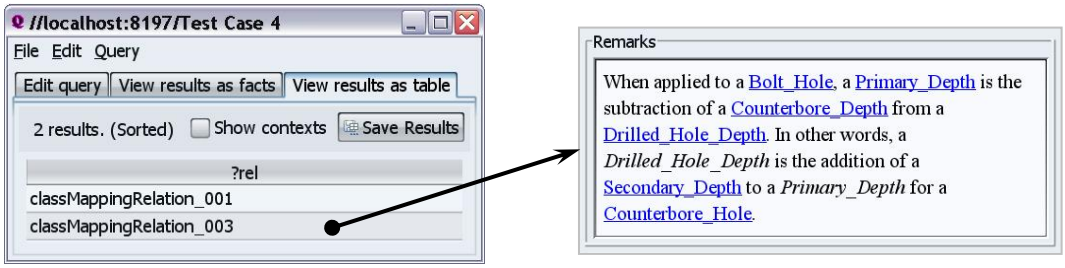


Figure 8-44 Viewing and Browsing Semantic Mapping Concepts between the Classes "Primary_Depth" and "Drilled_Hole_Depth"

8.6.4.2 Discovery of Semantic Mapping Concepts at the Function Level

inch v/s inch

Figure 8-45 identifies the result of executing a query to retrieve all semantic mapping concepts that exist between the two ontological functions “inch” in “Design Hole Feature Ontology A” and “inch” in “Machining Hole Feature Ontology A”. It is clear from the result that both functions are semantically equivalent as there are no perceivable mismatches between them.

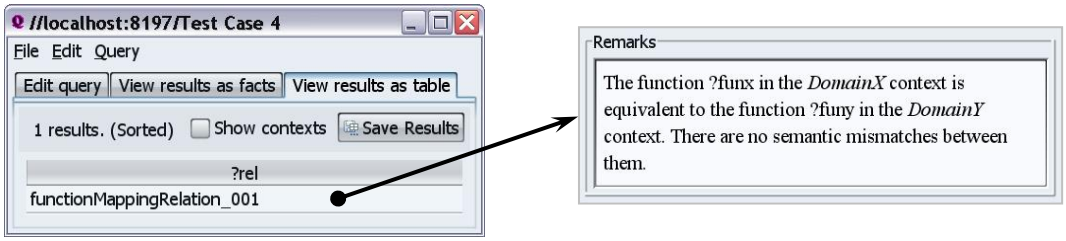


Figure 8-45 Viewing and Browsing Semantic Mapping Concepts between the Ontological Functions "inch" and "inch"

8.6.4.3 Discovery of Semantic Mapping Concepts at the Instance Level

External_Flange_A v/s Turned_Flange_A

There exists only one alignment between the instances “External_Flange_A” and “Turned_Flange_A”. As can be seen in Figure 8-46, the remarks associated to the semantic mapping concept “instanceMappingRelation_002” include similarities and limitations to the interoperability between the two individuals.

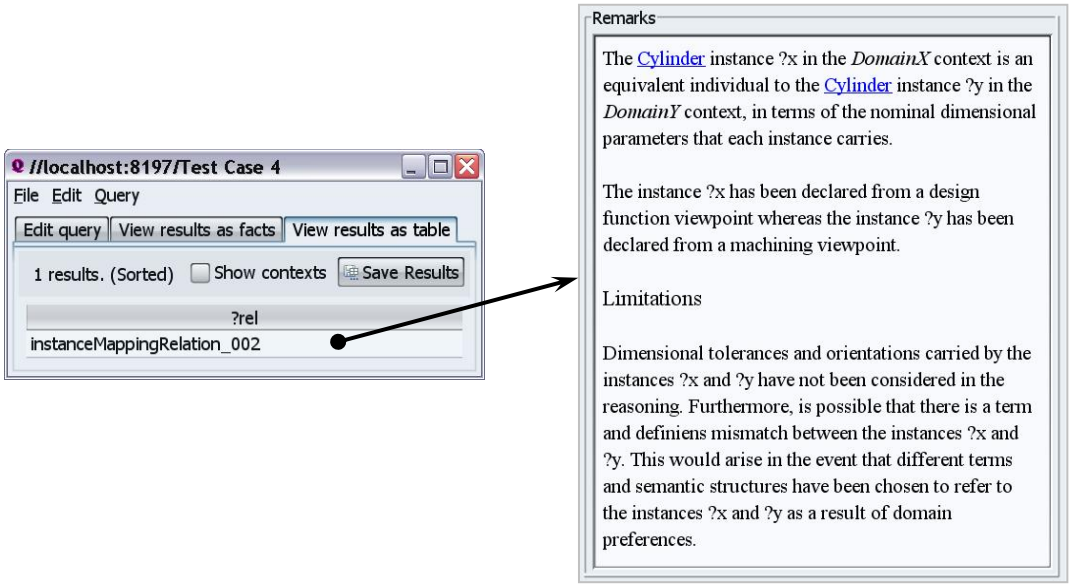


Figure 8-46 Viewing and Browsing Semantic Mapping Concepts between the Instances "External_Flange_A" and "Turned_Flange_A"

Plain_Hole_A v/s Drilled_Hole_D

When the query for finding semantic mapping concepts between the instances “Plain_Hole_A” and “Drilled_Hole_D” is retrieved and executed, one correspondence is obtained (see Figure 8-47). The semantic mapping concept “instanceMappingRelations_001” carries valuable informal semantics in the form of remarks, which the user can interpret.

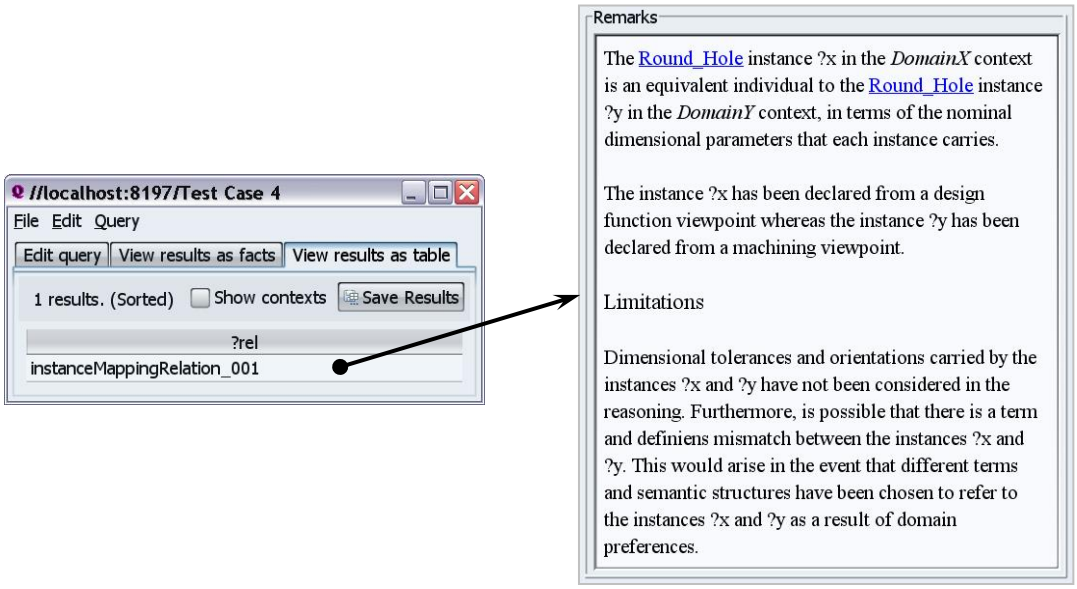


Figure 8-47 Viewing and Browsing Semantic Mapping Concepts between the Instances "Plain_Hole_A" and "Drilled_Hole_D"

8.6.5 Discussions and Validation of Results

Test Case 4 has illustrated the effectiveness of semantic mapping concepts based on known cross-domain correspondences. These semantic mapping concepts have been used to evaluate and verify the correspondences that hold between cross-domain arguments that could be present at the class, function and instance levels of domain models within a single system domain. The test case results confirm that the competency question identified in section 8.6.1 has been positively answered.

The principle for semantic reconciliation documented in this section could also be used for inter-system interoperability. If two inter-system domains are to the reconciled using this approach, this would require that the knowledge engineer understands the ontological structures from both domain representations before formulating the relevant semantic mapping concepts. Furthermore, unlike the other two modes of semantic mapping concepts, those based on known cross-domain correspondences carry more accurate interoperability information.

However, the formal logic governing these semantic mapping concepts may, in certain cases, be less rigorous than in the other two modes of semantics mapping concepts. For example, it has not been possible to fully capture the formal semantics to express that the class “Primary_Depth”, when applied to the “Bolt_Hole” class, is the subtraction of the “Counterbore_Depth” from the “Drilled_Hole_Depth” classes. Consequently, this indicates that extensions to the logical basis of semantic mapping concepts based on known cross-domain correspondences may be required in certain situations.

8.7 Summary of Chapter

This chapter has concentrated on four test cases as part of a complete case study in order to provide a proof of concept for the Semantic Manufacturing Interoperability Framework (SMIF) (meets the fifth objective of this research in Chapter 1 section 1.3.1). Figure 8-48 identifies how the four test cases

implemented in this chapter demonstrate that the framework satisfies the semantic requirements (see Chapter 3). The four test cases altogether show that it has been possible to capture a number of viewpoints in domain models **(A4)** (Requirement 1). The various semantic relationships present in the ontological definition of the heavyweight manufacturing ontological foundation have enabled the implemented domain models in the test cases to reuse these relationships to link multiple viewpoints **(B4)** (Requirement 2).

	Framework Components	Test Cases	
Requirement 1	Combined approach using PSL, CPM and ISO 10303 AP224 in the Foundation Layer	**(A4)**	A number of viewpoints have been considered in the various test cases including GD & T, design entity information, machining entity information, design function and process planning.
Requirement 2	Definition of relationships between entities and processes, entities and their functions, etc. in the Foundation Layer	**(B4)**	Defined relations supported in the heavyweight manufacturing ontological foundation have been reused during the implementation of the various domain models in the test cases.
Requirement 3	Heavyweight semantics of core concepts developed in the Foundation Layer, which are extended in the Domain Ontology Layer	**(C4)**	The semantics from the Foundation Layer have been reused and extended for the construction of domain models in test cases 1 and 2. These domain models occur across system domains.
Requirement 4	Combined semantic technologies used throughout the SMIF	**(D4)**	The implementation of the SMIF using the IODE platform has been useful for enabling the deployment of the various test cases.
Requirement 4a	Exploitation of the Knowledge Framework Language (KFL)	**(E4)**	The maintained use of the Common Logic-based KFL provides a highly expressive formalism to encode ontology-based content and discrete knowledge in KBs. Appendix B also supports the motivation for the use of KFL.
Requirement 4b	The Semantic Reconciliation Layer	**(F4)**	Commitment of the test case ontologies to the Foundation Layer help reduce relation mismatches via the controlled specialisation approach. Semantic mapping concepts have helped to identify possible semantic mismatches.
Requirement 4c	Specification of logic-based semantic mapping concepts and ontology mapping processes in the third framework layer	**(G4)**	The three modes in which semantic mapping concepts can occur have been tested in test cases 2, 3 and 4. Reconciliation has been shown at the class, function and instance levels of domain models.
Requirement 4d	The Semantic Reconciliation and Interoperability Evaluation layers	**(H4)**	The use of the Interoperability Evaluation assistant interface and the query tool in IODE enable rapid query responses to be obtained. The Interoperability Evaluation Assistant has been optimised for use with the SMIF components.

Figure 8-48 Implications of the Test Cases on the Semantic Requirements

It has been possible using the set of core concepts from the Foundation Layer to develop the “Machining Hole Feature Ontology A” and “Machining Hole Feature Ontology B” domain models, which reside across system domains **(C4)** (Requirement 3). Furthermore, the implementation of all the levels of the SMIF using the IODE platform, as providing a suite of semantic technologies, has enabled the deployment and reconciliation of the test case domain models **(D4)** (Requirement 4). It has also been possible via the implementation of the four test cases to satisfy the sub-requirements of Requirement 4 (refer to labels **(E4)** to **(H4)** on Figure 8-48).

The first test case has focused on the integrity-driven specialisation of a “Machining Hole Feature Ontology A” and during the process, it has been possible to prove that the first aspect of the research hypothesis is feasible. Overall, Test Case 1 has demonstrated that the specification of a heavyweight manufacturing ontological foundation provides a basis for the integrity-driven specialisation of domain models. The understanding has also been applied to “Design Hole Feature Ontology A”, “Machining Hole Feature Ontology B” as well as the “ISO Tolerance Band Model”. Test cases 2, 3 and 4 have specifically looked at the second aspect of the research hypothesis. These test cases have shown that the specification of a heavyweight manufacturing ontological foundation also supports the capability to evaluate and verify correspondences between pairs of domain models that have been specialised from the heavyweight manufacturing ontological foundation.

The logic-based definitions of semantic mapping concepts is key to enabling the interoperability evaluation and verification process. In the various test cases, the verification part is included in the successful retrieval of semantic mapping concepts, as these results already confirm that they have been verified through deductive reasoning before being displayed. Furthermore, this chapter has shown the importance of a system for aiding the management and formulation of interoperable knowledge queries prior to being executed. The significance of the Interoperability Evaluation Assistant has been particularly pertinent in satisfying this purpose.

9 Discussions, Conclusions and Future Work

9.1 Introduction

The research work documented in this thesis has investigated a novel ontology-based framework to support semantic interoperability in product design and manufacture. The four levels of the Semantic Manufacturing Interoperability Framework (SMIF) have been explored alongside the interactions and mechanisms that hold between the different levels. The deployment of an experimental system and conduction of a number of test cases applied to the framework has culminated in a valuable understanding of the potentials and limitations of the researched approach.

This chapter compiles the overall understanding developed in this research and exposes a discussion of the outcome of the implemented framework with respect to various issues and concerns in section 9.2. Section 9.3 provides the concluding remarks to this work and section 9.4 proposes important recommendations for future work.

9.2 Discussions

9.2.1 Ontology Development Methodology

The ontology development methodology applied to this research has consisted of the Knowledge Engineering Methodology (Noy and McGuinness, 2001) accompanied by the use of IDEF5 schematics (Knowledge Based Systems Inc., 1994) for graphically representing ontological content prior to implementation. The two combined methods have proved adequate into setting a strategic view on the ontology-based framework, for example, through the investigation of requirements to support semantic interoperability in product design and manufacture investigated in Chapter 3, as well as to support the design and implementation of the various ontological structures.

The only probable issue with the actual state of IDEF5 schematics for ontology development is that there is currently no commercially-available tool which could be sourced for directly deriving Common Logic code from the schematics. Therefore, at present, IDEF5 schematics only visually help the ontology development process. An attractive step ahead would involve closely interfaced ontology graphical tools in order to enhance the development of ontologies using Common Logic-based formalisms.

9.2.2 Semantic Technologies

The alignment of the SMIF with the investigated requirements has been documented in Chapter 4 (section 4.8). In this section, one of the requirements is further reviewed, namely Requirement 4, which has been restated next.

- **Requirement 4:** There is a need for harnessing the appropriate semantic technologies in order to facilitate the formal capture of domain semantics and to support shared meaning across domain ontologies.

Additional understanding gathered during the experimental implementation and case study has shown distinct benefits in relationship to the above requirement. These include the idea of supporting heavyweight semantics using expressive Common Logic-based formalisms, such as KFL, and the performance of semantic mapping mechanisms using the SMIF and its implementation platform. One of the factors related to the performance of semantic reconciliation is related to the time for processing semantic alignments as well as the time spent in deductive reasoning during querying.

During implementation it was observed that the loading and saving of about 100 semantic mapping concepts, accompanied by heavyweight logic, took about one minute to be performed. Querying procedures for resolving all semantic mapping concepts, that hold between two cross-domain arguments, in a single transaction took less or about 10 seconds to be processed. This clearly indicates that an attractive direction for ontology matching exists when

weighted against other ontology mapping methods which may take several minutes, hours or even days to complete a matching task (Shvaiko and Euzenat, 2008). However, it should not be forgotten that the performance of mapping is dependent on the size of the ontologies to be reconciled as well as the size of the file containing the semantic alignments. Overall, opportunities still remain for comparing various ontology mapping methods in terms of their performance and exactness of semantic reconciliation processes as well as their support for the evaluation and verification of interoperable knowledge.

9.2.3 Semantic Structures

In the context and scope of this work, a specific set of semantics of core feature-based concepts arising in product design and manufacture has been investigated. However, it is seen that the semantic structures have been narrowed down to simple product representations, involving hole features and process ordering semantics from PSL, in order to provide the ability to explore all the levels of the SMIF. Hence, it is clear that the breadth of concepts arising in the Foundation Layer needs to be expanded to embrace more complex product lifecycle semantics, for example from (1) Product Life Cycle Support (PLCS) (ISO 10303 AP239) and (2) the inclusion of other theories supported in PSL.

PSL, for example, comprises concepts from various other theories like the duration and ordering theories and resource theories (PSL Website, 2009). The latter would be particularly relevant for capturing the semantics of resource requirements in process execution sequences, where its extensions would allow the definition of resource roles and the way resources are consumed during the course of manufacturing process sequences. These important aspects in the world of product design and manufacture have not been considered in this work, and for this reason, a need is identified for supporting similar core intuitions.

Furthermore, from the case study, it is evident that confined samples of common hole features have been considered in the design and manufacturing

domains. It is well-known that an extensive range of hole features exist such as gun-drilled holes, ground holes and electrical discharge machined holes. Considerations for similar hole features would imply alternative representations through the extension and application of the modelling approach explored in the various domain models from the case study. Such representations on real parts would not only imply understanding the relationships between the hole features but also relationships between the types of parts on which the hole features are designed and manufactured.

Hence, another attractive opportunity exists for incorporating additional core semantics for capturing feature definitions in the context of design and manufacturing part families. Studies in the area of part families and features and their relationships would provide a suitable basis for formalising more complex semantics between entity and manufacturing resource information as well as process concepts with respect to the notion of part families. Figure 9-1 shows an IDEF5 schematic version of a significant portion of the original high-level UML diagram proposed by Gunendran and Young (2008), identifying a generic, yet meaningful interpretation of the need to reinforce relationship semantics between manufacturing features within a part family context.

The branch of the diagram following “Machining Operation Sequence”, “Machining Operation”, “Setup Sequence”, “Setup”, “Step Sequence” and “Step”, provides the necessary details for part family manufacturing method descriptions. The other branch corresponding to “Stage Sequence”, “Stage”, “Step Sequence” and “Step”, on the other hand, supports the description of manufacturing methods in relationship to manufacturing features. The main observation made from the high-level model is that there is a need to understand and define semantic relationships between the classes “Machining Operation Sequence” **(A)** and “Stage Sequence” **(B)** and to establish the conditional relationships between different kinds of information sets (Gunendran and Young, 2008) such as the influential semantics between the class “Stage Sequence” **(B)** and “Setup Sequence” **(C)**.

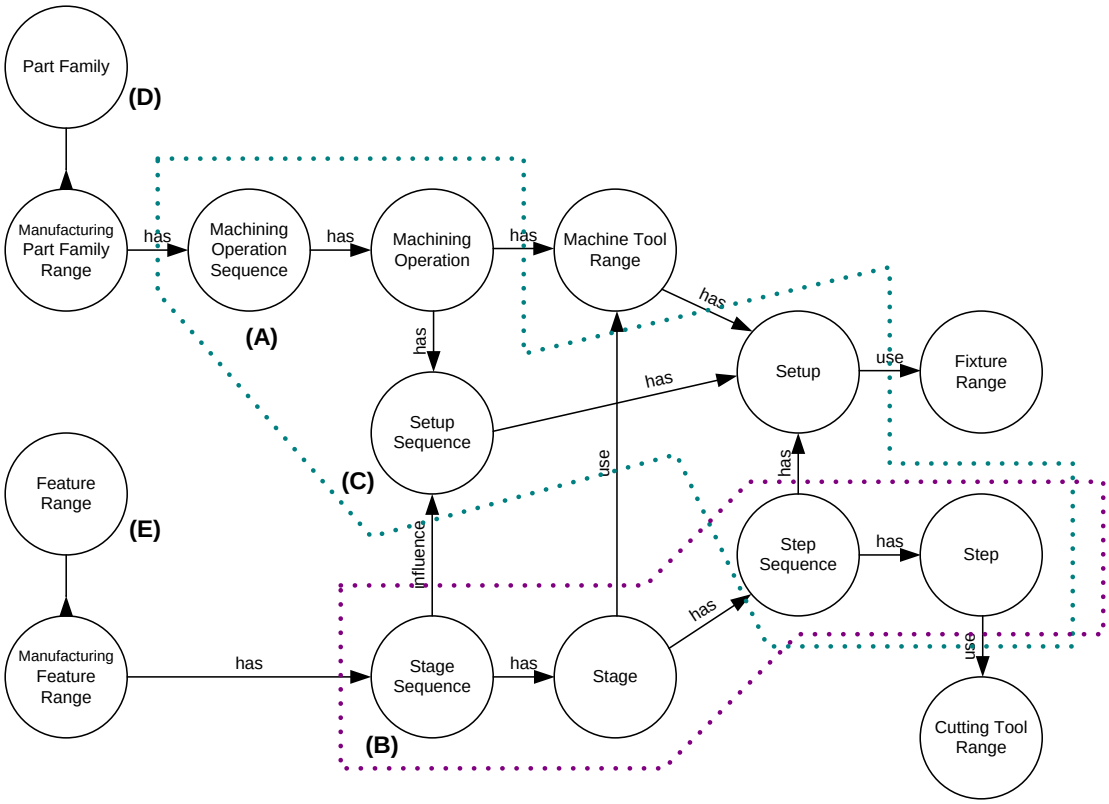


Figure 9-1 The Need for Capturing Relationship Semantics between Part Families and Features (Adapted from Gunendran and Young (2008))

In the context of the Foundation Layer, visible associations can be made between the foundation class “Artifact” and “Part Family” **(D)**. Moreover, the foundation class “Feature” neatly maps to the “Feature Range” **(E)** concept, thereby implying that the Foundation Layer is able to support extensions to accommodate part family semantics as well. It should not be forgotten, however, that the proposed heavyweight manufacturing ontological foundation in this work has focused on a restricted set of product viewpoints. Therefore, from a product lifecycle perspective, the expansion of foundation semantics should also be attuned to the representation of core operation, service and disposal semantics across system boundaries.

9.2.4 Knowledge Bases

The current approach taken during the proposal and implementation of the framework has witnessed the interoperation at the instance level of domain models, i.e. at the KB level, between systems that use the same type of KB.

For example, in the various test cases, the Object Management System (OMS) KB had been deployed, and reconciliation has taken place between OMSs of the same sort (with their reasoning engine all based on Java SQL). An issue is likely to emerge in the situation that different types of KBs are developed from the same or different domain ontologies. This understanding is illustrated in Figure 9-2 at the KB level **(F)**.

An initial concern is linked to the interoperation of multiple KBs that have been based on the same domain ontology (see label **(G)** on Figure 9-2). This is because different KBs naturally imply different applied computational principles. This issue is further aggravated when different KBs, coming from heterogeneous domain ontologies require interoperation **(H)**. It is, therefore, necessary to explore the related implications in more detail, as the mentioned situation is bound to happen in supply chain premises and collaborative product development. A possible direction in order to tackle similar problems would require a solid understanding of the software technologies and platform independent and specific structures innate to the various KBs that are being deployed, and that need to interoperate.

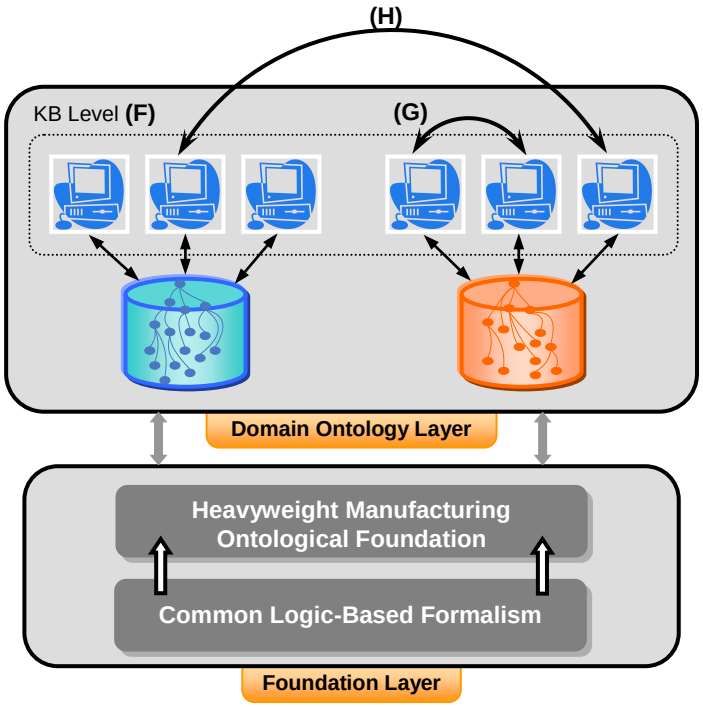


Figure 9-2 Developing Multiple KBs from the Same Domain Ontologies

9.2.5 Knowledge Sharing

In concurrent engineering and collaborative supply chain premises, knowledge sharing remains a relatively delicate aspect as far as inter-system interoperability is concerned. In many situations, due to data protection agreements, intellectual property rights, trust and security issues linked to proprietary information, the sharing of knowledge across domain ontologies and their related KBs may not always be a straightforward task.

Specifically for this purpose, the “simple merging process” explored in the Semantic Reconciliation Layer may not provide an optimal ontology mapping process. Two possible approaches could be applied in order to remedy the problem. The first involves keeping different domain ontologies and KBs in their distinct OMSs so that full control on sensitive ontological content is maintained. Then using Application Programming Interfaces (APIs), semantic mapping concepts would be applied to relevant portions of the domain models to be reconciled. The understanding is pictured in Figure 9-3, where for example, the two domain models “Machining Hole Feature Ontology A” and “Machining Hole Feature Ontology B” have remained distinct to their OMSs. Protection on appropriate ontological content would be supported in each model and semantic mapping concepts would interface only with the allowable cross-domain arguments for reconciliation.

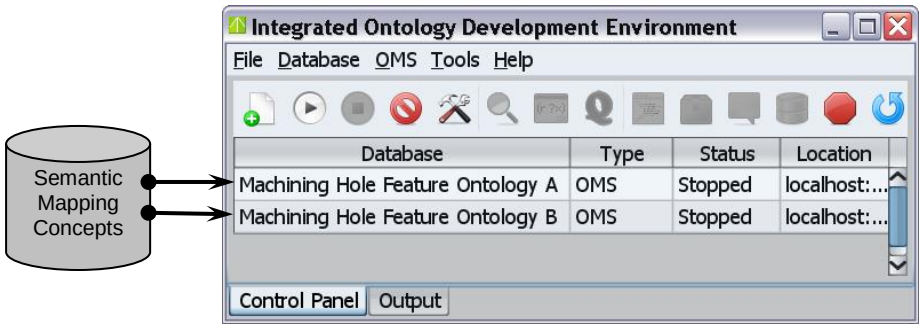


Figure 9-3 Interfacing Semantic Mapping Concepts to Domain Models without Undergoing the Simple Merging Process

The second way of ensuring that only the relevant ontological content and KB objects are reconciled between two domains, is to prune sensitive semantic

structures prior to undergoing the simple merging process. In this way only specific portions of the domain models would undergo reconciliation and knowledge sharing.

At present, the Interoperability Evaluation Layer supports useful methods for discovering and interpreting cross-domain correspondences. However, because the interoperability discovery process is dependent on a view of what is to be reconciled between domains in the first place, there is a need to include a way for reporting established cross-domain correspondences. A possible way for so doing would be to support the compilation of an evaluation report, post-interoperability evaluation and verification at the fourth level of the SMIF. On the other hand, the interpretation of cross-domain semantic mapping concepts would prove more effective if accompanied by diagrams in the ontological environment itself. Unfortunately, the current status of the IODE ontological environment does not allow pictures nor hyperlinks to pictures to be referenced in the informal remarks for interpreting semantic mapping concepts.

9.2.6 Positioning of the Framework

The literature review in Chapter 2 has identified the importance of positioning the concepts proposed in this research according to the Model Driven Architecture (MDA) and Model Driven Interoperability (MDI). Based on an understanding of MDA and MDI related to the various concepts developed in the SMIF, Figure 9-4 depicts the relevant MDI view on the investigated ontology-based framework for supporting semantic interoperability in product design and manufacture.

The diagram first shows that the investigated requirements for supporting semantic interoperability in product design and manufacture (Chapter 3) fall at the CIM level **(I)**. This is because the strategic nature of the requirements remains at a high-level for identifying the intended expectations of the developed framework.

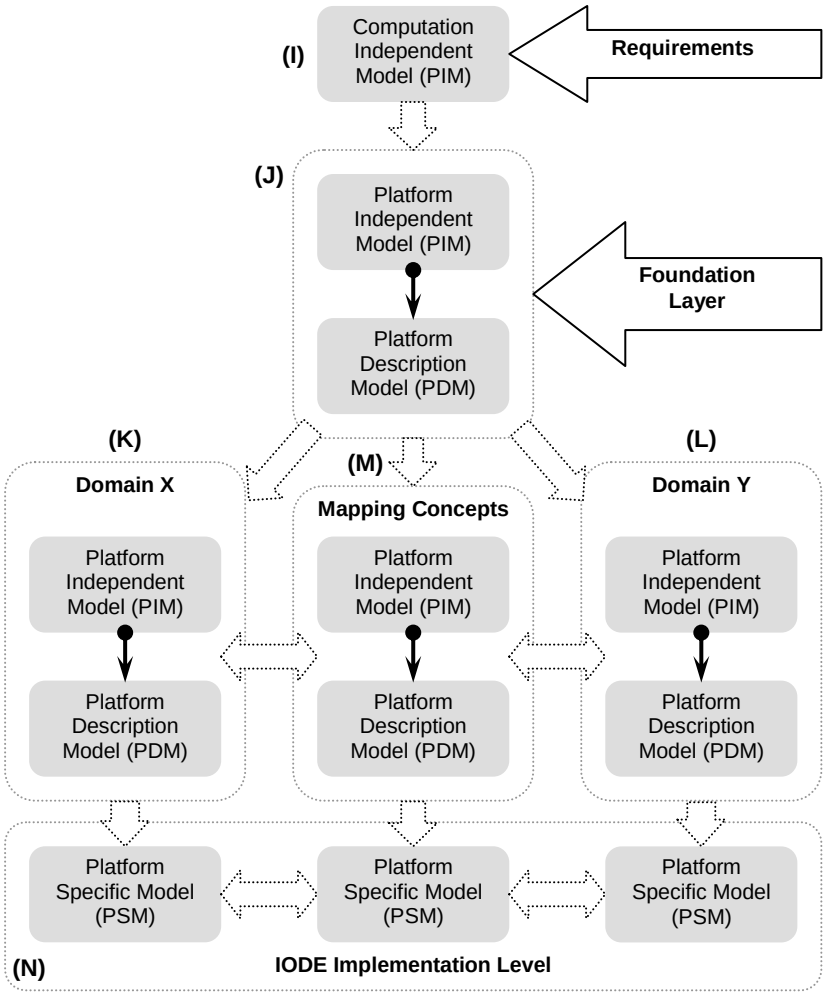


Figure 9-4 Model Driven Interoperability View on the Research Framework

The development of the Foundation Layer can be established at the PIM level of the MDA **(J)**. It is evident, from the ontology development methodology adopted, that IDEF5 schematics used to model the fundamental semantics of the heavyweight manufacturing ontological foundation provide a platform-independent way of representing ontological content. However, because KFL-based semantics have been added to the Foundation Layer, this implies that the PIM level is accompanied by a Platform Description Model (PDM). A PDM is used to specify the architecture for implementation and relevant technologies being harnessed. In this case, the PDM occurs as the consequence of the dependence of KFL on the configuration of the IODE implementation platform. Had the Common Logic Interchange Format (CLIF) been purely used, this would have implied that the Foundation Layer would have resided at a standalone PIM level without the related PDM.

The combination of the PIM and PDM for the Foundation Layer has then been exploited to develop other PIMs and PDMs, during specialisation into domain models such as two virtual ontologies “Domain X” **(K)** and “Domain Y” **(L)**. Semantic mapping concepts **(M)** constitute the interoperability models for reconciling between pairs of domain models. The Foundation Layer, domain models and models for semantic mapping concepts are driven to the PSM level **(N)** during implementation in IODE. One important observation to be made is that the various models have been implemented under the same implementation environment (i.e. IODE), which to some extent contributes to the ability to interoperate at the PSM level. Different platform-specific models using different implementation environments would lead to the heterogeneous KB issue identified previously in section 9.2.4.

Based on an understanding of the SMIF and its implementation, it also becomes possible to position the framework in relation to other interoperability frameworks. The SMIF, as opposed to interoperability frameworks such as IDEAS interoperability framework (Chen *et al*, 2004), the Framework for Enterprise Interoperability (CEN/ISO 11354, 2008), the Zachman Framework (The Zachman Framework Website, 2009) and The Open Group Architecture Framework (TOGAF) (TOGAF Website, 2009), does not aim at providing a novel way of redefining general concepts for interoperability. This is because, the SMIF remains focused at the issue of semantic interoperability in design and manufacture. Figure 9-5 positions the main concepts of the SMIF, using the Framework for Enterprise Interoperability as a benchmark.

As can be seen in the picture, the main concepts explored in the SMIF fall under three main blocks as a result of (1) considerations for unified processes using PSL, (2) considerations for unified entity information semantics at the unified (product) data level and (3) the harnessing of appropriate semantic technologies to support integrated technological advances.

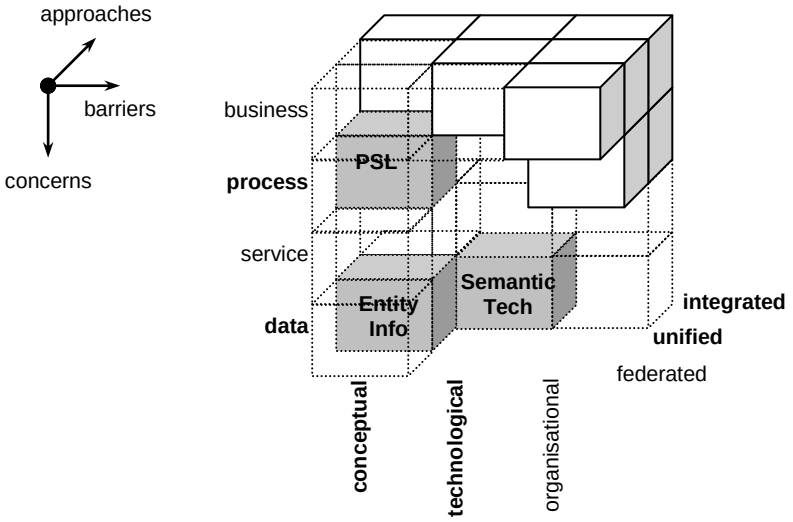


Figure 9-5 Positioning the Key SMIF Concepts in the Framework for Enterprise Interoperability

Compared to other similar approaches attuned specifically to semantic interoperability such as the eCOIN framework (Firat *et al*, 2007), the approach explored by Patil *et al* (2005) and that of Gupta and Gurumoorthy (2008), the SMIF has contributed to the identification and application of more formal ways (heavyweight Common Logic-driven) for capturing knowledge, starting at a low level of abstraction, including the geometry, dimensional and process sequencing semantics. In addition to this, more effective methods have been investigated in order to achieve meaningful interoperable knowledge sharing between domain models during their reconciliation. The interpretation of the interoperable knowledge, backed by tractable reasoning, overtakes the simple mapping relations used, for instance, in OWL-based reconciliation and reasoning.

More recently, an initial proposal for a future SC4 architecture has been realised (Leal *et al*, 2009). Interestingly, the structure of this future architecture bears some striking similarities to the fundamental concepts explored in the SMIF. The underlying understanding behind this proposed architecture is portrayed in Figure 9-6. The architecture is composed of:

- Natural language terms and their definitions related to the concepts within the SC4 standards **(O)**.
- The ontology-driven formalised representation of the more general concepts covered by the SC4 standards, referred to as “resource parts” **(P)**. This is analogous to the semantics of core concepts in the Foundation Layer of the SMIF.
- The ontology-driven formalised representation of the more discipline-specific concepts covered by the SC4 standards, referred to as “domain extensions” **(Q)**. In the context of the SMIF, this understanding is reflected in the Domain Ontology Layer.
- A set of implementation technology solutions for specific use cases that are mapped to and from the elements in the formal ontological representations, examples of which are called “constrained exchange subsets” **(R)** and “web service definitions” **(S)** (Leal *et al*, 2009). When viewed from the SMIF approach, this may involve the development of multiple domain KBs from domain models, thereby resulting in the plural nature of PSMs. This aspect, however, has not been probed into in the current research framework, but the necessary implications have been identified in section 9.2.4.
- Appropriately-formalised mappings and/or references between the terms and definitions, ontology-driven representations and implementation technology solutions **(T)**. In the SMIF, the definition of semantic mapping concepts to support interoperable knowledge sharing, provides a useful means of performing the required mappings.

It is further to be noted that during the proposal of the above-mentioned SC4 architecture, references have been made to possible modelling languages such as OWL, CL, UML, XML Schema and the Web Service Description Language (WSDL). This clearly illustrates that Common Logic-based knowledge representation formalisms have been acknowledged as forming part of the category of ontological formalisms that possess attractive capabilities to address the requirements of future standards-based integration architectures.

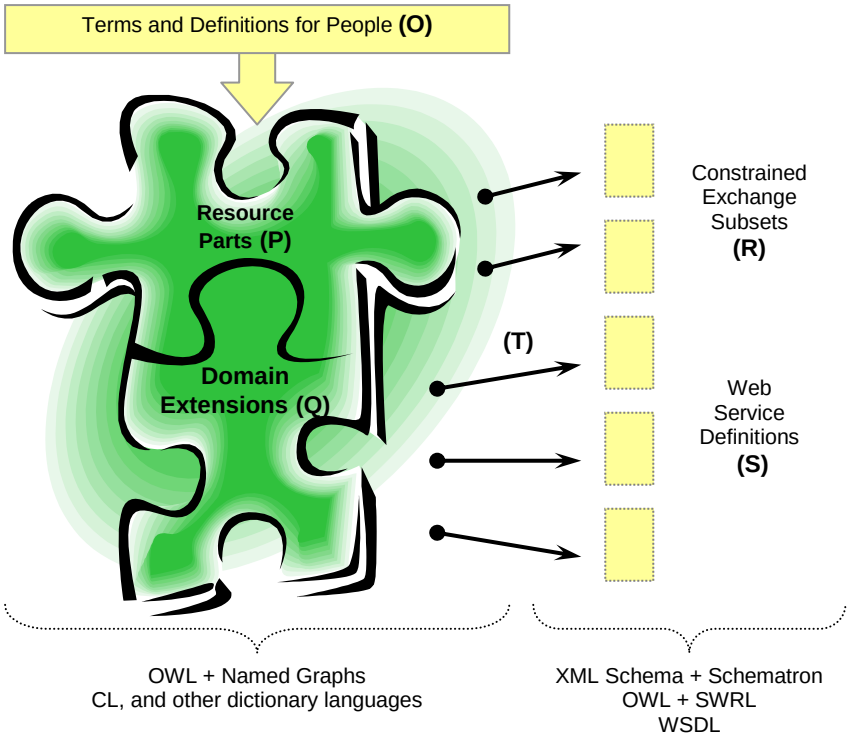


Figure 9-6 Future SC4 Architecture Based on Ontology Representations of Engineering Data (Adapted from Leal et al (2009))

9.2.7 Potential Industrial Applications

There exist wide-ranging potential applications of the proposed Semantic Manufacturing Interoperability Framework (SMIF) in manufacturing enterprises. At present, the relevance of ontologies in industry is obvious as several enterprises like DaimlerChrysler are, for example, adopting ontology-driven methods to support a range of design activities (Lukibanov, 2005).

Figure 9-7 illustrates a possible configuration of the SMIF with respect to its interactions with elements of wider design and manufacturing systems in PLM, within individual manufacturing enterprises. Domain ontologies that derive from the Foundation Layer of the SMIF could be interfaced with CAE applications, for example, a CAD environment could be linked to a domain ontology that fully captures the semantics in solid modelling (see label **(U)** on Figure 9-7). The KB related to the domain ontology would be used as a repository for storing, accessing, updating and creating parts information.

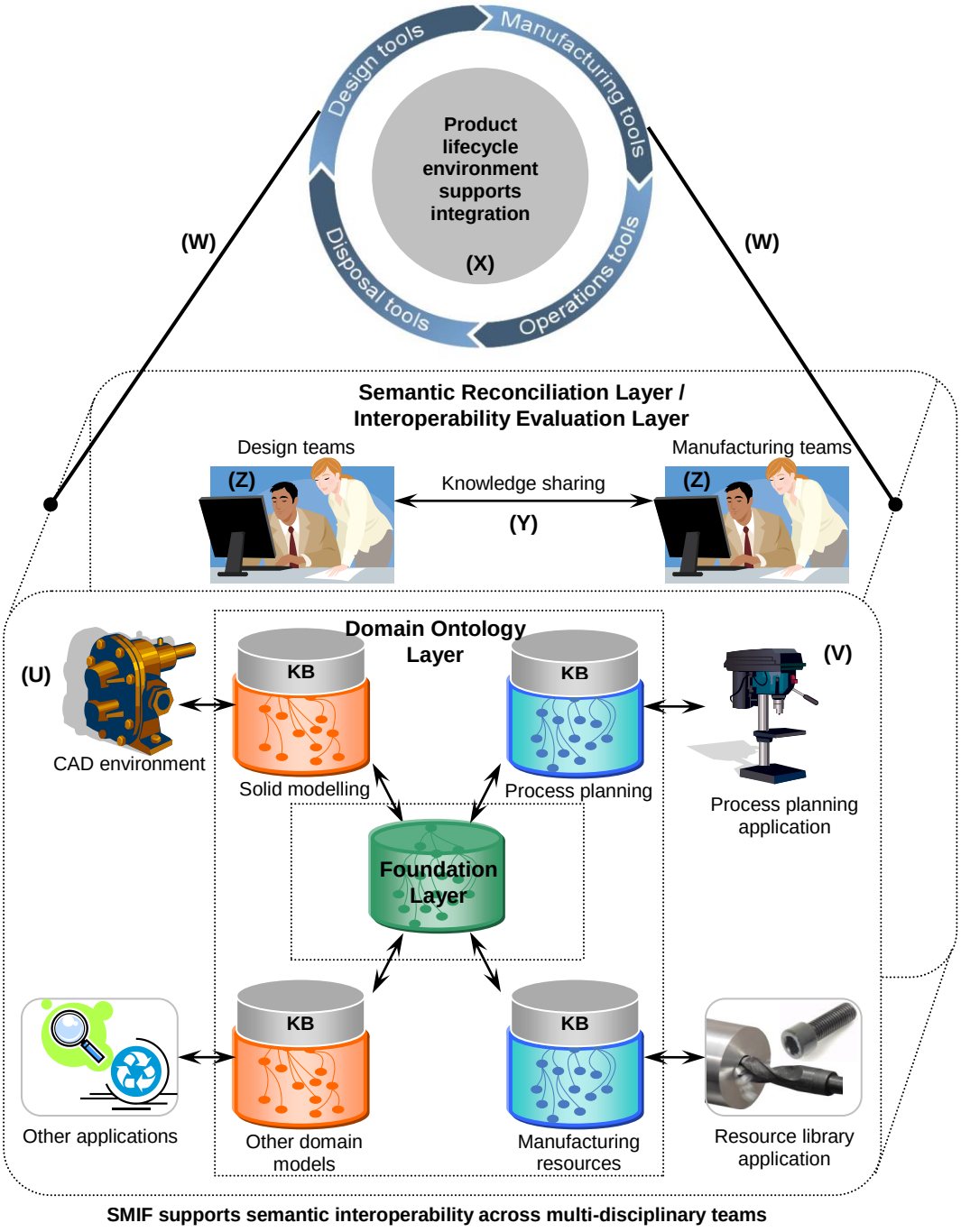


Figure 9-7 Visualising the SMIF within Integrated and Interoperability-Driven PLM

In addition, rigorous heavyweight semantics from PSL could be exploited towards monitoring shop-floor activities such as automated machining and assembly sequences following process planning **(V)**. This is one example 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. Extensions to the framework aided through the set up

of interfaces with PLM environments **(W)** would potentially help support not only the integration **(X)**, but also the level of semantic interoperability required in effectively sharing knowledge across multi-disciplinary teams involved in collaborative PLM **(Y)**.

The SMIF approach could further be integrated as part of a knowledge management initiative for building large repositories of design and manufacture knowledge. Knowledge would be accessed via shared ontologies and mapping mechanisms would be present for comparing various information sources for effectively clarifying intent and sharing knowledge. The ability to create and reuse meaningful best practice knowledge in computational form could also be supported, as this constitutes a powerful asset for the utilisation of historical as well as future information gathered during the continuous evolution of company structures. Additionally, Web-based company applications could be linked to the appropriate levels of the SMIF to support information searches and user-defined queries **(Z)**.

It is to be noted that unless appropriate user interfaces are supported for building such queries, adequate training of users would be required for interacting with an ontological platform such as IODE. In terms of performance, the use of IODE Object Management Systems (OMSs) would not provide a scalable approach to the creation of large KBs. This is because an IODE OMS is limited to the number of knowledge elements stored. Hence, this clearly implies that for meeting the needs of large design and manufacture KBs, industry-robust KBs would be required. In addition to this, important concerns are likely to remain notably in terms of the costs involved in carrying out technology change procedures and the general acceptance of the approach.

9.3 Conclusions

The Semantic Manufacturing Interoperability Framework (SMIF) investigated in this work has supported a further step towards the overall improvement of interoperability for effectively sharing knowledge across decision support systems. It has been possible through the proposal, thorough investigation and relevant testing of the framework, to achieve the aim of progressing knowledge on ontology-based approaches to support semantic interoperability applied to the field of product design and manufacture.

Sections 9.2 and 9.4 of this chapter document the relevant proposition for extensions and modifications to the SMIF in order to support future work, thereby meeting the sixth and final objective of this work (see Chapter 1 section 1.3.1). Furthermore, the various objectives set at the beginning of the thesis have been met (refer to cross-references between the objectives in Chapter 1 section 1.3.1 to the occurrence of their achievement at various points throughout the thesis). This clearly suggests that the research methodology undertaken in this work has successfully supported the achievement of the aim of this research.

Figure 9-8 depicts a diagram that summarises the key aspects of the SMIF with respect to the relevance of automation at various stages namely: ontology development, semantic reconciliation and interoperable knowledge discovery. It is clear from the concepts explored in this work that the process of ontology development is semi-automatic, especially since the knowledge engineer and the ontological environment are the prime agents in ontology building and deployment. Moreover, the first two stages of the semantic reconciliation phase, notably that of context adjustment and the simple merging of domain models, are semi-automatic processes.

Context adjustment of domain models, as witnessed in some of the test cases (see test cases 2 and 4 in Chapter 8) is essentially a manual process. The simple merging process as part of semantic reconciliation is a semi-automatic stage as it requires the user choosing the necessary ontology and instance

files to be processed by the ontological environment. The semantic alignment process is entirely automated, as a result of logic-based definitions for semantic mapping concepts, which automatically align cross-domain content. The final phase related to interoperable knowledge discovery is semi-automatic as it relies on appropriate user actions and interactions with interfaces for creating and running queries as well as for browsing the results of queries.

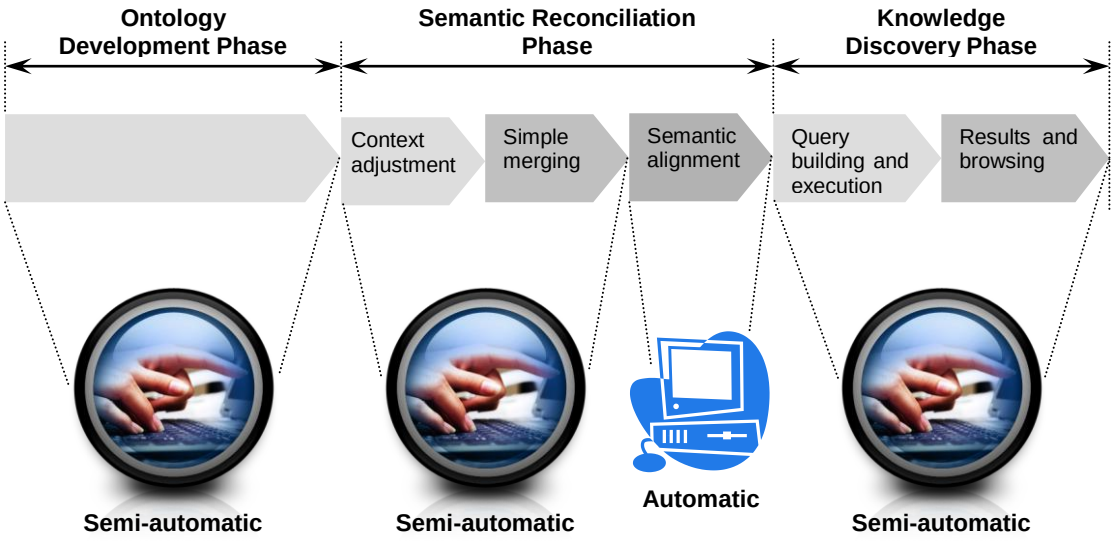


Figure 9-8 The Relevance of Automation in the SMIF

This view on the current state of automation of the main phases involved in the SMIF illustrates that there exist potentials for enhancing the performance of the framework by automating relevant processes. The ontology development and knowledge discovery phases are very likely to remain semi-automatic as user interactions are unavoidable. However, additional tools and methods need to be integrated with the SMIF implementation environment in order to support automatic context adjustment and simple merging.

In the knowledge discovery phase, it has been witnessed that the query tool in IODE supports the rapid processing of complex Common Logic-based queries performed on a single workstation. However, in rare cases when queries are not well-formed by the user, this may result in excessive memory consumption in trying to retrieve a possible answer to an inaccurate query. In other situations, it may be impossible to reach the result of a query based on

deductive reasoning especially if certain facts do not exist in a KB. This consequently implies that in industrial settings, adequate user training would be required in order to interact with the various elements of the SMIF.

The issue of processing time is likely to have a repercussion during collaborative activities between different agents. Therefore, it is still important to understand the extent to which the processing time remains beneficial across a collaborative environment. It is possible that there would be a need for optimisation which would result in higher performance, thereby enabling multiple queries to be performed from various workstations, whose query tools are simultaneously linked to the same KB found on a server. Figure 9-9 identifies a possible configuration of a server based system for querying against a KB. The potentials of Graphical Processing Units (GPUs) may be required for their high computing power, in order to compensate for the lower speed of Central Processing Units (CPUs) against GPUs.

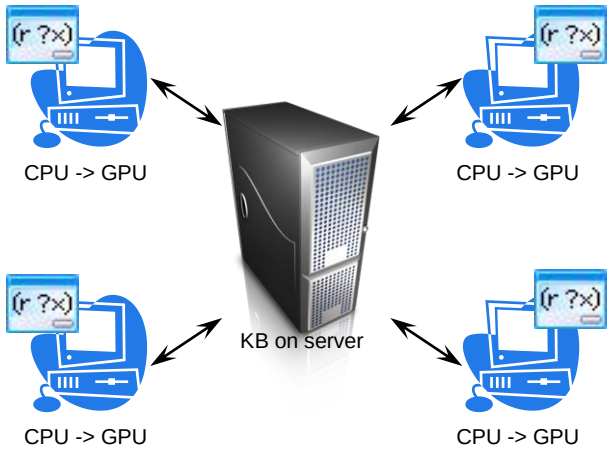


Figure 9-9 A Server-Based Configuration for Multiple Interacting Workstations

Based on the observations made during the discussions section of this chapter, a number of concerns have been depicted. The primary observation is that framework extensions are required. These extensions should accommodate further types of generic intuitions towards more defined product lifecycle semantics, altogether captured within the heavyweight manufacturing ontological foundation. Moreover, there is a need for refining the Domain Ontology Layer to include a clear demarcation between domain ontologies over which several platform-specific KBs could be established.

Hence, supporting the continuous evolution of the SMIF would help foster a leap towards intelligent automated paper-free knowledge sharing. The overall benefits would promote the enhancement of knowledge management strategies. On the whole, a progression of the framework shall continue to provide a competitive edge related to (1) the use of effective foundation ontology approaches to support knowledge capture and (2) the application of semantic methods for knowledge sharing across decision support systems in product design and manufacture.

9.4 Recommendations for Future Work

The discussions section of this chapter has helped orientate appropriate attention onto relevant areas where future work could apply. First of all, it would be highly desirable to explore an extended heavyweight manufacturing ontological foundation which would capture more complicated feature ranges such as pockets, splines, complex closed profiles, etc. Moreover, to enable the unambiguous definition of manufacturing features, the semantics of part families would deserve attention. An engaging starting point would consist of a mapping of the high-level diagram proposed by Gunendran and Young (2008) (see Figure 9-1) to foundation semantics, or vice versa.

Future work should also concentrate on identifying the different nuances within the Foundation Layer. At present, the Foundation Layer consists of two blocks namely the Common Logic-based ontological formalism over which the heavyweight manufacturing ontological foundation is established. However, this heavyweight foundation, during expansion and implementation would inevitably lead to different levels of conceptualisations within a single foundation. Figure 9-10 summarises this understanding and exemplifies the idea behind having different harmonised nuances within a single foundation.

From the figure, it becomes clear that some meta-ontology is bound to exist, such as the Ontology Works ULO, at the bottommost section of the Foundation Layer, over which the main ontological formalism is built. Generic intuitions need to be developed to capture broad concepts that cut across

several more specific product lifecycle intuitions. Such extensible generic concepts may include (1) the Process Specification Language (PSL) as a basis for describing processes of various sorts, (2) the Core Product Model (CPM) for capturing generic product model information, (3) generic models such as the model of measures (Ontology Works Inc., 2009) implementing NIST's publication on the International System of Units (SI) (Taylor and Thompson, 2008), (4) models of events featuring the participation semantics of objects in relation to events and (5) the temporal model based on the Temporal Interval Calculus of J.F. Allen (Ontology Works Inc., 2009).

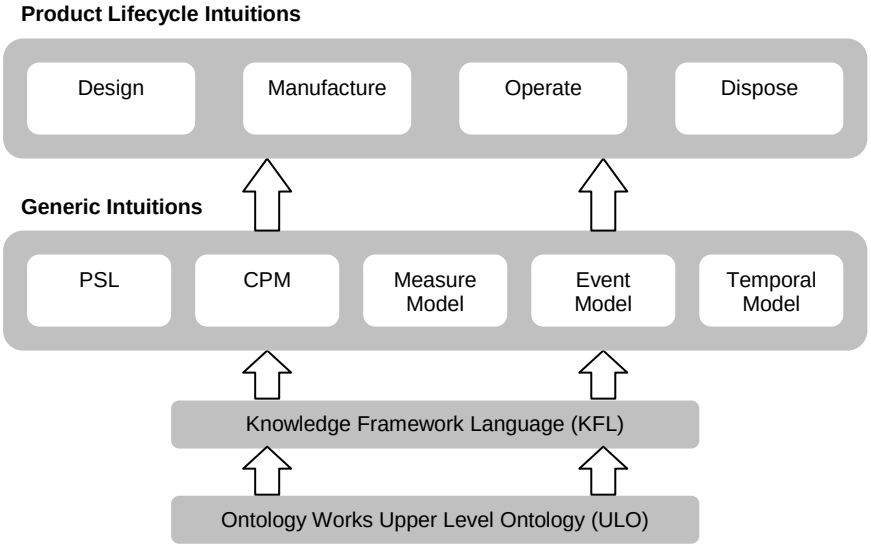


Figure 9-10 Expansion of Levels of Conceptualisation within a Single Product Lifecycle Foundation

Based on the view that different foundation levels of conceptualisation would arise as a result of an expansion of the Foundation Layer, this would necessarily imply that the methods for facilitating the reconciliation and verification across different domain extensions would also require evolvement. On the other hand, to further explore the application-oriented benefits of using heavyweight ontological approaches, it would be a challenging task to experience with the programming of application interfaces between, for example, CAD/CAM software and the KBs supported by domain ontologies developed from the Foundation Layer. In this way a concrete opportunity would arise to test the true potentials of ICs for articulating user inputs,

providing intelligent suggestions and preventing unwanted actions from being committed during the use of ontology-driven CAD/CAM environments.

Finally, there is still a need to conduct test cases, applied to the SMIF, based on comprehensive industrial scenarios. These scenarios would bring considerable value to the applicability of the proposed framework within an industrial setting. Possible case studies originating, for example, from the aerospace and automotive industries would help support the breadth of product lifecycle concepts required for further testing the Semantic Manufacturing Interoperability Framework.

Publications

- Chungoora, N. and Young, R.I.M., 2008a. Semantic interoperability requirements for manufacturing knowledge sharing. In: Mertins, K., Ruggaber, R., Popplewell, K. and Xu, X., eds. Enterprise interoperability III: new challenges and industrial approaches. pp. 411-422. London, UK: Springer-Verlag.
- Chungoora, N. and Young, R.I.M., 2008b. Ontology mapping to support semantic interoperability in product design and manufacture. In: *Proceedings of the 1st International Workshop on Model Driven Interoperability for Sustainable Information Systems (MDISIS'08) in Conjunction with the CAiSE'08 Conference*. 340, pp. 1-15. Montpellier, France. Available at:
  http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-340/
- Young, R.I.M., Gunendran, G., Chungoora, N., Harding, J. and Case, K., 2009. Enabling interoperable manufacturing knowledge sharing in PLM. In: *Proceedings of the 6th International Product Lifecycle Management Conference*. Bath, UK: University of Bath.
- Chungoora, N., Canciglieri, O.J. and Young, R.I.M. Semantic manufacturing knowledge sharing: a case study based on lightweight and heavyweight ontological approaches. *Computers in Industry*. [submitted for review].
- Chungoora, N. and Young, R.I.M. The configuration of design and manufacture knowledge models from a heavyweight ontological foundation. *International Journal of Production Research.* [submitted for review].
- Chungoora, N. and Young, R.I.M. A framework to support semantic interoperability in product design and manufacture. Submitted for review in: *20th CIRP Design Conference*.

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, pp. 389-415.

Adobe Photoshop. [Online] Available at: **http://www.adobe.com**

Adobe Flash. [Online] Available at: **http://www.adobe.com**

Advanced and Innovative Models and Tools for the development of Semantic-based systems for Handling, Acquiring and Processing knowledge Embedded in multidimensional digital objects. [Online] Available at: **http://www.aimatshape.net/**

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

Al-Ashaab, A.H., 1994. A manufacturing model to capture injection moulding process capabilities to support design for manufacture. Ph.D. Loughborough, UK: Loughborough University.

Al-Ashaab, A.H., Rodriguez, K., Molina, A., Cardenas, M., Aca, J., Saeed, M. and Abdalla, H. 2003. Internet-based collaborative design for an injection-moulding system. *Concurrent Engineering Research And Applications*, 11, pp. 289-299.

AMICE, 1993. *CIMOSA – Open System Architecture for CIM*. 2nd ed. Berlin, Germany: Springer-Verlag.

Anderl, R., 1997. Trends in product modelling. In: *Proceedings of the 11th International Conference on Engineering Design*. pp. 113-120. Tampere University of Technology.

Ang, D.S., 1998. Identifications of part families and bottleneck parts in cellular manufacturing. *Industrial Management and Data Systems*, 98(1), pp. 3-7.

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.

Bach, T.L., Dieng-Kuntz, R. and Gandon, F., 2004. On ontology matching problems for building a corporate semantic web in a multi-communities organization. In: *Proceedings of the 6th International Conference on Enterprise Information Systems*. Porto, Portugal.

Balogun, O., Hawisa, H. and Tannock, J., 2004. Knowledge management for manufacturing - the product and process database. *Journal of Manufacturing Technology Management*, 15(7), pp. 575-584.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F. and Stein, L.A., 2004. OWL Web Ontology Language Reference. [Online] W3C Recommendation. World Wide Web Consortium. Available at: **http://www.w3.org/TR/owl/ref/**

Bock, C., 2006. Interprocess Communication in the Process Specification Language. [Online] NISTIR 7348. NIST: Gaithersburg, MD, USA. Available at: **www.mel.nist.gov/psl/pubs.html**

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

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.

Bourey, J.-P., 2007. *Model Driven Interoperability and Service-Oriented Architecture*. PowerPoint Presentation. IMS Seminar: Zurich, Switzerland.

Bourey, J.P., Grangel, R., Doumeingts, G. and Berre, A., 2006. INTEROP NoE - Deliverable DTG2.2 - Report on Model Interoperability.

Brunetti, G. and Golob, B., 2000. A feature-based approach towards an integrated product model including conceptual design information. *Computer Aided Design*. 32(14), pp. 877-887.

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

Canciglieri, O.J., 1999. Product model based translation mechanism to support multiple viewpoints in the design for manufacture of injection moulded products. Ph.D. Loughborough, UK: Loughborough University.

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), pp. 1565-1586.

CEN/ISO 11354, 2008. Requirements for establishing manufacturing enterprise process interoperability - Part 1 - Framework for Enterprise Interoperability.

Chan, F.T.S., Lau, K.W., Chan, P.L.Y. and Choy, K.L., 2006. Two-stage approach for machine-part grouping and cell layout problems. *Robotics and Computer-Integrated Manufacturing*. 22(3), pp. 217-238.

Chandra, C. and Kamrani, A.K., 2003. Knowledge management for consumer-focussed product design. *Journal of Intelligent Manufacturing*. 14, pp. 557-580.

Chang, X. and Terpenny, J., 2009. Ontology-based data integration and decision support for product e-design. *Robotics and Computer-Integrated Manufacturing*. 25, pp. 863-870.

Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D. and Rice, J.P., 1998. OKBC: a programmatic foundation for knowledge base interoperability. In: *Proceedings of the 15th National Conference on Artificial Intelligence AAAI-98 and the 10th Conference on Innovative Applications of Artificial Intelligence IAAI-98*, pp. 600-607.

Chen, D. and Doumeingts, G., 1996. The GRAI-GIM reference model, architecture and methodology. In: Bernus, P. et al, eds. *Architectures for enterprise integration*. London: Chapman and Hall.

Chen, D., Doumeingts, G. and Vernadat, F., 2008. Architectures for enterprise integration and interoperability: past, present and future. *Computers in Industry*. Vol. 59, pp. 647-659.

Chen, D., Knothe, T. and Zelm, M., 2004. ATHENA integrated project and the mapping to International Standard ISO 15704. In: *Proceedings of the International Conference on Enterprise Integration Modelling Technology*, pp. 9-11. Valencia, Spain.

Chen, W. and Stuckenschmidt, H., 2008. Towards industrial strength knowledge bases for Product Lifecycle Management. In: *Proceedings of the 16th European Conference on Information System – Special Track on Semantic Web and Information Systems*.

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), pp. 307-328.

Chin, K.S., Zhao, Y. and Mok, C.K., 2002. STEP based multi-view integrated product modelling for concurrent engineering. *International Journal of Advanced Manufacturing Technology*. pp. 896-906.

Cho, J., Han, S. and Kim, H., 2006. Meta-ontology for automated information integration of parts libraries. *Computer Aided Design*. 38, pp. 713-725.

Chungoora, N. and Young, R.I.M., 2008a. Semantic interoperability requirements for manufacturing knowledge sharing. In: Mertins, K., Ruggaber, R., Popplewell, K. and Xu, X., eds. *Enterprise interoperability III: new challenges and industrial approaches*. pp. 411-422. London: Springer-Verlag.

Chungoora, N. and Young, R.I.M., 2008b. Ontology mapping to support semantic interoperability in product design and manufacture. In: *Proceedings of the 1st International Workshop on Model Driven Interoperability for Sustainable Information Systems (MDISIS'08) in Conjunction with the CAiSE'08 Conference*. 340, pp. 1-15. Montpellier, France. [Online] Available at: **http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-340/**

Cingil, I. and Dogac, A., 2001. An architecture for supply chain integration and automation on the internet. *Distributed and Parallel Databases*. 10, pp. 59-102.

Cochrane, S.D., 2006. Manufacturing knowledge verification in design support system. Ph.D. Loughborough, UK: Loughborough University.

Corcho, O., 2005. *A layered declarative approach to ontology translation with knowledge preservation*. Frontiers in Artificial Intelligence and Applications. 116. Netherlands: IOS Press.

Costa, C.A, Salvador, V.L, Meira, L.M., Rechden, G.F. and Koliver, C., 2007. Product ontology supporting information exchanging in global furniture industry. In: Goncalves, R.J. et al., eds. *Enterprise interoperability II: new challenges and approaches*. pp. 278-280. London, UK: Springer-Verlag.

Cutting-Decelle, A.F., Bourey, J.-P., Grangel, R. and Young, R.I.M., 2006. Ontology based communications through model driven tools - the MDA approach feasibility of the approach in urban engineering projects. In: *COST C21 1st Workshop on Ontologies for Urban Development – Interfacing Urban Information Systems*. Geneva, Switzerland.

Cutting-Decelle, A.F., Dubois, A.M. and Dubois, J.E., 2002. An information system for the building industries: a communication approach based on industrial components. *Data Science Journal*. 1(2), pp. 257-270.

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), pp. 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), pp. 396-410.

Delugach, H.S., 2005. *Common Logic in support of metadata and ontologies*. PowerPoint Presentation. Open Forum 2005 on Metadata Registries. [Online] Available at: **http://common-logic.org/docs/cl/Berlin_OpenForum_Delugach.ppt**

Didonet del Fabro, M., Albert, P., Bézevin, J. and Jouault, F., 2008. Industrial-strength rule interoperability using model driven engineering. Research Report Version 1. France: Institut de Recherche en Informatique et en Automatique, Rennes.

Doerr, M., Hunter, J. and Lagoze, C., 2003. Towards a core ontology for information integration. *Journal of Digital Information*. 4(1).

Elvesæter, B., Hahn, A., Berre, A.-J. and Neple, T., 2006. Towards an interoperability framework for model-driven development of software systems. In: Konstantas, D., Bourrières, J.-P., Léonard, M. and Boudjilida, N., eds. *Interoperability of enterprise software and applications*. pp. 409-420. London, UK: Springer-Verlag.

Euzenat, J., Mocan, A. and Scharffe, F., 2008. Ontology alignment: an ontology management perspective. In: Hepp, M., De Leenheer, P., de Moor, A. and Sure, Y., eds. O*ntology management - semantic web, semantic web services and business applications*. Boston, USA: Springer, SPVU.

Euzenat, J. and Shvaiko, P., 2007. *Ontology Matching*. Berlin Heidelberg: Springer-Verlag.

Euzenat, J. and Valtchev, P., 2004. Similarity-based ontology alignment in OWL-lite. In: *Proceedings of the 15th European Conference on Artificial Intelligence*. Valencia, Spain.

Farquhar, A., Fikes, R. and Rice, J., 1997. The Ontolingua server: a tool for collaborative ontology construction. *International Journal of Human Computer Studies*. 46(6), pp. 707-727.

Feng, S.C. and Song, E.Y., 2003. A manufacturing process information model for design and process planning integration. *Journal of Manufacturing Systems*. 22(1).

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, pp. 238-246.

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. USA: National Institute of Standards and Technology.

Fernández-Breis, J.T. and Martínez-Béjar, R., 2002. A cooperative framework for integrating ontologies. *International Journal of Human-Computer Studies*. 56, pp. 665-720.

Fernández-López, M. and Gómez-Pérez, A., 2002. Overviews and analysis of methodologies for building ontologies. *The Knowledge Engineering Review*. 17(2), pp. 129-156.

Fiorentini, X., Gambino, I., Liang, V.-C., Rachuri, S., Mani, M. and Bock, C., 2007. An ontology for assembly representation. NISTIR 7436. USA: National Institute of Standards and Technology.

Firat, A., Madnick, S., and Grosof, B., 2007. Contextual alignment of ontologies in the eCOIN semantic interoperability framework. *Information Technology Management*. 8, pp. 47-63.

Fowler, J., 1996. *STEP for data management, exchange and sharing*. Great Britain: Technology Appraisals.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A. and Schneider, L., 2002. Sweetening ontologies with DOLCE. In: *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*. LNCS 2473, pp. 166-182. Siguenza, Spain.

Genesereth, M.R. and Fikes, R.E., 1992. Knowledge Interchange Format. Version 3.0. Reference Manual. Technical Report Logic-92-1. California: Stanford University.

Giachetti, R.E., 1999. A standard manufacturing information model to support design for manufacturing in virtual enterprises. *Journal of Intelligent Manufacturing*. 10, pp. 49-60.

Gnägi, H.R., Morf, A. and Staub, P., 2006. Semantic interoperability through the definition of conceptual model transformations. In: *Proceedings of the 9th AGILE Conference on Geographic Information Science*. Visegrád, Hungary.

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.

Grenon, P., 2003. Nuts in BFO's nutshell: revisions to the bi-categorial axiomatisation of BFO. INFOMIS Reports. ISSN 1611-4019. Germany: University of Leipzig.

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

Gruber, T.R., 1993. A translation approach to portable ontology specification. *Knowledge Acquisition*. 5(2), pp. 199-220.

Gruninger, M. and Fox, M.S., 1994. An activity ontology for enterprise modelling. In: *Workshop on Enabling Technologies – Infrastructures for Collaborative Enterprises*. West Virginia University, USA.

Gruninger, M. and Kopena, J., 2005. Semantic integration through invariants. *AI Magazine*. 26(1), pp. 11-20.

Gu, P., 1994. A feature representation scheme for supporting integrated manufacturing. *Computers and Industrial Engineering*. 26(1), pp. 55-71.

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), pp. 326-338.

Gunendran, A.G., Young, R.I.M., Cutting-Decelle, A.F. and Bourey, J.P., 2007. Organising manufacturing information for engineering interoperability. In: *Proceedings of the Interoperability for Enterprise Software and Applications Conference*. Madeira Island, Portugal.

Gunendran, A.G. and Young, R.I.M., 2008. Methods for the capture and reuse of manufacturing best practice in Product Lifecycle Management. In: *Proceedings of the 5th International conference on Product Lifecycle Management*. Seoul, Korea.

Gupta, R.K. and Gurumoorthy, B., 2008. A feature-based framework for semantic interoperability of product models. *Journal of Mechanical Engineering*. 54(6), pp. 446-457.

Halpin, T., 1999. UML data models from an ORM perspective: Part 1 - 10. *Journal of Conceptual Modelling*.

Hameed, A., Preece, A. and Sleeman, D., 2004. Ontology reconciliation. In: Staab, S. and Studer, R., eds. *Handbook on Ontologies*. pp. 231-250. International Handbooks on Information Systems. Springer.

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

ISO 286-2, 1988. Tables of standard tolerance grades and limit deviations for holes and shafts.

ISO/IEC 10746-3, 1996. Information technology - open distributed processing - reference model: architecture.

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 10303-239, 2005. STEP - product data representation and exchange, application protocol: Product Life Cycle Support (PLCS).

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

Kalfoglou, Y. and Schorlemmer, M., 2002. Information-flow-based ontology mapping. In: *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA and ODBASE*. LNCS 2519, pp. 1132-1151. Springer.

Kalfoglou, Y. and Schorlemmer, M., 2003. Ontology mapping: the state of the art. *Knowledge Engineering Review*. 18(1), pp. 1-31.

Kalpakjian, S., 2001. *Manufacturing engineering and technology*. 4th ed. New Jersey, USA: Prentice Hall Inc.

Kendall, E.F., Frankel, D.S., Hayes, P.J. and McGuinness, D.L., 2004. Simple Common Logic: a constraint language for the ODM. In: *Proceedings of the 1st International Workshop on the Model-Driven Semantic Web*.

Kent, R.E., 2000. The information flow foundation for conceptual knowledge organization. In: *Proceedings of the 6th International Conference of the International Society for Knowledge Organization*. Toronto, Canada.

Kim, K.-Y., Manley, D.G. and Yang, H., 2006. Ontology-based assembly design and information sharing for collaborative product development. *Computer-Aided Design*. 38, pp. 1233-1250.

Kiryakov, A.K., Simov, K. Iv. and Dimitrov, M., 2001a. OntoMap: ontologies for lexical semantics. In: *Proceedings of the Euro Conference Recent Advances on Natural Language Processing*. pp. 142-148. Bulgaria.

Kiryakov, A.K., Simov, K. Iv. and Dimitrov, M., 2001b. OntoMap: portal for upper-level ontologies. In: *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems*. Ogunquit, Maine, USA.

Klein, M., 2001. Combining and relating ontologies: an analysis of problems and solutions. In: *Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Ontologies and Information Sharing*. pp.53-62. Seattle, USA.

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

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

Labrou, Y., 2002. Agents and ontologies for e-business. *The Knowledge Engineering Review*. 17(1), pp. 81-85.

Lassila, O. and Swick, R.R., 1999. Resource Description Framework (RDF) model and syntax specification. W3C Recommendation. World Wide Web Consortium. Cambridge, MA, USA.

Lazenberger, M., Sampson, J.J., Rester, M., Naudet, Y. and Latour, T., 2008. Visual ontology alignment for knowledge sharing and reuse. *Journal of Knowledge Management*. 12(6), pp. 102-120.

Leal, D., Price, D., Barnard Feeney, A. and Bock, C., 2009. Future SC4 architecture PWI overview and plan. In: *57th ISO TC 184 SC4 Plenary Meeting*. Approved Resolution by ISO TC 184 SC4. Parksville, Canada.

Lee, G., Eastman, C.M., Sacks, R. and Navathe, S.B., 2006. Grammatical rules for specifying information for automated product data modelling. *Advanced Engineering Informatics*. 20, pp. 155-170.

Leitão, P., Colombo, A. and Restivo, F., 2005. ADACOR - a collaborative production automation and control architecture. *IEEE Intelligent System*. 20(1), pp. 58-66.

Lenat, D.B. and Guha, R.V., 1990. *Building large knowledge-based systems: representation and inference in the Cyc project*. Boston, USA: Addison-Wesley.

Li, Y., Lu, Y., Liao, W. and Lin, Z., 2006. Representation and share of part feature information in web-based parts library. *Expert Systems with Application*. 31, pp. 697-704.

Lin, H.K. and Harding, J.A., 2007. A manufacturing engineering ontology model on the semantic web for inter-enterprise collaboration. *Computers in Industry*. 58(5), pp. 428-437.

Liping, Z., Guangyao, L., Yongquan, L. and Jing, S., 2007. Design of ontology mapping framework and improvement of similarity computation. *Journal of Systems Engineering and Electronics*. 18(3), pp. 641-645.

Liu, S., 2004. Manufacturing information and knowledge models to support global manufacturing coordination. Ph.D. Loughborough, UK: Loughborough University.

Liu, S. and Young, R.I.M., 2004. Utilising information and knowledge models to support global manufacturing coordination decisions. *International Journal of Computer Integrated Manufacturing*. 17(4), pp. 479-492.

Loss, L., Rabelo, R. J. and Pereira-Klen, A. A., 2005. For an intelligent decision support system for supply chain management. In: *Proceedings of the 38th CIRP International Seminar on Manufacturing Systems*. Brazil.

Lukibanov, O., 2005. Use of ontologies to support design activities at DaimlerChrysler. In: *Proceedings of the International Protégé Conference*. Madrid, Spain.

Lyons, J., 1977. *Semantics*. 1. Cambridge, Great Britain: Cambridge University Press.

Ma, Y.-S., Britton, G.A., Tor, S.B. and Jin, L.Y., 2007. Associative assembly design features - concept, implementation and application. *International Journal of Advanced Manufacturing Technology*. 32, pp. 434-444.

Madhavan, J., Bernstein, P.A., Domingos, P. and Halevy, A., 2002. Representing and reasoning about mappings between domain models. In: *Proceedings of the 18th National Conference on Artificial Intelligence*. Edmonton, Alberta, Canada.

Maedche, A. and Staab, S., 2000. Semi-automatic engineering of ontologies from texts. In: *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering*. pp. 231-239. Chicago, IL, USA.

Maedche, A., Motik, B., Silva, N. and Volz, R., 2002. A MApping FRAmework for distributed ontologies. In: *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, Ontologies and the Semantic Web*. LNCS 2473, pp. 235-250.

Markovits, H., 2004. The development of deductive reasoning. In: Sternberg, R.J. and Leighton, J.P., eds. *The Nature of Reasoning*. pp. 313-338. Cambridge, UK: Cambridge University Press.

Martino, T.D. and Giannini, F., 1998. Feature-based product modelling in concurrent engineering. In: *Proceedings of the 10th International Conference on Information Technology in the Product Realisation Process*. Trento, Italy.

Mäs, S., Wang, F. and Reinhardt, W., 2005. Using ontologies for integrity constraint definition. In: *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.

McGuinness, D., Fikes, R., Rice, J. and Wilder, S., 2000. An environment for merging and testing large ontologies. In: *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*. Colorado, USA.

MDA GUIDE, 2003. *MDA Guide*. Version 1.0.1. Document Number: omg/2003-06-01.

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*. pp. v-vi. London, UK: Springer-Verlag.

Microsoft Office FrontPage. [Online] Available at:
http://office.microsoft.com/en-gb/frontpage/default.aspx

Microsoft Office Visio. [Online] Available at:
http://office.microsoft.com/en-gb/visio/default.aspx

Miller, G.A., 1995. WordNet: a lexical database for English. *Communications of the ACM*. 3(4), pp. 39-41.

Mitra, P. and Wiederhold, G., 2002. Resolving terminological heterogeneity in ontologies. In: *Proceedings of the Workshop on Ontologies and Semantic Interoperability at the 15th European Conference on Artificial Intelligence*. pp. 45-50. Lyon, France.

Mitra., P., Noy, N.F. and Jaiswal, A.R., 2004. OMEN: a probabilistic ontology mapping tool. In: *Workshop on Meaning Coordination and Negotiation at the 3rd International Conference on the Semantic Web*. Hiroshima, Japan.

Mizoguchi, R., Vanwelkenhuysen, J. and Ikeda, M., 1995. Task ontology for reuse of problem solving knowledge. In: Mars, N., ed. *Towards very large knowledge bases: knowledge building and knowledge sharing*. pp. 46-57. Amsterdam, Netherlands: IOS Press.

Moalla, N., Ouzrout, Y., Neubert, G. and Bouras, A., 2008. Model-driven interoperability to enhance product data quality. In: *Proceedings of the 1st International Workshop on Model Driven Interoperability for Sustainable Information Systems (MDISIS'08) in Conjunction with the CAiSE'08 Conference*. 340, pp. 121-133. Montpellier, France. [Online] Available at: **http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-340/**

Molina, A., Ellis, T.I.A., Young, R.I.M. and Bell, R., 1995. Modelling manufacturing capability to support concurrent engineering. *Concurrent Engineering Research and Applications*. 3(1), pp. 29-42.

Nassehi, A., Liu, R. and Newman, S.T., 2007. A new software platform to support feature-based process planning for interoperable STEP-NC manufacture. *International Journal of Computer Integrated Manufacturing*. 20(7), pp. 669-683.

Neumann, G., Backofen, R., Baur, J., Becker, M. and Braun, C., 1997. An information extraction core system for real world German text processing. In: *Proceedings of the ANLP-97*, Washington, USA.

Notepad++. [Online] Available at: **http://notepad-plus.sourceforge.net/uk/site.htm**

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

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

Noy, N.F. and Musen, M.A., 2000. PROMPT: algorithm and tool for automated ontology merging and alignment. In: *Proceedings of the 17th National Conference on Artificial Intelligence*. Austin, TX, USA.

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, pp. 983-1024.

Noy, N.F. and Stuckenschmidt, H., 2005. Ontology alignment: an annotated bibliography. In: Kalfoglou, Y., Schorlemmer, M., Sheth, A., Staab, S. and Uschold, M., eds. *Semantic Interoperability and Integration*. Dagstuhl, Germany: Dagstuhl Seminar Proceedings.

Nurmilaakso, J.-M., 2004. Supply chain scheduling and distributed parallel simulation. *Journal of Manufacturing Technology Management*. 15(8), pp. 756-770.

Ontology Works Inc. [Online] Available at: **http://www.ontologyworks.com**

Otto, H.E., 2001. From concepts to consistent object specifications - translation of a domain-oriented feature framework into practice. *Journal of Computer Science and Technology*. 16(3), pp. 208-230.

PABADIS' PROMISE (Plant Automation based on Distributed Systems, Product-oriented Manufacturing Systems for Re-Configurable Enterprises), 2006. Development of product and production process description language: development of manufacturing ontology (PABADIS' PROMISE Ontology). Work Package 3 Task 3.1. [Online] Available at:
http://www.uni-magdeburg.de/iaf/cvs/pabadispromise/dokumente/Del_3_1_Final.pdf

Pakalnickiene, E. and Nemuraite, L., 2007. Checking of conceptual models with integrity constraints. *Information Technology and Control*. 36(3).

Panetto, H., Scannapieco, M. and Zelm, M., 2004. INTEROP NoE – interoperability research for networked enterprises applications and software. In: Meersman, R. et al, eds. *On the Move to Meaningful Internet Systems*. OTM 2004 Workshops. LNCS 3292, pp. 866-882. Berlin Heidelberg: Springer.

Patil, L., Dutta, D. and Sriram, R., 2005. Ontology-based exchange of product data semantics. *IEEE Transactions on Automation Science and Engineering*. 2(3), pp. 213-225.

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

Pouchard, L., Ivezic, N. and Schlenoff, C., 2000. Ontology engineering for distributed collaboration in manufacturing. In: *Proceedings of the AIS2000 Artificial Intelligence and Simulation Conference*. Tuscon, Arizona, USA.

Pretorius, A.J., 2004. Ontologies - introduction and overview. In: Chapter 2 of MSc Thesis (Unpublished Version). Brussels, Belgium: Vrije Universiteit Brussel.

Process Specification Language (PSL). [Online] Available at:
http://www.mel.nist.gov/psl

Protégé (Ontology Editor and Knowledge Acquisition System). [Online] Available at: **www.protege.stanford.edu**

Rabe, M. and Gocev, P., 2008. Semantic web framework for rule-based generation of knowledge and simulation of manufacturing systems. In: Mertins, K., Ruggaber, R., Popplewell, K. and Xu, X., eds. *Enterprise interoperability III: new challenges and industrial approaches*. pp. 397-409. London, UK: Springer-Verlag.

Ray, S.R, 2004. *NIST's Semantic Approach to Standards and Interoperability*. PowerPoint Presentation. [Online] Available at:
http://ontolog.cim3.net/file/resource/presentation/NIST_Semantics--SteveRay_20040212a.ppt

Ray, S.R. and Jones, A.T., 2003. Manufacturing interoperability. In: *Proceedings of the 10th ISPE International Conference*. pp. 535-540. Madeira Island, Portugal.

Ray, S.R. and Jones, A.T., 2006. Manufacturing interoperability. *Journal of Intelligent Manufacture*. 17, pp. 681-688.

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

Research Triangle Institute, 1999. Interoperability cost analysis of the U.S. automotive supply chain. 99-1 Planning report prepared for the National Institute of Standards and Technology. [Online] Available at:
http://www.nist.gov/director/prog-ofc/report99-1.pdf

Rodriguez, K. and Al-Ashaab, A., 2005. Knowledge web-based system architecture for collaborative product development. *Computers in Industry*. 56, pp. 125-140.

Ruggaber, R., 2006. ATHENA – Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications. In: Konstantas, D., Bourrières, J.-P., Léonard, M. and Boudjilida, N., eds. *Interoperability of enterprise software and applications*. pp. 459-460. London, UK: Springer-Verlag.

Schlenoff, C., Gruninger, M., Ciocoiu, M., Lee, J., 1999. The essence of the Process Specification Language. *Special Issue on Modelling and Simulation of Manufacturing Systems in the Transactions of the Society for Computer Simulation*.

SCRA, 2006. STEP Application Handbook - ISO 10303 Version 3. [Online] Available at: **http://www.tc184-sc4.org/**

Semantic Web Rule Language (SWRL). [Online] Available at:
http://www.w3.org/Submission/SWRL/

Semantic Web Rule Language (SWRL) Built-Ins. [Online] Available at:
http://www.daml.org/rules/proposal/builtins.html#8.1

Seo, W., Lee, S., Kim, K., Kim, B.-I. and Lee, J.Y., 2006. Product data interoperability based on a layered reference ontology. In: *Proceedings of the 1st Asian Semantic Web Conference*. pp. 573-587. Beijing, China.

Shah, J.J., 1995. *Parametric and feature-based CAD/CAM - concepts, techniques and applications*. New York, USA: Wiley.

Shvaiko, P. and Euzenat, J., 2008. Ten challenges for ontology matching. In: *Proceedings of the 7th International Conference on Ontologies, Databases and Applications of Semantics*. Monterrey, Mexico.

Siemens PLM Software Website. [Online] Available at:
http://www.plm.automation.siemens.com/en_gb/products/nx/

Simpson, T.W., 2004. Product platform design and customization - status and promise. *Artificial intelligence for engineering design, analysis and manufacturing*. 18(2), pp. 3-20.

Studer, R., Benjamins, V.R. and Fensel, D., 1998. Knowledge engineering: principles and methods. *Data and Knowledge Engineering*. 25, pp. 161-197.

Stumme, G. and Maedche, A., 2001a. Ontology merging for federated ontologies on the semantic web. In: *Proceedings of the International Workshop for Foundations of Models for Information Integration*. Viterbo, Italy.

Stumme, G. and Maedche, A., 2001b. FCA-Merge: bottom-up merging of ontologies. In: *Proceedings of the 7th International Conference on Artificial Intelligence*. pp. 225-230. Seattle, WA.

Stumme, G., Taouil, R., Bastide, Y., Pasquier, N. and Lakhal, L., 2000. Fast computation of concept lattices using data mining techniques. In: *Proceedings of the 7th International Workshop on Knowledge Representation Meets Databases*. pp. 21-22. Berlin.

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. In: *Proceedings of the International Conference on Product Lifecycle Management*. pp. 495-506.

Sudarsan, R., Fenves, S.J., Sriram, R.D. and Wang, F., 2005. A product information modelling framework for Product Lifecycle Management. *Computer Aided Design*. 37, pp. 1399-1411.

Szykman, S., Fenves, S., Keirouz, W. and Shooter, S., 2001. A foundation for interoperability in next-generation product development systems. *Computer Aided Design*. 33(7), pp. 545-559.

Tam, S., Lee, W.B., Chung, W.W.C. and Lau, H.C.W., 2000. An object-based process planning and scheduling model in a product design environment. *Logistics Information Management*. 13(4), pp. 191-200.

Taylor, B.N. and Thompson, A., eds., 2008. The international system of units. National Institute of Standards and Technology (NIST) Special Publication 330.

TC184/SC4. Setting the standards for industrial data. [Online] Available at: **http://www.tc184-sc4.org/**

The Open Group Architecture Framework (TOGAF). [Online] Available at: **http://www.togaf.org/**

The Zachman Enterprise Framework. [Online] Available at: **http://www.zachmaninternational.com/**

Thomas, H., Markscheffel, B. and Redmann, T., 2008. From subject to concept clouds - why semantic mapping is necessary. In: *Proceedings of the 1st International Conference on Knowledge Federation*. Croatia: Inter University Centre Dubrovnik.

Van Heijst, G., Schreiber, A.Th. and Wielinga, B.J., 1997. Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies*. 45, pp. 183-292.

Vetere, G. and Lenzerini, P., 2005. Models for semantic interoperability in Service-Oriented Architectures. *IBM Systems Journal*. 44(4), pp. 887-903.

Visser, P.R.S, Jones, D.M, Bench-Capon, T.J.M and Shave, M.J.R., 1997. An analysis of ontology mismatches: heterogeneity vs. interoperability. In: *AAAI 1997 Spring Symposium on Ontological Engineering*. Stanford, USA.

Vujasinovic, M., Ivezic, N., Kulvatunyou, B. and Barkmeyer, E., 2007. An industrial validation of a semantic-mediation architecture. USA: National Institute of Standards and Technology. [Online] Available at:
http://www.mel.nist.gov/msidlibrary/doc/IEEE_validation.pdf

Wang, F., Fenves, S.J., Sudarsan, R. and Sriram, R.D., 2003. Towards modelling the evolution of product families. In: *Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conferenc*e/. Chicago, Illinois, USA.

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.

Web Ontology Language (OWL). [Online] Available at:
http://www.w3.org/TR/owl-features/

Westkämper, E., Schmidt, T. and Wiendahl, H.H., 2000. Production planning and control with learning technologies - simulation and optimisation of complex production processes. In: Leondes, G., ed. *Knowledge-based systems*. New York Academic Press.

Wei, S., Qin-yi, M., Tian-yi, G., 2009. An ontology-based manufacturing design system. *Information Technology Journal*. 8(5), pp. 643-656.

Wikipedia - The Free Encyclopaedia. [Online] Available at:
http://www.wikipedia.org

Williams, T.J., 1994. The Purdue Enterprise Reference Architecture. *Computers in Industry*. 24(2-3), pp. 141-158.

Yang, D., Dong, M. and Miao, R., 2008. Development of a product configuration system with an ontology-based approach. *Computer-Aided Design*. 40, pp. 863-878.

Yang, M. and Yang, J., 2008. Machine-part cell formation in group technology using a modified ART1 method. *European Journal of Operational Research*. 188(1), pp. 140-152.

Yang, Q.Z. and Zhang, Y., 2006. Semantic interoperability in building design: methods and tools. *Computer Aided Design*. 38, pp. 1099-1112.

Ye, Y., Yang, D., Jiang, Z. and Tong, L., 2008. Ontology-based semantic models for supply chain management. *International Journal of Advanced Manufacturing Technology*. 37, pp. 1250-1260.

Yongtao, H. and Jingying, M., 2006. A knowledge-based auto-reasoning methodology in hole-machining process planning. *Computers in Industry*. 57, pp. 297-304.

Young, R.I.M. and Bell, R., 1993. Design by features - advantages and limitations in machine planning integration. *International Journal of Computer Integrated Manufacturing*. 6(1-2), pp. 105-112.

Young, R.I.M., Gunendran, G., Chungoora, N., Harding, J. and Case, K., 2009. Enabling interoperable manufacturing knowledge sharing in PLM. In: *Proceedings of the 6th International Product Lifecycle Management Conference*. Bath, UK: University of Bath.

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), pp.1505-1519.

Zhao, J., Cheung, W.M. and Young, R.I.M., 1999. A consistent manufacturing data model to support virtual enterprise. *International Journal of Agile Management Systems*. 1(3), pp. 150-158.

Zhou, X., Qiu, Y., Hua, G., Wang, H., Ruan, X., 2007. A feasible approach to the integration of CAD and CAPP. *Computer-Aided Design*. 39, pp. 324-338.

APPENDICES

A The Knowledge Engineering Methodology and IDEF5 Schematics for Ontology Development

A.1 The Knowledge Engineering Methodology

The Knowledge Engineering Methodology has been prescribed by Noy and McGuinness (2001) and consists of a stepwise approach in the process of developing ontologies. The diagram in Figure A-1 illustrates a typical ontology development process following the Knowledge Engineering Methodology and applied to this research work.

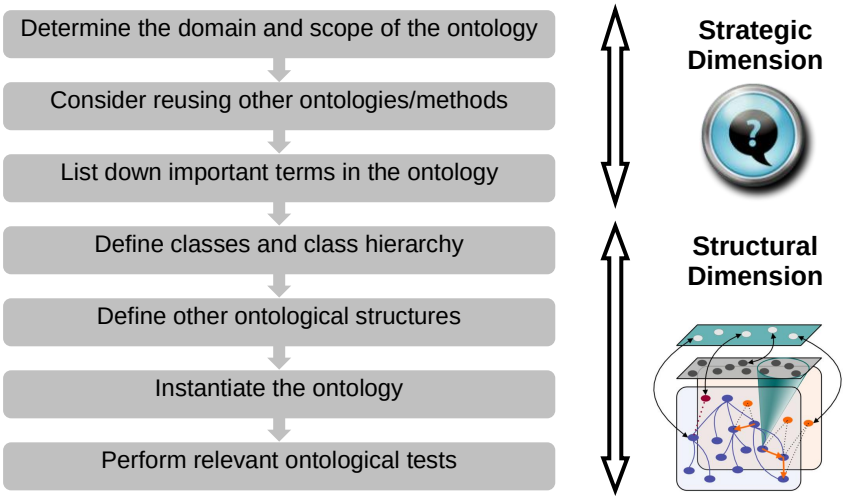


Figure A-1 The Knowledge Engineering Methodology (Adapted from Noy and McGuinness (2001))

The first stage in the process is concerned with the specification of the domain and scope of the ontology. Some questions that need to be asked at this stage are, for example:

- What should the domain and scope of the ontology cover?
- Who are the parties involved in exploiting the ontology?
- For what types of questions should the concepts developed in the ontology support answers to?

This first strategic dimension of the Knowledge Engineering Methodology is generally accompanied by the definition of competency questions to be

answered after the ontology development process is performed. These competency questions form part of a checklist for assessing whether the objectives of the ontology have been achieved or not.

The second stage in the process involves the consideration for reusing other ontologies and/or methods. This process also forms part of the strategic view on the ontology. For example, in this research work, the Process Specification Language ontology has been reused and formalised in the framework under development. In this way, the time taken to develop an ontology can be shortened.

The third process considers the enumeration of vital terms to go in the ontology. It is important to list down the terms, concepts, verbs and statements that fall within the scope of the ontology intended to be modelled. This acts as a mind-map which can later be refined as the structural dimension of the ontology is tackled.

As part of the structural dimension of an ontology, and subsequently a KB based on the ontology, classes and the class hierarchy are first defined. These capture the taxonomy, or backbone, of main concepts in the ontology.

Next, other ontological structures are modelled. These involve relations of the required arities in order to bind classes together to create statements. Ontological functions, which are a special case of relations, are also defined at this stage. Relevant axioms governing the way in which ontological content is to be formally interpreted are also specified.

The next stage consists of populating the ontology with instances of the developed classes, and using the ontological structures to create fact sentences, in order to capture discrete knowledge in the KB supported by the ontology.

Relevant ontological tests are performed in order to investigate the extent to which the initially-set competency questions are met.

A.2 IDEF5 Schematics

At present, the graphical representation of ontological content is partly dependent on the implementation platform in which an ontology is being modelled. There is currently no de facto ontology representation schematics in order to aid the visual communication of ontological content. Diagrams provided in the Unified Modelling Language (UML), for example, UML class diagrams or EXPRESS-G schematics could be exploited towards the graphical representation of ontologies. However, the unique issue with similar diagrams is that they do not allow the complete representation of certain types of relations, notably higher-arity relations.

After careful scrutiny, it was discovered that the IDEF5 schematic language (Knowledge Based Systems Inc., 1994) serves as a very suitable candidate for allowing the informal representation of ontological content in the form of schematics with a clear set of primitive semantics. Figure A-2 identifies the primitive symbols from the IDEF5 schematic language exploited in this work. Note that because no commercial tool is currently available for drawing IDEF5 schematics, a Microsoft Visio template has been constructed for optimising the reuse of symbols in the IDEF5 schematic language.

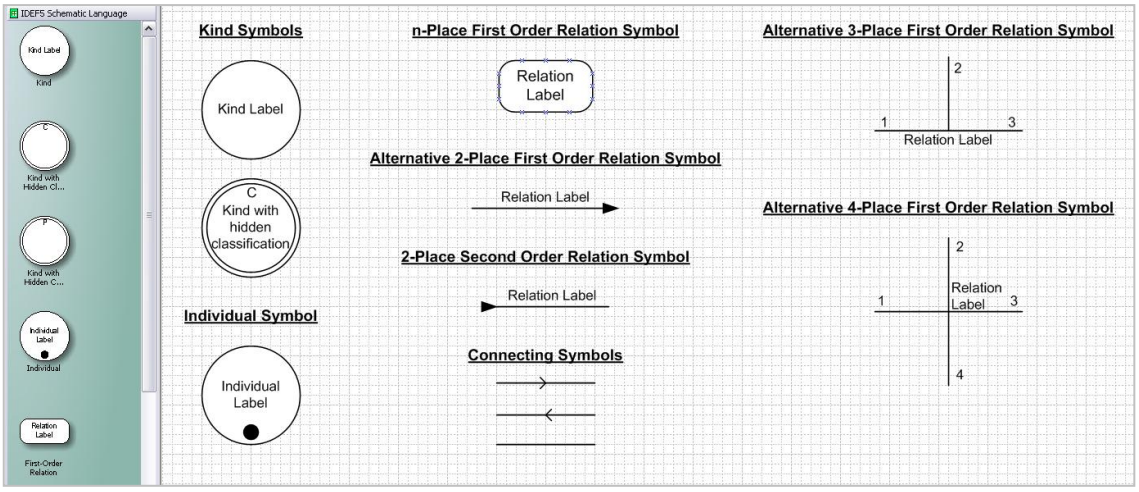


Figure A-2 Microsoft Visio Template and Symbols Used in the IDEF5 Schematic Language

An overview on how the various symbols are put together to represent ontological content is next identified. Figure A-3 illustrates how a taxonomy of

classes is organised by using “kind symbols” and the “2-place second order relation symbol” with or without labelling (note that the relation without the “sup” labelling assumes the same semantics as with the “sup” labelling. Both refer to the notion of “has super-class”). In the event that a certain class possesses a hidden taxonomy of its own, this is represented using the symbol “kind with hidden classification”.

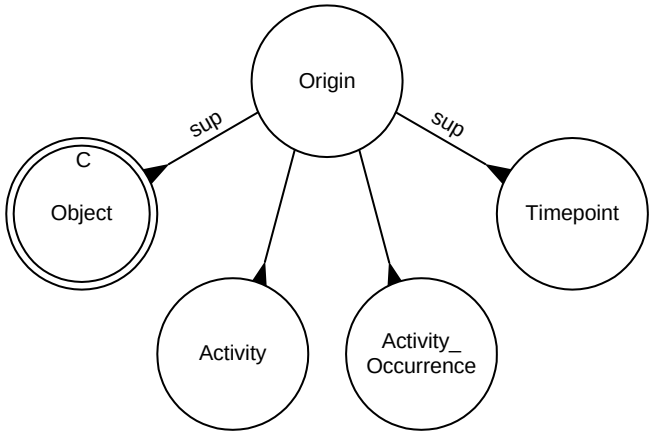


Figure A-3 Representing a Taxonomy of Classes Using IDEF5 Schematics

IDEF5 schematics can also be used to capture statements about how classes are bound together through relations of various arities. Figure A-4 depicts a binary relation (of arity 2) named “occurrence_of” which binds the classes “Activity_Occurrence” and “Activity”, read in the direction of the arrows. In the figure, two alternative ways of representing the same information is illustrated.

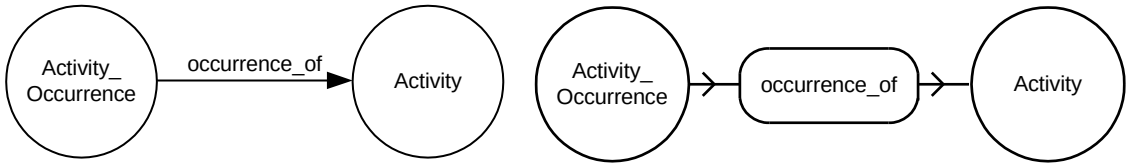


Figure A-4 Alternative Ways of Representing a Binary Relation between Two Classes Using IDEF5 Schematics

Relations of higher arities can readily be captured using the IDEF5 schematic language. The diagram portrayed in Figure A-5 exemplifies a ternary relation (of arity 3), called “participates_in”, which involves three argument classes to one relation. The way in which the relation is read follows the order in which the numbers appear in the relation. In this case the interpretation would state

that the class “Object” in the first argument position “participates_in” the class “Activity_Occurrence” in the second argument position at the class “Timepoint” in the third argument position.

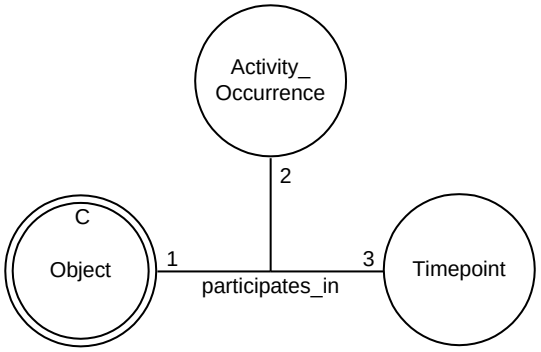


Figure A-5 Representing a Ternary Relation among Three Classes Using IDEF5 Schematics

Instances are organised using the “2-place second order relation symbol” with the directive “inst” as a label to the relation. The “inst” labelling captures the “instance-of” relation that holds between classes and their individuals. Figure A-6 illustrates two instances of the class “Timepoint” namely “inf-“ and “inf+”.

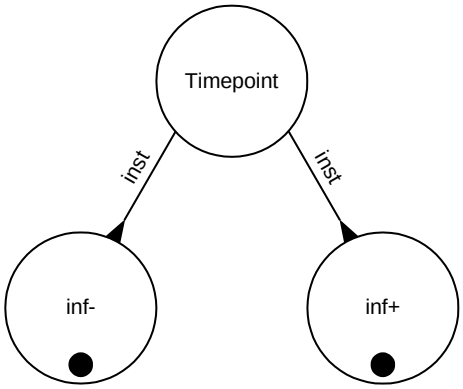


Figure A-6 Organising Instances of Classes Using IDEF5 Schematics

Similar to the way in which classes can be bound to relations, individuals (instances) can also be stated as being bound to the relations inherited from the classes that the individuals instantiate. Overall, IDEF5 schematics provide an attractive way of organising and representing ontological content prior to implementation in a suitable ontological environment. In other words, IDEF5 schematics help obtain a platform independent model of ontological information, which an important facet in Model Driven Architectures.

B Justification of the Chosen Common Logic-Based Ontological Formalism

B.1 Introduction

The present capability that ontology-based approaches offer to formally represent and share product design and manufacture semantics is partly dependent on the choice of ontology representation formalism. Since there currently exists a spectrum of these formalisms, it is an important requirement to understand which family of formalisms allows the expressive capture and representation of product design and manufacture semantics (see Requirement 4a, section 3.3.4.1). The aim of this chapter is to justify the choice of the Common Logic-based formalism used throughout this work, as a viable direction to meet the semantic interoperability needs across product design and manufacture. In order to establish this direction, two recognised heavyweight ontological formalisms are first explored and tested, namely:

- Frames and First Order Logic (Gómez-Pérez et al, 2004). In this case, Protégé Frames with its first order constraint language PAL (Protégé Axiom Language) are investigated in section B.2.
- Description Logics (Gómez-Pérez et al, 2004). In this case, the Web Ontology Language (OWL) with the rule language SWRL (Semantic Web Rule Language) are investigated in section B.3.

The main focus of investigating the two above-mentioned ontological formalisms is to identify their potentials and limitations for expressively capturing and representing entity information and process semantics, a significant requirement which the framework concept needs to satisfy. In the explorations, sample ontologies are constructed following the knowledge engineering methodology prescribed by Noy and McGuiness (2001). Section B.4 then covers the main reasons why Common Logic-based formalisms possess better semantic capabilities compared to the two analysed formalisms. Finally, section B.5 summarises this appendix.

B.2 An Exploration of Frames with a First Order Constraint Language

B.2.1 Aim of Investigation

The aim of this exploration is to comprehensively evaluate the capabilities and suitability of Frames with a first order constraint language as ontological formalism to model heavyweight entity information and process semantics. Following the knowledge engineering methodology (Noy and McGuiness, 2001), a number of competency questions have been identified. The significance of the competency questions is such that at the discussion stage, these questions can be checked against the observations made in order to propose appropriate recommendations. In this first study, the following list of competency questions has been formulated:

- Is Frames with a first order constraint language sufficiently expressive to support the representation of entity information semantics?
- Is Frames with a first order constraint language sufficiently expressive to support the representation of process semantics?
- Is it possible using Frames with a first order constraint language to specify entity information and process semantic relationships?

B.2.2 Objectives

A number of objectives has been identified in order to meet the aim of the investigation:

- Firstly, it is required to verify the extent to which the semantics of different contexts can be captured in a Machining Hole Feature Ontology. Basic entity information semantics are being considered partly from an external source (Canciglieri, 1999) and a view on STEP 10303-224, whilst selected Process Specification Language (PSL) concepts provide the fundamental process semantics.

- To use Protégé Frames with the Protégé Axiom Language (PAL) as heavyweight ontological formalism in the Protégé version 3.4 ontology environment (Protégé Website, 2009). This is primarily because the Protégé environment is consensually regarded as the most mature tool for knowledge modelling (PABADIS' PROMISE Deliverable 3.1, 2006).

B.2.3 Machining Hole Feature Ontology

A number of reasons account for the choice of a Machining Hole Feature Ontology in the first instance. The main one lies in the fact that the test ontology acts as a suitable starting point as it regroups three contexts (not be to be confused with namespaces in this case). These contexts involve a manufacturing process viewpoint to capture process semantics and a feature representation coupled with a geometry context to capture entity information semantics of hole features from a machining and GD & T viewpoint. The second reason is concerned with feature information serving as the bridge to a high level integration between design, analysis, process planning and manufacturing (Zhou et al, 2007), hence implying that certain relationships can be captured between process and entity information semantics in the test ontology.

B.2.3.1 Entity Information Semantics

Several classes and their respective taxonomy, relationships to other classes, ontological functions and instances have been defined for capturing entity information semantics following the ontology development procedure. Figure B-1 which follows identifies a screenshot of the Machining Hole Feature Ontology developed in the Protégé environment. A number of ontological entities are highlighted in the diagram along with short comments detailing the nature of these entities.

Consider the concrete class "Simple Hole" **(A)** found in the hierarchy of the abstract class "Machining Hole Feature" **(B)**. The latter can purposely be made abstract so as to imply that it cannot have direct instances or

individuals, meaning any machining hole feature should in fact exist as an instance of one of the concrete subclasses of the abstract class. Instances are regarded as being the most specific concepts represented in a knowledge base (Noy and McGuiness, 2001). In the figure, for example, it is possible to depict an instance of “Simple Hole” named “Hole 13.00” **(C)**.

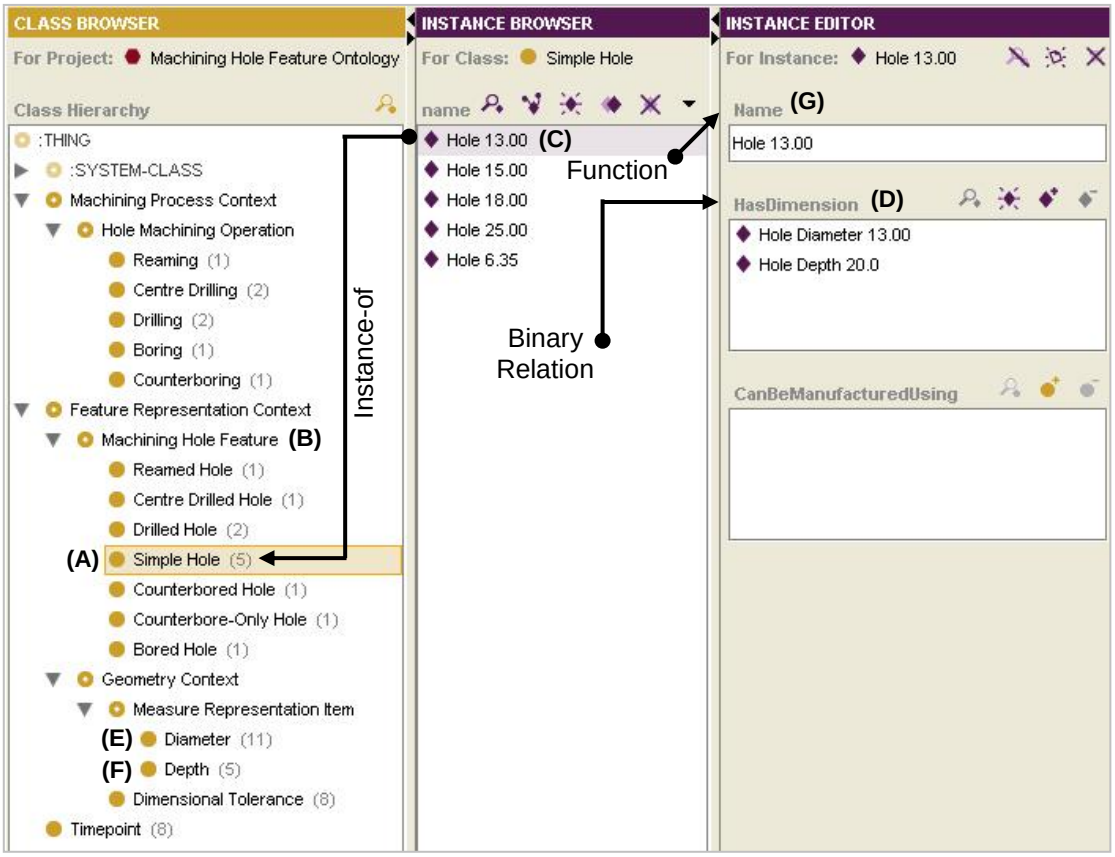


Figure B-1 Entity Information Semantics in the Machining Hole Feature Ontology

The ontological formalism under investigation also allows binary relations to be captured. Binary relations are ontological entities that bind two sets of classes or arguments together. The “hasDimension” **(D)** relation in Figure B-1 is an example of a binary relation defined to relate the class “Simple Hole” **(A)** to a union of the classes “Diameter” **(E)** and “Depth” **(F)**. An example of an ontological function is the parameter called “Name” **(G)**, whose value is of type string. This function (in the ontological sense) works very similar to attributes in Object-Oriented languages.

So far, the types of entity information semantic structures exposed remain lightweight in nature. In order to account for heavyweight semantics in the ontology, an additional axiom layer is required. The capability to do so is dependent on the specification of axioms or integrity constraints using the Protégé Axiom Language (PAL). This constraint language accommodates first order semantics which is very expressive.

The underlying philosophy of PAL is model-checking (Protégé Website, 2009) and hence, PAL-formalised integrity constraints are used in the heavyweight approach to restrict the interpretation of ontological entities. These constraints are primarily written to ascertain that the semantic structures are carefully respected when knowledge is asserted in the ontology and are an essential asset for the capture of semantics and intent. In order to verify whether asserted ontological knowledge violates or conforms to semantics, integrity constraints can be processed and a number of results are retained in the event that these constraints are infringed. In other words, integrity constraints contribute to the semantic integrity and enrichment of ontologies.

In the Machining Hole Feature Ontology, a number of integrity constraints have been specified. The expression listed next (Expression B-1) gives an example of a simple integrity constraint axiom whose purpose is to ensure that all instances of the class “Simple Hole” **(A)** are only allowed to have exactly two allowable related dimensional parameters and it is compulsory that these parameters include one instance of the class “Diameter” **(E)** and one instance of the class “Depth” **(F)** (see Figure B-1). If, for example, an instance of “Simple Hole” were asserted as having (1) more than two related dimensional parameters or (2) a combination of two dimensional parameters that did not comprise of a diameter and a depth or (3) no dimensional parameters at all, then an execution of the PAL constraint would show that this instance violates the constrained semantics captured in the integrity constraint.

```
(defrange ?hole :FRAME 'Simple Hole')
(defrange ?dia :FRAME 'Diameter')
(defrange ?depth :FRAME 'Depth')
(forall ?hole
        (=> (instance-of ?hole 'Simple Hole')
            (and (= (number-of-slot-values hasDiameter ?hole) 2))
                 (exists ?dia (exists ?depth
                              (and (instance-of ?dia 'Diameter')
                                   (instance-of ?depth 'Depth')
                                   (hasDimension ?hole ?dia)
                                   (hasDimension ?hole ?depth)))))))
```

Expression B-1 A Simple Integrity Constraint Written in the Protégé Axiom Language (PAL)

B.2.3.2 Process Semantics

A number of relevant ontological entities are considered for the definition of machining process semantics. Some of these notions derive from PSL since the latter explicitly and clearly defines the concepts intrinsic to manufacturing process information (Schlenoff et al, 1999). Hence, for describing the semantics of machining sequences, it is necessary to characterise processes such as "Centre Drilling" **(H)** (see Figure B-2) in terms of their beginning and completion times. In Figure B-2, two binary relations are present namely "hasStartTime" **(I)** and "hasEndTime" **(J)**, which directly relate instances of the concrete subclasses of "Hole Machining Operation" **(K)** to instances of the class "Timepoint" **(L)**. Another binary relation named "precedes" **(M)** has been defined with the intention of permitting the specification of precedence relationships over processes. From Figure B-2, it can also be seen that there is the notion of the class "Timepoint" **(L)** and a binary relation named "before" **(N)** that only holds between timepoints and provides linear ordering over timepoints. A timepoint instance also has a floated value type hence the "hasValue" **(O)** ontological function.

To attempt at capturing some of the heavyweight axioms governing the relation "before" **(N)** from PSL, PAL statements are written. Expression B-2 identifies one axiom that constraints the "before" **(N)** relation by assigning an irreflexive property to the relation. This expression informally states that if

there is a timepoint, then this timepoint can never happen before itself. Furthermore, a second axiom placed on the “before” **(N)** relation depicts the transitive nature of the relation i.e. if a timepoint ?t1 is before another timepoint ?t2 which is before another timepoint ?t3, then it is evident that ?t1 is before ?t3. The statement in Expression B-3 captures the transitive property of the relation.

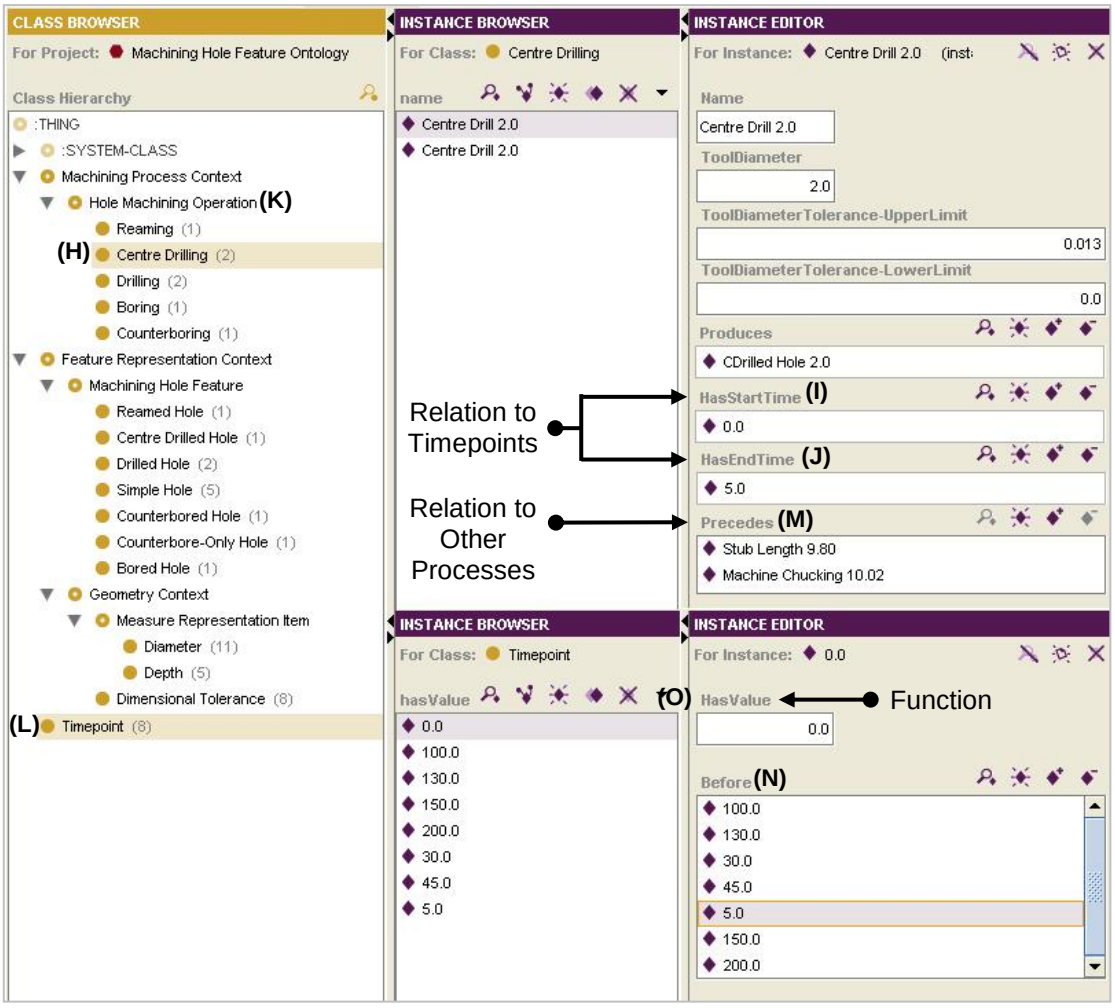


Figure B-2 Process Semantics in the Machining Hole Feature Ontology

```
(defrange ?t1 :FRAME Timepoint)
(forall ?t1
        (=> (instance-of ?t1 Timepoint)
            (not (before ?t1 ?t1))))
```

Expression B-2 Irreflexive Axiom for the “before” Relation

```
(defrange ?t1 :FRAME 'Timepoint')
(defrange ?t2 :FRAME 'Timepoint')
(defrange ?t3 :FRAME 'Timepoint')
(forall ?t1 (forall ?t2 (forall ?t3
        (=> (and (before ?t1 ?t2)
                 (before ?t2 ?t3))
            (before ?t1 ?t3)))))
```

Expression B-3 Transitive Axiom for the "before" Relation

B.2.3.3 Entity Information and Process Semantic Relationships

As previously discussed in section 3.3.2 (Requirement 2), providing semantic relationships among viewpoints conveys the capability to link entity information semantics to process semantics in a knowledgeable way. For example, in manufacturing, the dimensional and tolerance parameters of features have a direct influence on the choice of machining processes.

In order to further explore this understanding in the "Machining Hole Feature Ontology" using Protégé Frames and PAL, a binary relation called "canBeManufacturedUsing" **(P)** has been defined (see Figure B-3). This relation binds the subclasses of "Machining Hole Feature" **(B)** to subclasses of "Hole Machining Operation" **(K)** so that it can, for example, be stated that an instance "Hole 13.00" **(C)** of the class "Simple Hole" **(A)** can be manufactured using some instance of the class "Drilling" **(Q)**.

To support the knowledge which leads to the decision of which machining operation can be used to manufacture a certain machining hole feature, knowledge contained in tables from ISO Tolerance Band and machining processes associated with ISO IT Tolerance Grade (ISO 286-2, 1988) are exploited. Figure B-4 briefly demonstrates this knowledge acquisition process facilitated through the heavyweight formalisation of a relevant subset of the knowledge using PAL statements.

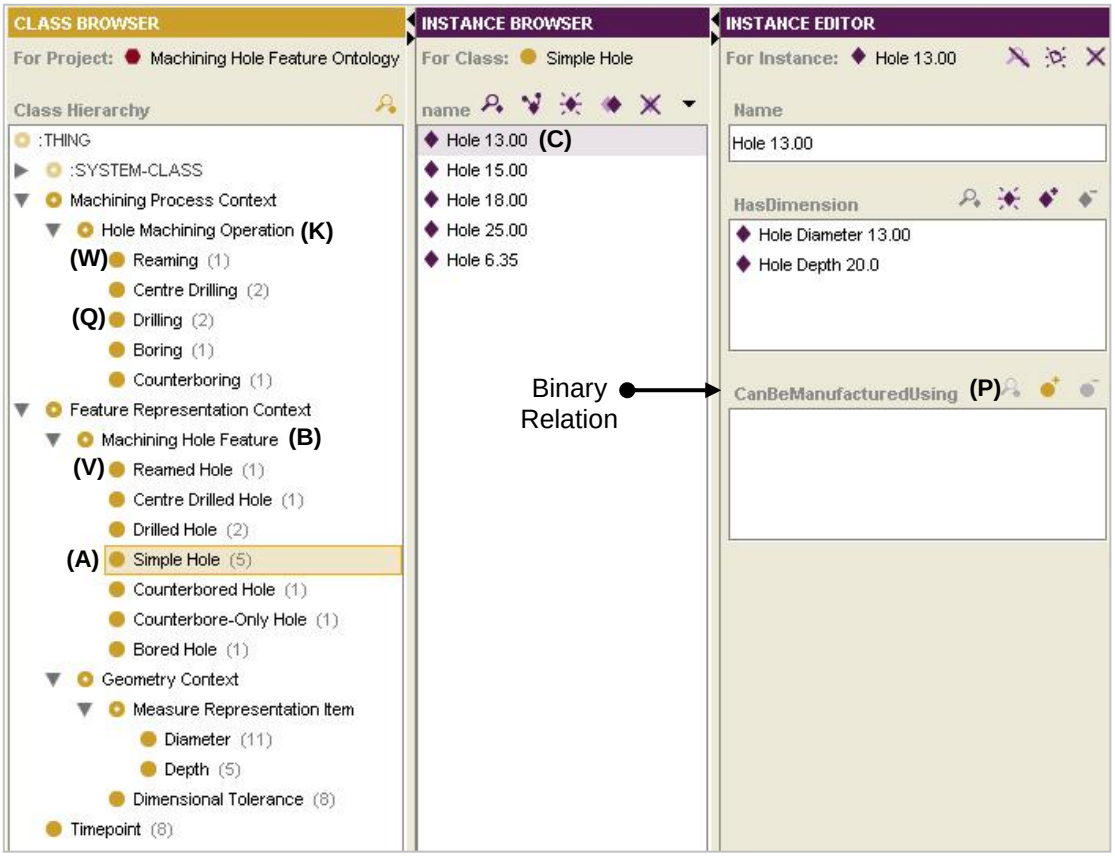


Figure B-3 Example of a Semantic Relationship between Entity Information and Process Semantics

The main reason which accounts for the use of information from ISO IT Tolerance Grade is because ISO Tolerance Band tables provide different ranges of dimensions and tolerances that different IT Grades can achieve. These IT Grades are reflected in the machining process table relating various machining processes and their corresponding IT Grade capabilities (see Figure B-4). For example, knowing that a reaming process can achieve nominal dimensions and dimensional tolerances in the range between IT5 and IT9 both inclusive **(R)**, then if the diameter of an instance of “Simple Hole” **(A)** is between 10 mm (exclusive) and 18 mm (inclusive) **(S)** with an absolute value for the diameter tolerance between 0.008 mm and 0.043 mm both inclusive **(T)** based on ISO Tolerance Band tables, this would imply that the simple hole feature fits the reaming process criteria **(U)**.

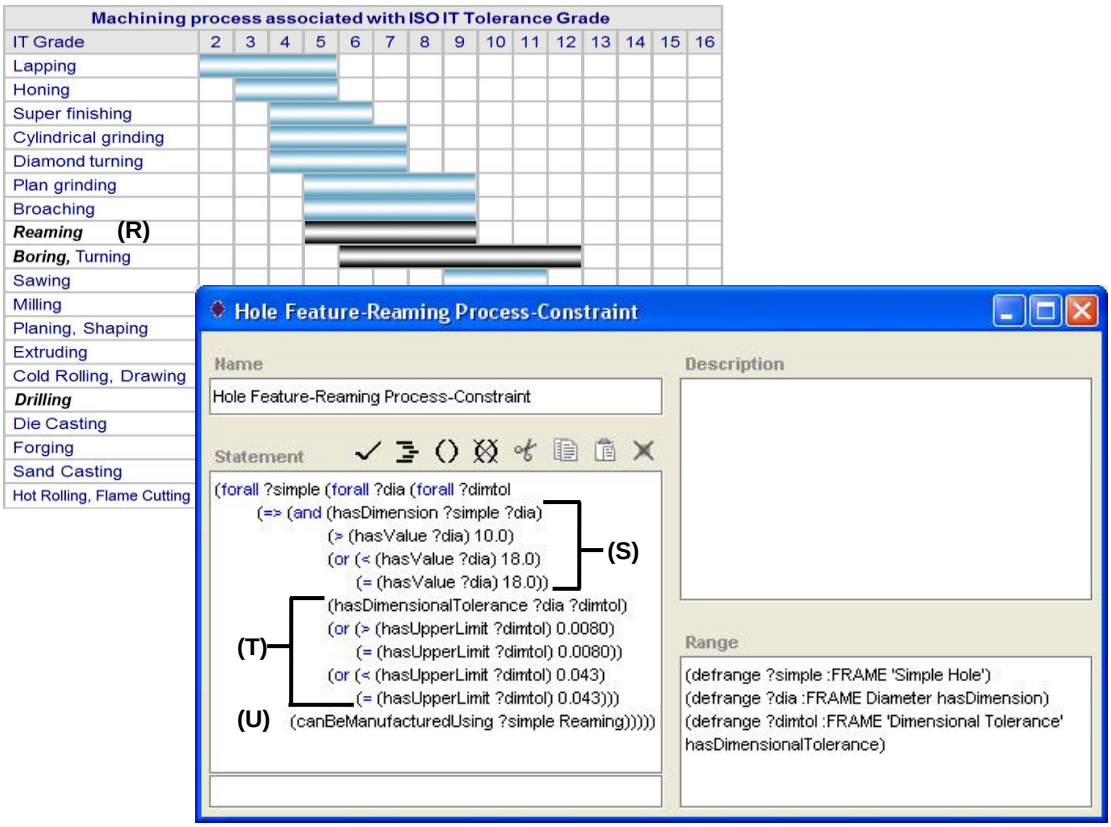


Figure B-4 Formalising Heavyweight Semantic Relationships across Contexts Using the Protégé Axiom Language (PAL)

Figure B-5 illustrates the result of evaluating the PAL constraint. The query responses clearly show that two instances of “Simple Hole” **(A)** conform to the formalised reaming constraint. This further implies that any instance of the class “Simple Hole” **(A)** that satisfies the reaming constraint can in fact exist as a “Reamed Hole” **(V)** which can in turn be produced by some defined “Reaming” process **(W)** (see Figure B-3). Such information can additionally be asserted in the ontology.

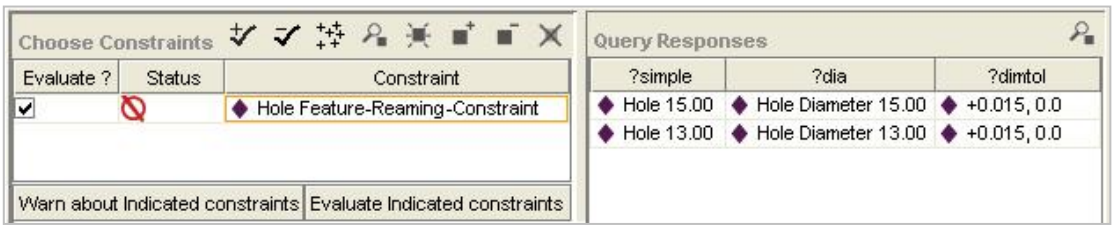


Figure B-5 Example of Query Responses Processed from a PAL Constraint

B.2.4 Discussions

The basic, primarily lightweight, representation of entity information and process semantics and their corresponding relationships is achievable through the specification of classes and their taxonomy accompanied by relations and ontological functions that hold for specific classes and between classes respectively. Furthermore, it is evident that the specification of an ontological axiom layer provides the additional heavyweight semantic structures needed to constrain and verify the interpretation of semantics at computational level. This axiom layer provides an enhanced basis for capturing semantically-enriched ontological concepts, formalised as a set of integrity constraints written in the Protégé Axiom Language (PAL). Competency questions set in section B.2.1 are reviewed next.

- Is Frames with a first order constraint language sufficiently expressive to support the representation of entity information semantics?

It is possible to conclude that Frames with a first order constraint language can be used to capture and represent the most critical types of entity information semantics, from a research scope point of view. However, the extent to which this ontological formalism is able to model more complex entity information semantics is debatable. For example, the ontological formalism only allows the representation of binary relations, which are relations that hold between two sets of classes. This could pose a problem if a relation should hold between three sets of classes. As an example, suppose a relation needs to be defined to express the positional tolerance of a feature. Then, it is very likely that this relation needs to encompass three sets of classes namely (1) the feature that holds the (2) positional tolerance with respect to some (3) toleranced dimension. Such a relation is referred to as a ternary relation as it involves three arguments. Higher-arity relations, i.e. relations with three arguments or more, cannot be captured using Frames with a first order constraint language.

- Is Frames with a first order constraint language sufficiently expressive to support the representation of process semantics?

To an appreciable extent, some of the very basic ontological entities fundamental for expressing process semantics have been represented. The ontological formalism allows some relations, pertinent to the description of manufacturing process sequences, to be defined. For example, it has been possible to probe into a subset of the semantics of the “before” **(N)** relation. However, more complex semantics from PSL cannot be represented since they involve, to a large extent, higher-arity relations and functions (in the ontological sense). Therefore, this where it is primarily perceived that heavyweight Frames with a first order constraint language does not provide sufficient expressivity. The issue with capturing PSL-based process semantics is further scrutinised in section B.3.

- Is it possible using Frames with a first order constraint language to specify entity information and process semantic relationships?

It has been possible to gather an understanding that as long as the defined relationships between entity information and process semantics remain binary relations, the ontological formalism is proficient. The investigation has shown that complex integrity constraints (see Figure B.4) can be specified using PAL. However, in certain cases, if PAL statements involve several different variables to be processed, then query time and responses tend to breakdown. This applies to any PAL constraint, viewed as being overly complex, although the latter could be syntactically sound. This drawback, however, is likely to be related to a limitation of the ontological environment itself rather than the ontological formalism.

B.3 An Exploration of OWL with a Rule Language

B.3.1 Aim of Investigation

The previous exploration suggests that process semantics based on the Process Specification Language (PSL) are the most intricate and difficult to capture and formalise compared to entity information semantics and semantic relationships across viewpoints. Hence, the investigation explained in this section is fully dedicated to the formalisation of PSL semantics, by exploiting another heavyweight ontological formalism namely the Web Ontology Language (OWL) with the Semantic Web Rule Language (SWRL) (W3C Website, 2009). It is to be noted that previous work indicates that OWL is capable of modelling entity information semantics (AIM@SHAPE Product Design Ontology, 2004; Kim et al, 2006; Chungoora and Young, 2008b) and, therefore, this exploration only targets the core issue of process semantics. In this study, a single key competency question is present:

- Is OWL with SWRL sufficiently expressive to support the representation of PSL-based process semantics?

B.3.2 Objectives

The objectives identified in order to meet the aim of the investigation are:

- It is essential to understand to what extent can PSL semantics be captured using OWL and SWRL. This is to be tested through the formalisation of concepts from the PSL Core theory involving PSL primitives and axioms.
- To use a combination of OWL Full, Description Logic-based ontological formalism, with SWRL in the Protégé OWL ontology development environment (Protégé version 3.4). Throughout this study, for simplicity, only the term OWL and SWRL are used.

B.3.3 Modelling PSL Core Semantics Using OWL with SWRL

B.3.3.1 PSL Core Original Semantics

The purpose of PSL Core is to axiomatise a set of intuitive semantic primitives that is adequate for describing the fundamental concepts of manufacturing processes (PSL Website, 2009). There are four kinds of entities that are required for describing process semantics namely:

- Activities which are reusable behaviours in the domain.
- Activity occurrences which are runtime executions of activities.
- Timepoints which provide a linear ordering of the points at which activity occurrences are taking place.
- Objects which are entity information semantics that are neither activities, nor activity occurrences nor timepoints.

PSL Core (refer to Appendix C if needed) consists of (1) a primitive and defined lexicon that identifies the basic semantic structures i.e. classes, relations, ontological functions and individuals, (2) a series of axioms that ensure semantic integrity of the ontology and (3) supporting definitions that provide rules for inference purposes (not investigated in this study as they are essentially axioms too). The following section, therefore, demonstrates how OWL with SWRL can be employed to attempt at modelling PSL semantics.

B.3.3.2 Classes and Binary Relations

OWL allows all PSL classes and binary relations (called properties in OWL) to be easily captured in the ontology. Note that at this stage, only the purely lightweight semantics are under consideration. Figure B-6 identifies the four classes and five binary relations that exist in PSL Core.

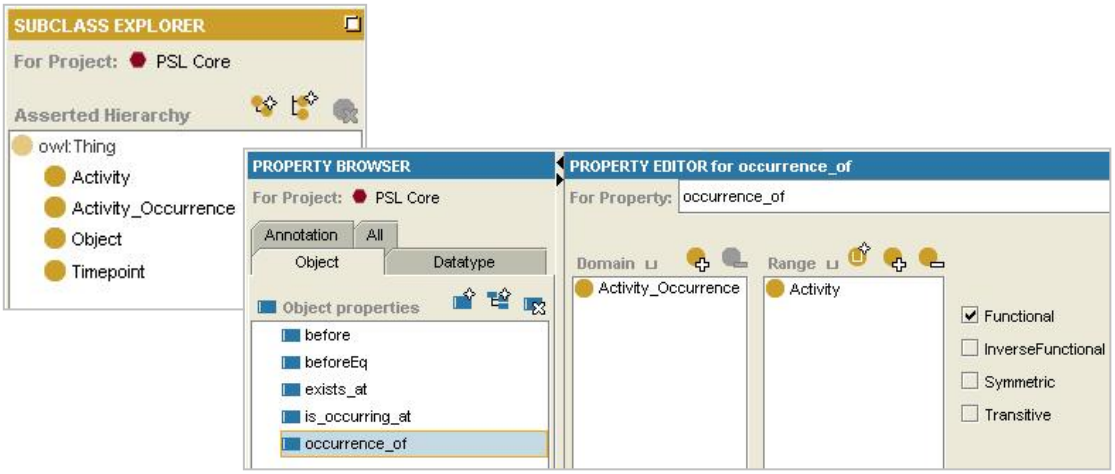


Figure B-6 Classes and Binary Relations from PSL Core in OWL

B.3.3.3 Ternary Relations Approximation to Binary Relations

Since OWL only allows binary relations that hold between two sets of classes, i.e. two arguments, to be represented, ternary relations from PSL Core cannot be exactly represented in the OWL ontology. The most obvious and probably closest approximation to a ternary relation in OWL can be obtained by specifying the relation to be binary in nature. Such a binary relation approximation to a ternary relation has one domain **(A)** to reflect one argument to the relation, with a range consisting of a union of two classes **(B)** to reflect the other two class arguments to the relation (see Figure B-7). It is also possible to break down a ternary relation to form two separate binary relations, but this aspect is not discussed in this study.

Figure B-7 depicts how the original semantics from the ternary relation “participates_in” should be interpreted versus a binary relation approximation of the “participates_in” relation specified in OWL. The ternary “participates_in” relation has been illustrated using a simple IDEF5 schematic. The figure also shows all the other binary relations that are used to approximate ternary relations from the original PSL Core ontology.

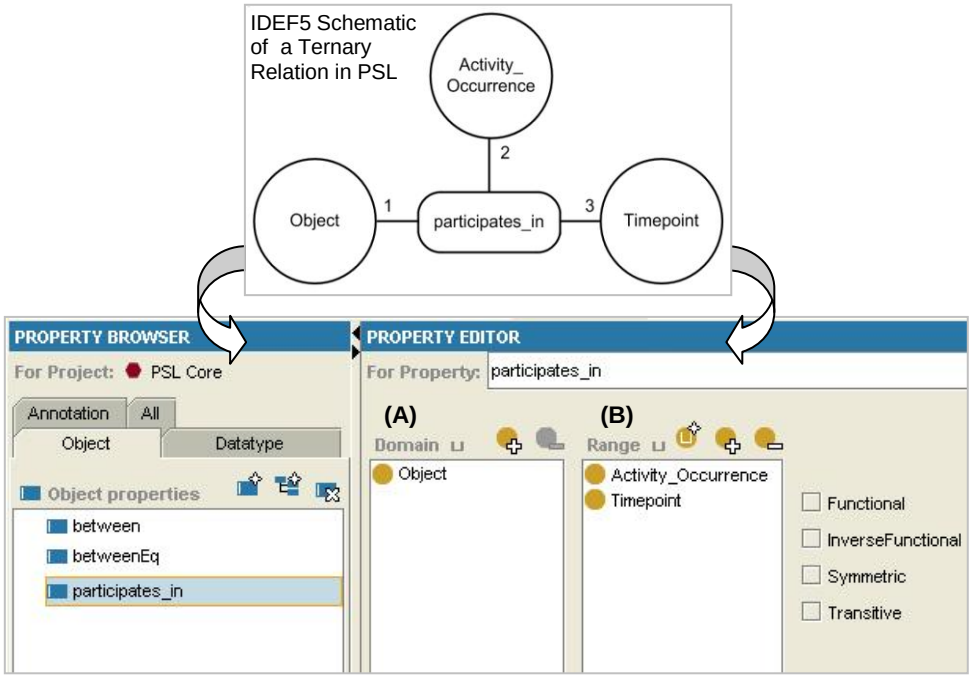


Figure B-7 Example of a Ternary Relation Approximation to a Binary Relation in OWL

B.3.3.4 Unary Functions Approximation to Binary Relations

Functions in ontological terms may be regarded as being particular traits that can hold for the individuals of classes in order to denote individuals from other classes. Two unary functions are present in original PSL Core semantics and they are (1) “beginof” and (2) “endof”. In the informal semantics of PSL Core, it is said that the begin of and end of activity occurrences or objects are timepoints. So, for example, the beginning of an activity occurrence “Drill_Hole_1” can be used to denote a specific timepoint i.e. (beginof Drill_Hole_1) denotes some timepoint ?t, although ?t does not need to be identified in the ontology since “beginof” “Drill_Hole_1” is known.

Functions like “beginof” and “endof” that are used to define the individuals of a class using individuals from another class cannot be specified in OWL. However, OWL does account for datatype properties which are very similar to unary functions used to associate float, integer, string and other types of values to individuals of classes. Due to the inability to model PSL functions in OWL, approximations to these have to be made. In the investigated OWL version of PSL Core, the original “beginof” and “endof” unary functions are

modelled as binary relations. Figure B-8 next illustrates how these binary relations in OWL attempt to capture the semantics behind unary functions, although original semantics are not preserved.

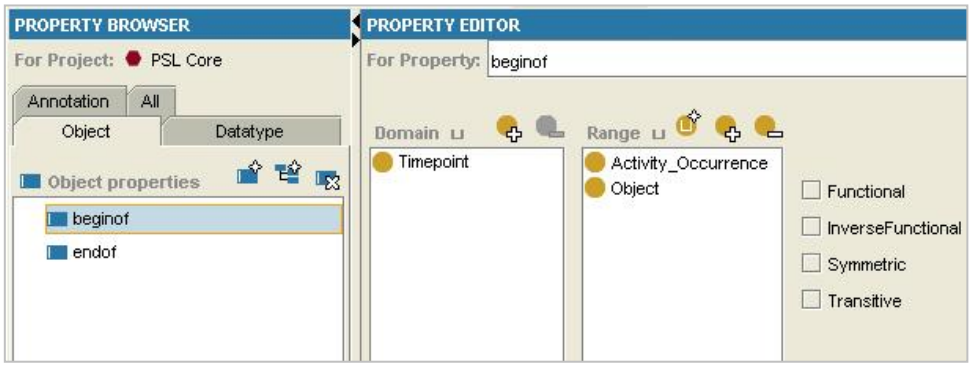


Figure B-8 Example of a Unary Function Approximation to a Binary Relation in OWL

B.3.3.5 Individuals

Two individuals are present from PSL Core theory and they are “inf-“ and “inf+”. These are instances of the class “Timepoint” and are used in the theory to refer to the timepoints that are before and after all other timepoints respectively. Figure B-9 below shows the two individuals of the class “Timepoint”. Note that they have been renamed to “inf_minus” **(C)** and “inf_plus” **(D)** respectively because the symbol “+” cannot be used in the name string of an individual in the Protégé OWL environment.

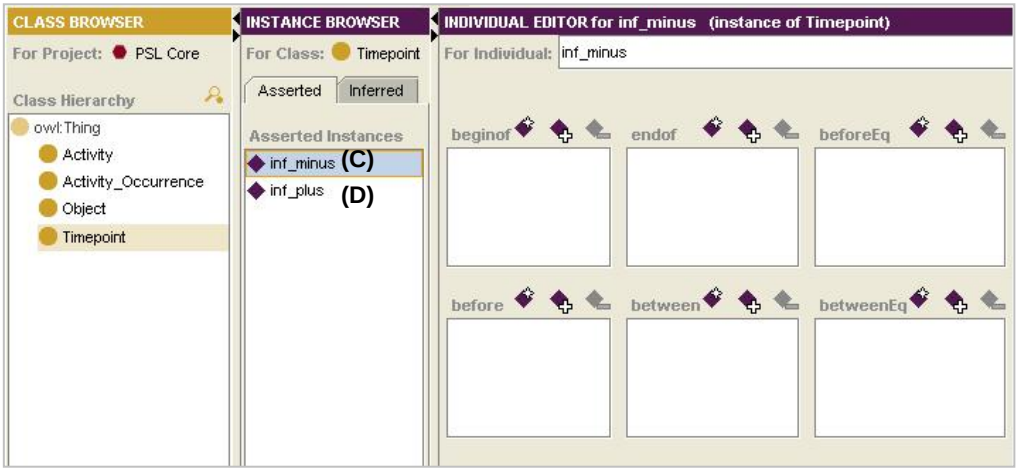


Figure B-9 Capturing Individuals from PSL Core in OWL

B.3.3.6 PSL Core Axioms

Until now, only the modelling of basic ontological structures of the PSL Core ontology have been explained. Heavyweight ontology development involves, apart from basic ontological structures, axioms or rules that are formalised to ensure the semantic integrity of the ontology.

Although OWL can be used to capture some notions of integrity constraints as necessary and necessary and sufficient conditions of classes, the representation of more complex constraints is either not straightforward or cannot be formalised. SWRL, on the other hand, has specifically been developed for adding an extra logic layer to OWL ontologies and to an extent allows more complex rules to be captured where these axioms are written in Horn-type logic. In the Protégé OWL ontology editor, a number of SWRL built-ins have been developed to improve the reasoning infrastructures of OWL ontologies. Documented next is a detailed account of how OWL with SWRL can be used to model the axioms from PSL Core.

Axiom 1 The before relation only holds between timepoints.

The semantics from the logical expression that governs Axiom 1 can readily be satisfied through the specification of the domain and range of the binary relation "before". The domain is the class “Timepoint” as well as the range.

Axiom 2 The before relation is total ordering.

This axiom states that if there are any two timepoints, then in the domain, these two timepoints can either be the same individuals, or take place before each other. OWL on its own cannot be used to specify such an axiom, and, therefore, in this case, SWRL is used to write it. However, this axiom cannot be captured in a single statement in SWRL primarily because SWRL does not support disjunctions of atoms i.e. logical statements involving “or”. So, three SWRL statements have to be written in Expression B-4 in order to capture the single axiom.

```
Timepoint(?t1)  ^ Timepoint(?t2) → sameAs(?t1, ?t2)
Timepoint(?t1)  ^ Timepoint(?t2) → before(?t1, ?t2)
Timepoint(?t1)  ^ Timepoint(?t2)  → before(?t2, ?t1)
```

Expression B-4 SWRL Expression for PSL Core Axiom 2

Axiom 3 The before relation is irreflexive.

This axiom states that an instance of the class “Timepoint” cannot be before itself. In order to capture this axiom in SWRL, the statement has to be worked around to preserve original semantics. This is because SWRL does not support negation as failure in its rules i.e. logical statements involving “not”. The rule listed in Expression B-5 approximates PSL Core Axiom 3 and captures the fact that if one timepoint ?t1 is before a timepoint ?t2, then ?t1 must be different from ?t2.

```
Timepoint(?t1)  ^ Timepoint(?t2)  ^ before(?t1, ?t2) →
differentFrom(?t1, ?t2)
```

Expression B-5 SWRL Expression for PSL Core Axiom 3

Axiom 4 The before relation is transitive.

This axiom can be fully captured in OWL on its own because of its support for relations as having transitive properties.

Axiom 5 The timepoint inf_minus is before all other timepoints.

This axiom has to be slightly worked around in SWRL due to the fact that SWRL does not support negation as failure e.g. to imply that the timepoint ?t is not the individual “inf_minus”, the “tbox:notEqualTo” built-in in used in the expression below (Expression B-6) to convey the same semantics.

```
Timepoint(?t)  ^  tbox:notEqualTo(?t,  inf_minus)  →
before(inf_minus, ?t)
```

Expression B-6 SWRL Expression for PSL Core Axiom 5

Axiom 6 Every other timepoint is before inf_plus.

For the same reason as in Axiom 5, a minor work around results in the SWRL statement listed next (Expression B-7), with original semantics preserved.

```
Timepoint(?t) ^ tbox:notEqualTo(?t, inf_plus) → before(?t, inf_plus)
```

Expression B-7 SWRL Expression for PSL Core Axiom 6

Axiom 7 Given any timepoint ?t other than inf_minus and inf_plus, there is a timepoint between inf_minus and ?t.

The original axiom is used to imply the existence of some timepoint ?u that always lies between the timepoint inf_minus and another timepoint ?t. In OWL, it is not possible to refer to instance values like “inf_minus” within existential restrictions. When SWRL is used to formalise the semantics of Axiom 7, Expression B-8 is captured where the semantics of the SWRL expression differs slightly from the original version because SWRL expressions cannot accommodate existential quantification.

```
Timepoint(?t) ^ Timepoint(?u) ^ tbox:notEqualTo(?t, ?u) ^ tbox:notEqualTo(?t, inf_plus) ^ tbox:notEqualTo(?t, inf_minus) ^ tbox:notEqualTo(?u, inf_plus) ^ tbox:notEqualTo(?u, inf_minus) → between(?u, inf_minus) ^ between(?u, ?t)
```

Expression B-8 SWRL Expression for PSL Core Axiom 7

Axiom 8 Given any timepoint ?t other than inf_plus and inf_minus, there is a timepoint between ?t and inf_plus.

The same problem and partial solution to the problem is encountered in Axiom 8 as in Axiom 7. Expression B-9 listed next exposes the SWRL rule.

```
Timepoint(?t) ^ Timepoint(?u) ^ tbox:notEqualTo(?t, ?u) ^ tbox:notEqualTo(?t, inf_plus) ^ tbox:notEqualTo(?t, inf_minus) ^ tbox:notEqualTo(?u, inf_plus) ^ tbox:notEqualTo(?u, inf_minus) → between(?u, inf_plus) ^ between(?u, ?t)
```

Expression B-9 SWRL Expression for PSL Core Axiom 8

Axiom 9 Everything is either an activity, activity occurrence, timepoint or object.

In the PSL Core theory, Axiom 9 implies that only the classes “Activity”, “Activity_Occurrence”, “Timepoint” and “Object” are instantiable. This can readily be accounted for in OWL.

Axiom 10 Objects, activities, activity occurrences, and timepoints are all distinct kinds of things.

The specification of disjointness among the classes “Activity”, “Activity_Occurrence”, “Timepoint” and “Object” ensures that this axiom is satisfied (see Figure B-10). OWL supports the specification of disjoint classes.

Axiom 11 The occurrence relation only holds between activities and activity occurrences.

The semantics from the logical expression that governs Axiom 11 can readily be satisfied through the specification of the domain and range of the binary relation "occurrence_of", where the domain is the class “Activity_Occurrence” while the range being the class “Activity”.

Axiom 12 Every activity occurrence is an occurrence of some activity.

Although the original axiom involves an existential quantifier, yet, a necessary condition to the class “Activity_Occurrence” can be specified in OWL, and this fully preserves original semantics. The figure next (Figure B-10) identifies how it can be made compulsory that the specification of an instance of the class “Activity_Occurrence” needs to be accompanied by the specification of an “occurrence_of” some instance of “Activity”.

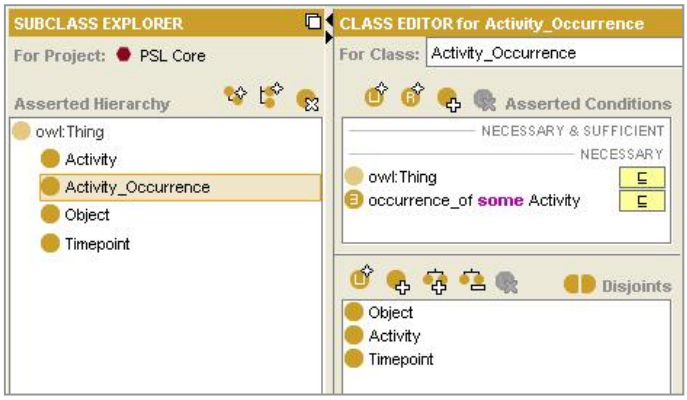


Figure B-10 Adding a Necessary Condition to Capture PSL Core Axiom 12

Axiom 13 An activity occurrence is associated with a unique activity.

By specifying that the "occurrence_of" relation is a functional binary relation, this axiom can be captured in OWL, thereby preserving the semantics that the original Axiom 13 carries.

Axiom 14 The begin and end of an activity occurrence or object are timepoints.

Since OWL does not support the capture of functions, with the “beginof” and “endof” unary functions approximated to binary relations, a specification of the domain of both relations to be the class “Timepoint” with the range being the union of the classes “Activity_Occurrence” and “Object” attempts to capture the semantics of Axiom 14 (see Figure B-8). However, the approximation only provides an acceptable work around of the original axiom.

Axiom 15 The begin point of every activity occurrence is before or equal to its end point.

By treating the unary functions “beginof” and “endof” as relations, the semantics of Axiom 15 can be covered using SWRL. The SWRL statement is identified next (Expression B-10).

Activity_Occurrence(?occ) ^ Timepoint(?t1) ^ Timepoint(?t2) ^ beginof(?t1, ?occ) ^ endof(?t2, ?occ) → beforeEq(?t1, ?t2)

Expression B-10 SWRL Expression for PSL Core Axiom 15

Axiom 16 The participates_in relation only holds between objects, activity occurrences, and timepoints, respectively.

Since the original “participates_in” ternary relation is approximated to a binary relation to allow implementation in OWL Full with SWRL, a specification of the domain and range of the relation ensures that it holds between the three classes “Object”, “Activity_Occurrence” and “Timepoint”. However, the initial ternary relation semantics are lost in the approximation process.

Axiom 17 An object can participate in an activity occurrence only at those timepoints at which both the object exists and the activity is occurring

This axiom informally states that if an object ?x is participating in an activity occurrence ?occ at a timepoint ?t, then it means that ?x exists at this timepoint ?t and that the activity occurrence ?occ is occurring at the same timepoint ?t. With “participates_in” approximated to a binary relation, the full semantics of the axiom can be captured in SWRL as follows (Expression B-11).

Object(?x) ^ Activity_Occurrence(?occ) ^ Timepoint(?t) ^ participates_in(?x, ?occ) ^ participates_in(?x, ?t) → exists_at(?x, ?t) ^ is_occurring_at(?occ, ?t)

Expression B-11 SWRL Expression for PSL Axiom 17

B.3.4 Verification of the OWL with SWRL Model of PSL Core

In order to delimit OWL used in conjunction with SWRL to model the PSL Core ontology, a simple scenario has been explored where a few individuals have been instantiated with some basic fact sentences asserted to these instances. Figure B-11 provides an IDEF5 schematic that depicts all the individuals defined with all relations asserted among them.

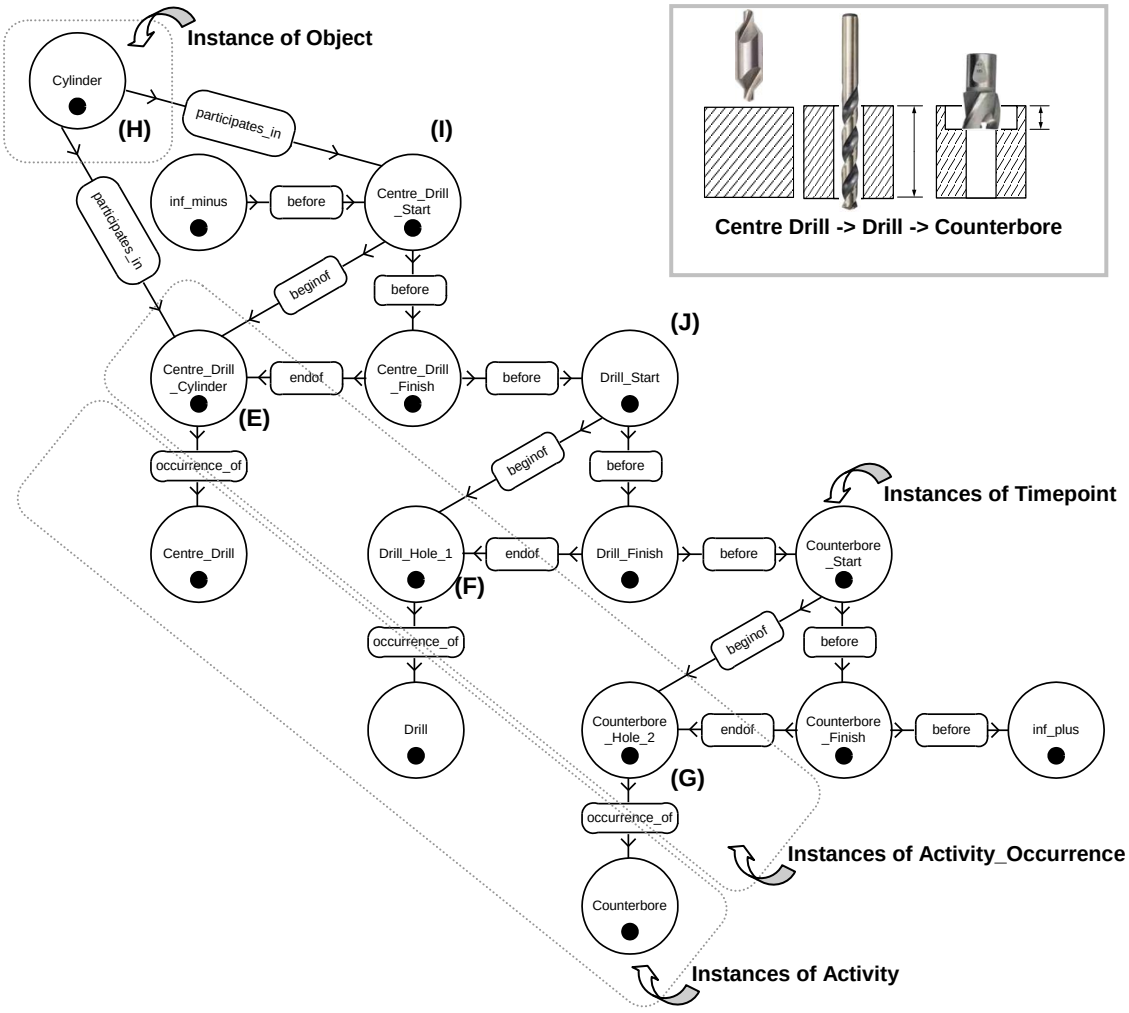


Figure B-11 IDEF5 Schematic of Asserted Instances in the OWL with SWRL-Formalised PSL Core Ontology

The scenario highlighted is based on a typical machining process sequence for the creation of a standard counterbore hole on a cylindrical part. The sequence informally consists of an execution of “Centre_Drill” **(E)** followed by an execution of “Drill” **(F)** followed by an execution of “Counterbore” **(G)**. The instance “Cylinder” **(H)** initially participates in “Centre_Drill_Cylinder” **(E)** at the timepoint “Centre_Drill_Start” **(I)**. Each activity occurrence is then sequentially carried out.

B.3.4.1 Expected Results

Based on the scenario identified in Figure B-11, it is clear that certain key results are expected on running SWRL rules that attempt to model the relevant PSL axioms. In this section, three of these axioms are considered although during the actual experiment, all axioms have been evaluated. The three axioms are:

- Axiom 5: The timepoint inf_minus is before all other timepoints.
- Axiom 8: Given any timepoint ?t other than inf_plus and inf_minus, there is a timepoint between ?t and inf_plus.
- Axiom 17: An object can participate in an activity occurrence only at those timepoints at which both the object exists and the activity is occurring.

On running the SWRL rule that models Axiom 5, it is expected that the rule engine would identify that the timepoint “inf_minus”, is before all the defined timepoints that have been instantiated e.g. logically, *before(inf_minus, Counterbore_Finish)*, although this fact has not been asserted in the first place. Similarly, on evaluating Axiom 8, the inference engine should be able to depict a series of inferred facts, for example, *between(Drill_Start, Centre_Drill_Finish)* and *between(Drill_Start, Counterbore_Start)*. Note that the original “between” ternary relation has been approximated to a binary relation in the OWL-based PSL Core ontology.

While the expected list of results on running axioms 5 and 8 is likely to consist of more than six derived facts due to the different combinations of

possibilities, the expected result on running Axiom 17 should consist of exactly two facts derived from the asserted information. The facts should include: *exists_at(Cylinder, Centre_Drill_Start) and is_occurring_at(Centre_Drill_Cylinder, Centre_Drill_Start)*.

B.3.4.2 Actual Results

Figure B-12 next illustrates some of the results obtained after the rule engine has been executed for PSL Core axioms 5, 8 and 17 formalised in SWRL. All the results retrieved for axioms 5 and 17 have been shown in the figure. Due to an extensive list of 36 results that has been obtained on running Axiom 8, only a subset of these results containing six inferred facts over the timepoint "Drill_Start" **(J)** (see Figure B-11) has been shown.

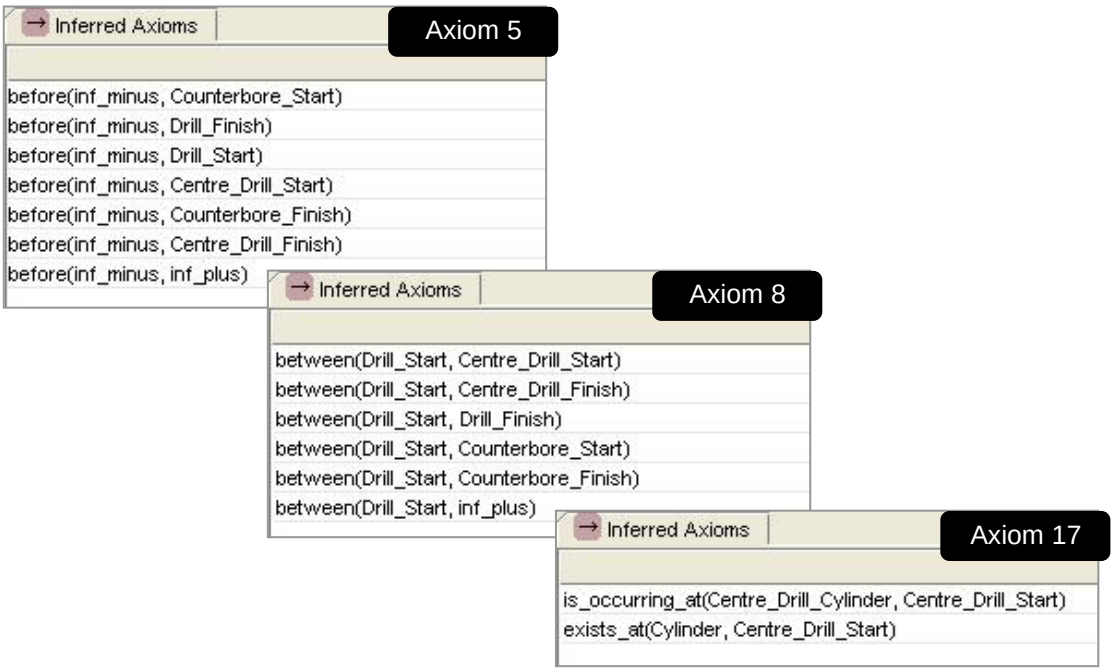


Figure B-12 Sample Results Obtained from the Evaluation of PSL Core Axioms Formalised in SWRL

B.3.5 Discussions

The actual results related to the execution of SWRL rules that model PSL Core axioms all agree with the expected results. This implies that, the semantics carried by SWRL rules can be used to infer new consequent

knowledge from existing asserted facts in a heavyweight ontology of process semantics. Furthermore, SWRL in Protégé OWL is accompanied by a set of more than 220 built-ins including built-ins for comparison, maths built-ins, built-ins for querying an OWL TBox and many more (SWRL Built-ins, 2009). These predefined SWRL built-ins can readily be exploited by the user to formulate different types of rules, for instance, in this current exploration, the "tbox:notEqualTo" built-in has been used to distinguish two separate instances in SWRL expressions.

During the execution of SWRL rules, it is important that the user runs them one at a time. However, if a SWRL rule has to be run and the antecedent of that rule involves the consequent of another SWRL rule, then both rules have to be executed concurrently. At one stage of the experiment, all SWRL rules that model axioms in PSL Core were simultaneously processed. This not only resulted in an extensive and confusing list of more than 150 inferred facts, but also led to unexpected behaviours and some incorrectly-derived facts. When SWRL rules were run individually or in small batches, this problem did not occur. The initial competency question set in section B.3.1 is answered next.

- Is OWL with SWRL sufficiently expressive to support the representation of PSL-based process semantics?

OWL used in conjunction with SWRL increases the logic expressiveness of Description Logic-based approaches to model heavyweight manufacturing ontologies. SWRL is highly effective as a rule language to drive knowledge inferences and with a competent rule engine, it compensates for the lower ability that OWL reasoners currently have to infer information over instances of classes. However, SWRL is not purposely a constraint language and, therefore, its support for PSL Core axioms used as integrity constraints falls slightly behind. In the experiment, it has nevertheless been shown that it is possible to infer from SWRL rules that attempt to model PSL Core axioms. Additionally, it is the user's task to ensure that axioms are properly identified in an ontology thereby ascertaining the correct way to interpret the derived facts after executing rules.

OWL with SWRL as a heavyweight ontological formalism is not able to capture higher-arity relations present in the PSL Core theory, although workarounds are possible. However, having to approximate higher-arity relations and ontological functions to binary relations inevitably leads to a loss of original semantics. Such an approximation can lead to ambiguously-defined instances and the issue is inevitably carried forward to the SWRL logic layer, thereby producing incorrectly-derived facts. Thus, it can be extrapolated that OWL with SWRL is sufficiently robust to support heavyweight semantics as long as these structures do not involve higher-arity relations and functions (in the ontological sense). Unfortunately, to meet the requirements of the Semantic Manufacturing Interoperability Framework in this work, it is evident that a more powerful formalism is required to address the formal semantics of the PSL ontology.

B.4 Motivation for a Common Logic-Based Ontological Formalism

From a semantic point of view, it has been demonstrated that there is one major issue in relation to Frames with a first order constraint language and OWL with a rule language as possible ontological formalisms to be used within the framework. This issue is concerned with the inability of these two knowledge representation formalisms to fully capture and represent the semantics from the Process Specification Language (PSL). Since PSL constitutes a fundamental element of the framework concept, it is thus necessary to identity a suitable ontological formalism, which helps support the semantic needs throughout the four layers of the framework.

Based on an understanding of the explored ontological formalisms, it is possible to extrapolate that Common Logic-based formalisms are favoured. 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). Furthermore, the PSL ontology is available in a

number of first order formats including the Common Logic Interchange Format (CLIF) (PSL Website, 2009). This implies that in order to replicate the exact semantics of PSL from its CLIF form, it is necessary to identify a suitable Common Logic-based formalism that is either completely CLIF-based or has equal semantic potentials to CLIF.

After careful scrutiny, it was decided that the Knowledge Framework Language (KFL) developed by Ontology Works Inc. (Ontology Works Inc., 2009) constitutes an ideal candidate. KFL is a Common Logic-based ontological formalism that provides expressive logic in which to encode the subject matter ontology (Ontology Works Inc., 2009). Broadly speaking, Common Logic is a logical framework intended for information exchange and transmission and has some novel features, chief among them being a syntax that is signature-free, while preserving a first-order model theory (ISO/IEC 24707, 2007). This clearly implies that KFL as an ontological formalism is able to provide the necessary syntax and expressive first-order semantics for developing the heavyweight manufacturing ontological foundation as well as to support the semantic considerations needed in the other layers of the SMIF.

B.5 Summary

The arguments discussed in this appendix have revealed that the ability to support the semantic needs of the ontology-based Semantic Manufacturing Interoperability Framework (SMIF) is directly dependent on the choice of ontological formalism. This is particularly significant for allowing PSL-based process semantics to be fully captured and exploited in the framework. The choice of the Knowledge Framework Language (KFL) used in this work has been justified based on an exploration of the capabilities and limitations of two other known heavyweight ontological methods (sections B.2 and B.3) and an assessment of the acknowledged benefits of Common Logic (section B.4).

In the first place, an investigation of Frames with a first order constraint language (Protégé Frames and Protégé Axiom Language) has been carried

out. A sample “Machining Hole Feature Ontology” regrouping different viewpoints across product design and manufacture has been tested to reveal the ability of the formalism to model simple entity information semantics, process semantics and semantic relationships between entities and processes. The main conclusion derived from this experiment has pointed towards important limitations of the ontological formalism for capturing and representing PSL-type process semantics.

This has constructively led to a second experiment, which this time uses the formalism OWL with SWRL, to attempt at maximising the formal heavyweight representation of PSL semantics. The exploration involving OWL with SWRL has shown that this particular formalism, as part of Semantic Web technologies, is not rigorous enough to model PSL semantics. Furthermore, it has become evident that several workarounds and approximations need to be made, which lead to a loss of original PSL semantics. Thus, the second exploration has been a turning point for enabling the identification of a suitable ontological formalism with enhanced expressivity, capable of replicating higher-arity relations and ontological functions from PSL.

A brief account of the key benefits of Common Logic, in addition to a view on the resources available for research purposes, have decisively pointed towards KFL as best-fit ontological formalism. Hence, throughout the four levels of the SMIF explained in chapters 5 and 6, Common Logic-based KFL is exploited to provide the syntax and first order semantics necessary for the specification of relevant concepts, definitions and integrity constraints.

C Foundation Layer

Context Declaration

:Ctx Foundation
:Inst UserContext
:supCtx TopUserCtx
:name "Foundation Context"
:rem "This context enfolds the Process Specification Language (ISO 18629), and adapted concepts from ISO 10303 AP224 and the Core Product Model developed by NIST."

:Use Foundation

C.1 Process Specification Language (PSL)

C.1.1 PSL Core

The purpose of PSL Core is to axiomatise a set of intuitive semantic primitives that is adequate for describing the fundamental concepts of manufacturing processes. It is based on the following intuitions (PSL Website, 2009):

- There are four kinds of entities required for reasoning about processes namely activities, activity occurrences, timepoints, and objects.
- Activities may have multiple occurrences, or there may exist activities that do not occur at all.
- Timepoints are linearly ordered, forwards into the future, and backwards into the past.
- Activity occurrences and objects are associated with unique timepoints that mark the begin and end of the occurrence or object.

The following set of figures capture the IDEF5 schematics for the concepts present in PSL and the coding used during implementation. A list of the relevant implemented axioms is also displayed. Figure C-1 illustrates the initial organisation of PSL Core classes. Notice the class “Origin” which provides a root class for defining the taxonomy and is only present in order to keep the taxonomy tidy.

Classes

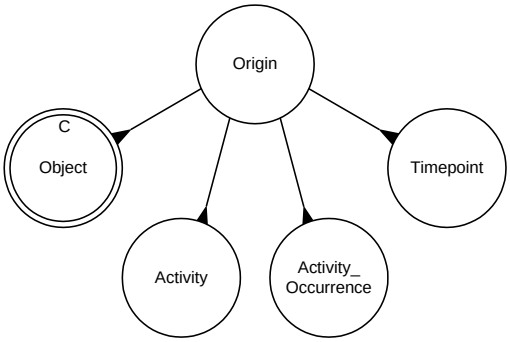


Figure C-1 PSL Core Classes

:Prop **Origin**
:Inst Property
:sup Top
:name "Origin"
:rem "This abstract class is at the root of the taxonomy of the concepts explored in the Foundation Layer."

:Prop **Object**
:Inst Property
:sup Origin
:name "Object"
:rem "(Object ?x) is TRUE in an interpretation of the Foundation Layer if and only if ?x is a member of the set of objects in the universe of discourse of the interpretation. An object is anything that is not a timepoint, nor an activity nor an activity-occurrence. Intuitively, an object is a concrete or abstract thing that can participate in an activity. Objects can come into existence and go out of existence at certain points in time. In such cases, an object has a begin and an end point. In some contexts it may be useful to consider some ordinary objects as having no such points either."

:Prop **Activity**
:Inst Property
:sup Origin
:name "Activity"
:rem "(Activity ?a) is TRUE in an interpretation of the Foundation Layer if and only if ?a is a member of the set of activities in the universe of discourse of the interpretation. Intuitively, activities can be considered to be reusable behaviours within the domain."

:Prop **Activity_Occurrence**
:Inst Property
:sup Origin
:name "Activity Occurrence"
:rem "(Activity_Occurrence ?occ) is TRUE in an interpretation of the Foundation Layer if and only if ?occ is a member of the set of activity occurrences in the universe of discourse of the interpretation. An activity occurrence is associated with a unique activity and begins and ends at specific points in time. Although there may exist activities that have no activity occurrence, all activity occurrences must be associated with an activity."

:Prop **Timepoint**
:Inst Property
:sup Origin
:name "Timepoint"
:rem "(Timepoint ?t) is TRUE in an interpretation of the Foundation Layer if and only if ?t is a member of the set of timepoints in the universe of discourse of the interpretation. Timepoints form an infinite linear ordering with endpoints at infinity."

Relations

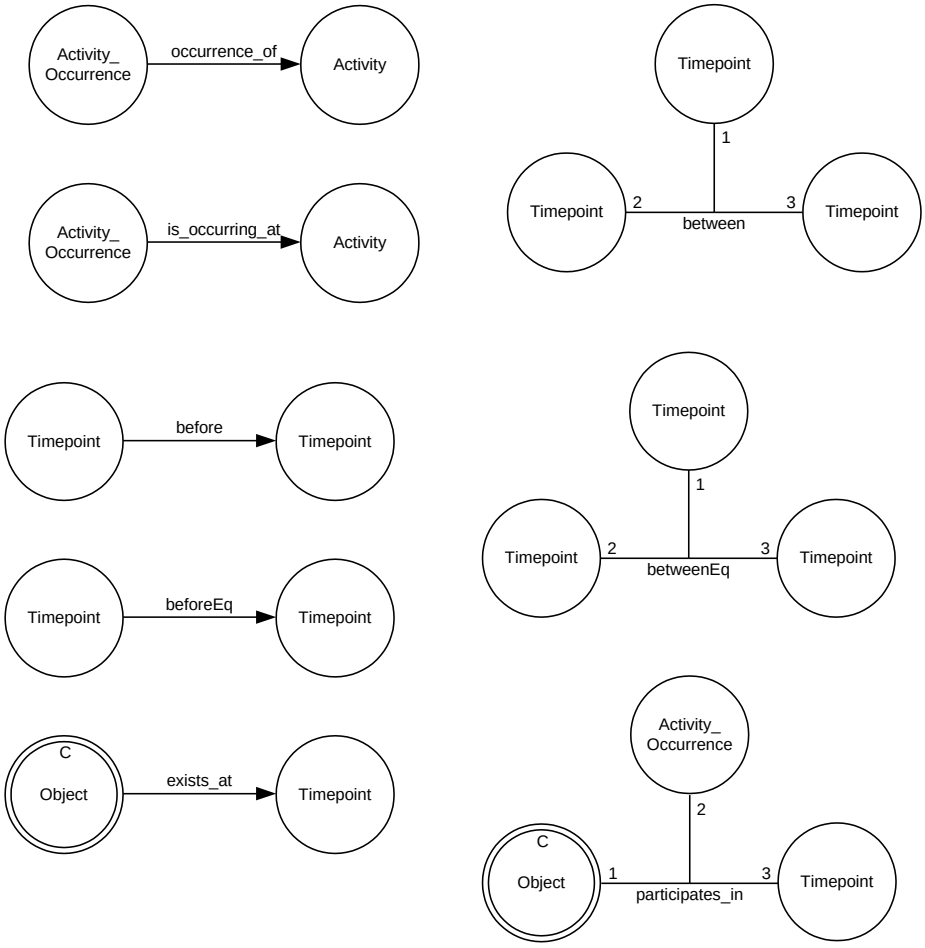


Figure C-2 PSL Core Relations

:Rel **before**
:Inst BinaryRel
:Inst IrreflexiveBR ;;; Axiom 3
:Inst TransitiveBR ;;; Axiom 4
:Sig Timepoint Timepoint
:name "before"
:rem "(before ?t1 ?t2) is TRUE in an interpretation of the Foundation Layer if and only if the timepoint ?t1 is earlier than ?t2 in the linear ordering over timepoints in the interpretation."

:Rel **occurrence_of**
:Inst BinaryRel
:Sig Activity_Occurrence Activity
:name "occurrence_of"
:rem "(occurrence_of ?occ ?a) is TRUE in an interpretation of the Foundation Layer if and only if ?occ is a particular occurrence of the activity ?a. occurrence_of is the basic relation between activities and activity occurrences. Every activity occurrence is associated with a unique activity. An activity may have no occurrences or multiple occurrences."

:Rel **participates_in**
:Inst TernaryRel
:Sig Object Activity_Occurrence Timepoint
:name "participates_in"

:rem "(participates_in ?x ?occ ?t) is TRUE in an interpretation of the Foundation Layer if and only if ?x plays some role that is not pre-specified in an occurrence of the activity occurrence ?occ at the timepoint ?t in the interpretation. An object can participate in an activity occurrence only at those timepoints at which both the object exists and the activity is occurring."

:Rel **between**
:Inst TernaryRel
:Sig Timepoint Timepoint Timepoint
:name "between"
:rem "(between ?t2 ?t1 ?t3) is TRUE in an interpretation of the Foundation Layer if and only if ?t1 is strictly less than ?t3 and strictly greater than ?t2 in the linear ordering over timepoints in the interpretation."

:Rel **beforeEq**
:Inst BinaryRel
:Sig Timepoint Timepoint
:name "beforeEq"
:rem "(beforeEq ?t1 ?t2) is TRUE in an interpretation of the Foundation Layer if and only if ?t1 is less or equal to ?t2 in the linear ordering over timepoints in the interpretation."

:Rel **betweenEq**
:Inst TernaryRel
:Sig Timepoint Timepoint Timepoint
:name "betweenEq"
:rem "(betweenEq ?t2 ?t1 ?t3) is TRUE in an interpretation of the Foundation Layer if and only if ?t1 is less or equal to ?t3 and greater or equal to ?t2 in the linear ordering over timepoints in the interpretation."

:Rel **exists_at**
:Inst BinaryRel
:Sig Object Timepoint
:name "exists_at"
:rem "The object exists at the given timepoint."

:Rel **is_occurring_at**
:Inst BinaryRel
:Sig Activity_Occurrence Timepoint
:name "is_occurring_at"
:rem "The specified activity occurrence is occurring at the specified timepoint."

Functions

:Fun **beginof**
:Inst UnaryFun
:Sig Activity_Occurrence -> Timepoint
:name "beginof"
:rem "If ?x is an activity occurence in the universe of discourse of an interpretation of the Foundation Layer, then (beginof ?x) has the value ?t if and only if ?t is the timepoint at which the activity occurrence ?x begins. If ?x is an object in the universe of discourse of an interpretation of the Foundation Layer, then (beginof ?x) has the value ?x if and only if ?t is the timepoint at which the object ?x becomes possible to participate in an activity."

:Fun **endof**
:Inst UnaryFun
:Sig Activity_Occurrence -> Timepoint
:name "endof"

:rem "If ?x is an activity occurrence in the universe of discourse of an interpretation of the Foundation Layer, then (endof ?x) has the value ?x if and only if ?t is the timepoint at which the activity occurrence ?x ends. If ?x is an object in the universe of discourse of an interpretation of the Foundation Layer, then (endof ?x) has the value ?x if and only if ?t is the timepoint at which the object ?x becomes no longer possible to participate in an activity."

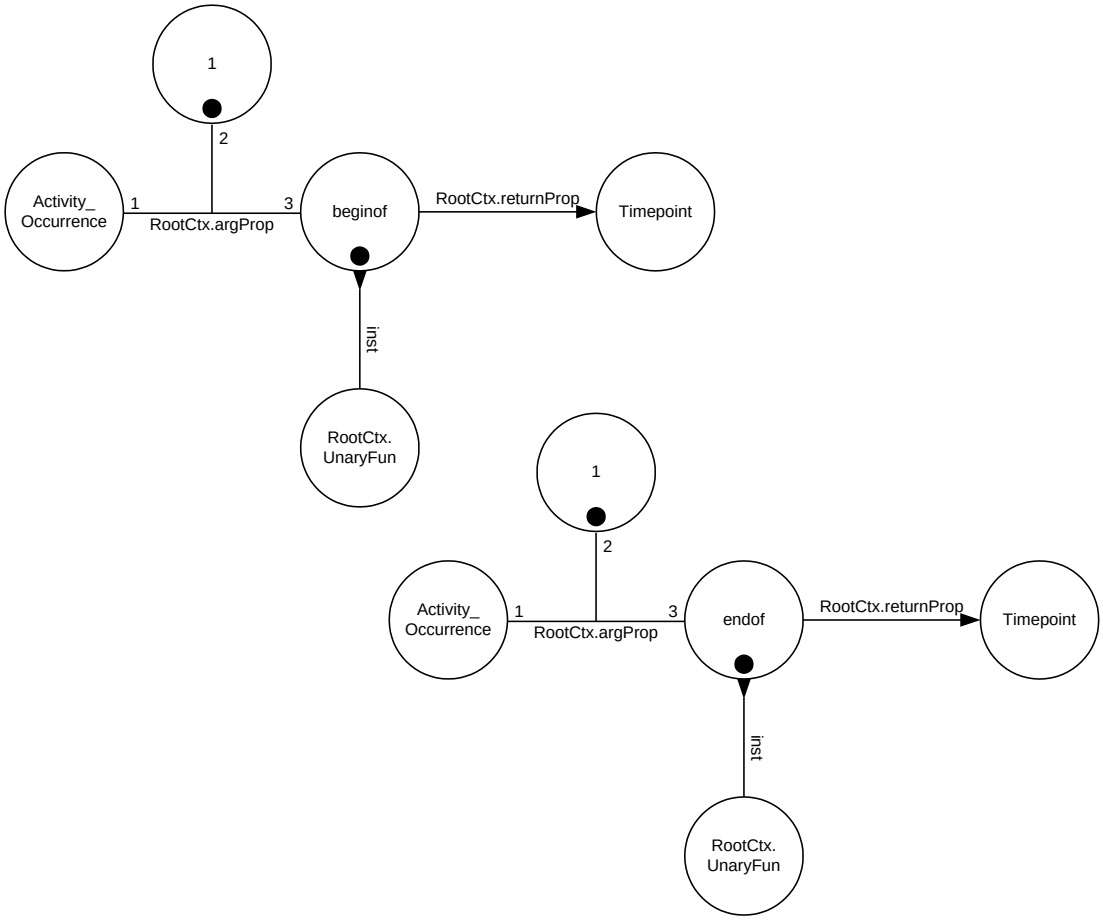


Figure C-3 PSL Core Functions

Individuals

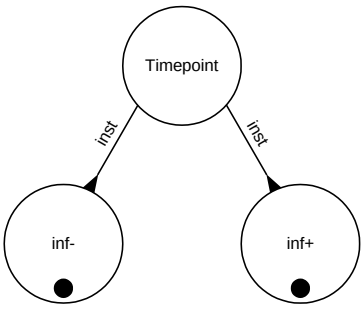


Figure C-4 PSL Core Individuals

(Timepoint Foundation.**inf-**)
(RootCtx.rem Foundation.inf- "(= ?t inf-) is TRUE in an interpretation of the Foundation Layer if and only if ?t is the unique timepoint that is before all other timepoints in the linear ordering

over timepoints in the universe of discourse of the interpretation. inf- plays the role of negative infinity. It is needed to specify objects that have not been created.")

(Timepoint Foundation.**inf+**)
(RootCtx.rem Foundation.inf+ "(= ?t inf+) is TRUE in an interpretation of the Foundation Layer if and only if ?t is the unique timepoint that is after all other timepoints in the linear ordering over timepoints in the universe of discourse of the interpretation. inf+ plays the role of positive infinity. It is needed to specify objects that are never destroyed.")

Axioms

```
(=> (Foundation.before ?t1 ?t2)
    (and (Timepoint ?t1)
         (Timepoint ?t2)))
:IC hard "The before relation only holds between timepoints."

(=> (and (Timepoint ?t)
         (/= ?t Foundation.inf-))
    (Foundation.before Foundation.inf- ?t))
:IC weak "The timepoint inf- is before all other timepoints."

(=> (and (Timepoint ?t)
         (/= ?t Foundation.inf+))
    (Foundation.before ?t Foundation.inf+))
:IC weak "Every other timepoint is before inf+."

(or (Activity ?x)
    (Activity_Occurrence ?x)
    (Timepoint ?x)
    (Object ?x))
:IC hard "Everything is either an activity, activity occurrence, timepoint or object."

(and (=> (Activity ?x)
         (not (or (Activity_Occurrence ?x) (Object ?x) (Timepoint ?x))))
     (=> (Activity_Occurrence ?x)
         (not (or (Object ?x) (Timepoint ?x))))
     (=> (Object ?x)
         (not (Timepoint ?x))))
:IC hard "Objects, activities, activity occurrences, and timepoints are all distinct kinds of things."

(=> (occurrence_of ?occ ?a)
    (and (Activity ?a)
         (Activity_Occurrence ?occ)))
:IC hard "The occurrence relation only holds between activities and activity occurrences."

(=> (Activity_Occurrence ?occ)
    (exists (?a)
        (and (Activity ?a)
             (occurrence_of ?occ ?a))))
:IC hard "Every activity occurrence is an occurrence of some activity."

(=> (and (Activity ?a1)
         (Activity ?a2)
         (Activity_Occurrence ?occ)
         (occurrence_of ?occ ?a1)
```

```
            (occurrence_of ?occ ?a2))
       (= ?a1 ?a2))
```
:IC hard "**An activity occurrence is associated with a unique activity.**"

```
(=> (and (Activity_Occurrence ?occ)
            (Activity ?a)
            (occurrence_of ?occ ?a))
     (and (Timepoint (beginof ?occ))
            (Timepoint (endof ?occ))))
```
:IC hard "The **begin and end of an activity occurrence are timepoints.**"

```
 (=> (and (Activity_Occurrence ?occ)
            (Timepoint (beginof ?occ))
            (Timepoint (endof ?occ)))
      (beforeEq (beginof ?occ) (endof ?occ)))
```
:IC hard "The **begin point of every activity occurrence is before or equal to its end point.**"

```
 (=> (participates_in ?x ?occ ?t)
      (and (Object ?x)
            (Activity_Occurrence ?occ)
            (Timepoint ?t)))
```
:IC hard "**The participates_in relation only holds between objects, activity occurrences, and timepoints, respectively.**"

```
 (=> (and (Object ?x)
            (Activity_Occurrence ?occ)
            (Timepoint ?t)
            (participates_in ?x ?occ ?t))
      (and (exists_at ?x ?t)
            (is_occurring_at ?occ ?t)))
```
:IC hard "**An object can participate in an activity occurrence only at those timepoints at which both the object exists and the activity is occurring.**"

Definitions

```
(<= (between ?t1 ?t2 ?t3)
     (and (Timepoint ?t1)
            (Timepoint ?t2)
            (Timepoint ?t3)
            (Foundation.before ?t1 ?t2)
            (Foundation.before ?t2 ?t3)))
```
:rem "**Timepoint ?t2 is between timepoints ?t1 and ?t3 if and only if ?t1 is before ?t2 and ?t2 is before ?t3.**"

```
 (<= (beforeEq ?t1 ?t2)
     (and (Timepoint ?t1)
            (Timepoint ?t2)
            (or (Foundation.before ?t1 ?t2)
            (= ?t1 ?t2))))
```
:rem "**Timepoint ?t1 is beforeEq Timepoint ?t2 if and only if ?t1 is before or equal to ?t2.**"

```
 (<= (betweenEq ?t1 ?t2 ?t3)
      (and (Timepoint ?t1)
            (Timepoint ?t2)
            (Timepoint ?t3)
```

```
            (beforeEq ?t1 ?t2)
            (beforeEq ?t2 ?t3)))
```
:rem "**Timepoint ?t2 is between or equal to timepoints ?t1 and ?t3 if and only if ?t1 is before or equal to ?t2, and ?t2 is before or equal to ?t3.**"

```
 (<= (exists_at ?x ?t)
      (and (Object ?x)
            (Timepoint (beginof ?x))
            (Timepoint (endof ?x))
            (Timepoint ?t)
            (betweenEq (beginof ?x) ?t (endof ?x))))
```
:rem "**An object exists at a timepoint ?t if and only if ?t is between or equal its begin and end points.**"

```
(<= (is_occurring_at ?occ ?t)
     (and (Activity_Occurrence ?occ)
            (Timepoint (beginof ?occ))
            (Timepoint (endof ?occ))
            (Timepoint ?t)
            (betweenEq (beginof ?occ) ?t (endof ?occ))))
```
:rem "**An activity is occurring at a timepoint t1 if and only if t1 is between or equal to the activity occurrence's begin and end points.**"

C.1.2 PSL Outer-Core

PSL Outer-Core introduces new terminology and concepts that extend PSL Core in order to provide more logical expressiveness to PSL semantics.

C.1.2.1 Theory of Subactivities

Relations

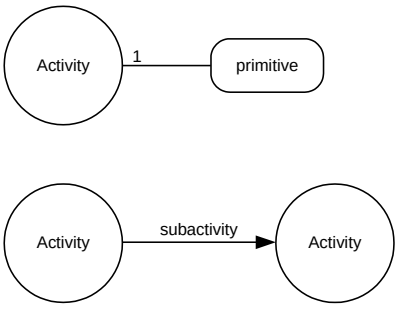


Figure C-5 Theory of Subactivities Relations

:Rel **subactivity**
:Inst BinaryRel
:Inst PartialOrderBR ;;; Axioms 2,3 and 4
:Sig Activity Activity
:name "subactivity"
:rem "(subactivity ?a1 ?a2) is TRUE in an interpretation of the Foundation Layer if and only if activity ?a1 is a subactivity of activity ?a2. The subactivity relation forms a discrete partial ordering over the set of activities."

:Rel **primitive**
:Inst UnaryRel
:Sig Activity
:name "primitive"
:rem "(primitive ?a) is TRUE in an interpretation of the Foundation Layer if and only if the activity ?a has no subactivities except for itself."

Axioms

```
(=> (subactivity ?a1 ?a2)
    (and (Activity ?a1)
         (Activity ?a2)))
```
:IC hard "**subactivity is a relation over activities.**"

```
(=> (and (Activity ?a1)
         (Activity ?a2)
         (/= ?a1 ?a2)
         (subactivity ?a1 ?a2))
    (exists (?a3)
        (and (Activity ?a3)
             (/= ?a3 ?a1)
             (subactivity ?a1 ?a3)
             (subactivity ?a3 ?a2))))
```
:IC hard "**The subactivity relation is a discrete ordering, so every activity has a downwards successor in the ordering.**"

```
(=> (and (Activity ?a1)
         (Activity ?a2)
         (/= ?a1 ?a2)
         (Activity ?a4)
         (subactivity ?a1 ?a2)
         (subactivity ?a1 ?a4)
         (subactivity ?a4 ?a3)
    (exists (?a3)
        (and (Activity ?a3)
             (/= ?a3 ?a1)
             (subactivity ?a1 ?a3)
             (subactivity ?a3 ?a2))))
             (or (= ?a4 ?a1)
                 (= ?a4 ?a3)))
```
:IC hard "**The subactivity relation is a discrete ordering, so every activity has a downwards successor in the ordering.**"

```
(=> (and (Activity ?a1)
         (Activity ?a2)
         (/= ?a1 ?a2)
         (subactivity ?a1 ?a2))
    (exists (?a3)
        (and (Activity ?a3)
             (/= ?a3 ?a2)
             (subactivity ?a1 ?a3)
             (subactivity ?a3 ?a2))))
```
:IC hard "**The subactivity relation is a discrete ordering, so every activity has an upwards successor in the ordering.**"

```
(=> (and (Activity ?a1)
         (Activity ?a2)
         (/= ?a1 ?a2)
         (Activity ?a4)
```

```
        (subactivity ?a1 ?a2)
        (subactivity ?a3 ?a4)
        (subactivity ?a4 ?a2)
    (exists (?a3)
        (and (Activity ?a3)
            (/= ?a3 ?a2)
            (subactivity ?a1 ?a3)
            (subactivity ?a3 ?a2))))
            (or (= ?a4 ?a2)
                (= ?a4 ?a3)))
```

:IC hard "**The subactivity relation is a discrete ordering, so every activity has an upwards successor in the ordering.**"

Definitions

```
 (<= (primitive ?a)
     (and (Activity ?a)
          (Activity ?a1)
          (subactivity ?a1 ?a)
          (= ?a1 ?a)
      (not (exists (?a2)
                   (and (Activity ?a2)
                        (subactivity ?a2 ?a)
                        (/= ?a2 ?a1)
                        (/= ?a2 ?a))))))
```

:rem "**An activity is primitive if and only if it has no subactivities except for itself.**"

C.1.2.2 Theory of Occurrence Trees

Relations

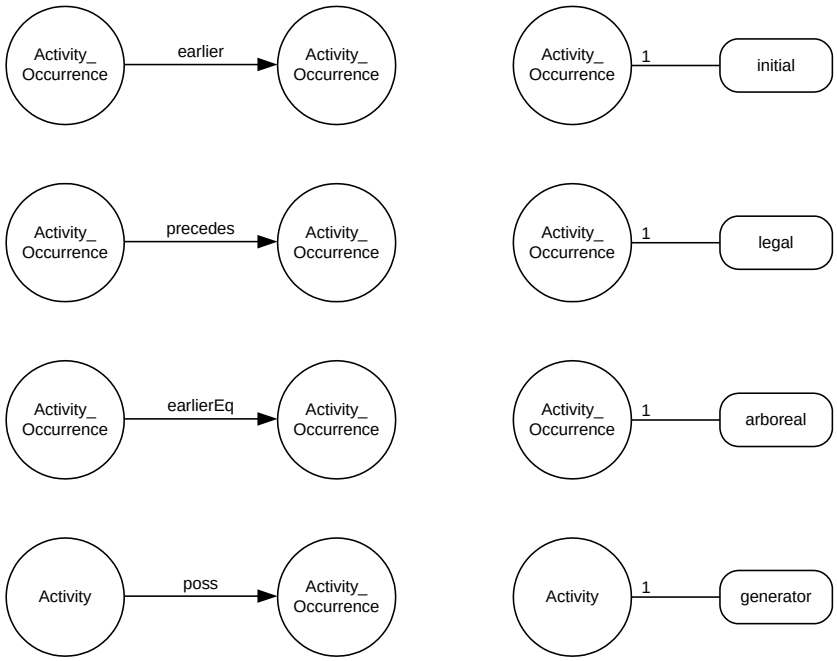


Figure C-6 Theory of Occurrence Trees Relations

:Rel **earlier**
:Inst BinaryRel

:Inst IrreflexiveBR ;;; Axiom 2
:Inst TransitiveBR ;;; Axiom 3
:Sig Activity_Occurrence Activity_Occurrence
:name "earlier"
:rem "(earlier ?occ1 ?occ2) is TRUE in an interpretation of the Foundation Layer if and only if the two activity occurrences ?occ1 and ?occ2 are on the same branch of the tree and ?occ1 is closer to the root of the tree than ?occ2. In interpretations of Occurrence Trees, the set of all sequences of activity occurrences forms a tree; the earlier relation specifies the partial ordering over the activity occurrences in this tree."

:Rel **initial**
:Inst UnaryRel
:Sig Activity_Occurrence
:name "initial"
:rem "(initial ?occ) is TRUE in an interpretation of the Foundation Layer if and only if the activity occurrence ?occ is the root of an occurrence tree."

:Rel **legal**
:Inst UnaryRel
:Sig Activity_Occurrence
:name "legal"
:rem "(legal ?occ) is TRUE in an interpretation of the Foundation Layer if and only if the activity occurrence ?occ is an element of the legal occurrence tree."

:Rel **precedes**
:Inst BinaryRel
:Sig Activity_Occurrence Activity_Occurrence
:name "precedes"
:rem "(precedes ?occ1 ?occ2) is TRUE in an interpretation of the Foundation Layer if and only if the activity occurrence ?occ1 is earlier than the activity occurrence ?occ2 in the occurrence tree and such that all activity occurrences between them correspond to activities that are possible. This relation specifies the sub-tree of the occurrence tree in which every activity occurrence is the occurrence of an activity that is possible."

:Rel **earlierEq**
:Inst BinaryRel
:Sig Activity_Occurrence Activity_Occurrence
:name "earlierEq"
:rem "(earlierEq ?occ1 ?occ2) is TRUE in an interpretation of the Foundation Layer if and only if the two activity occurrences ?occ1 and ?occ2 are on the same branch of the tree and ?occ1 is closer to the root of the tree than ?occ2, or ?occ1 and ?occ2 are the same activity occurrences."

:Rel **poss**
:Inst BinaryRel
:Sig Activity Activity_Occurrence
:name "poss"
:rem "(poss ?a ?occ2) is TRUE in an interpretation of the Foundation Layer if and only if the activity ?a has a legal occurrence that is a successor of the activity occurrence ?occ in the occurrence tree."

:Rel **generator**
:Inst UnaryRel
:Sig Activity
:name "generator"
:rem "(generator ?a) is TRUE in an interpretation of the Occurrence Tree Theory if and only if ?a is an activity whose occurrences are elements of the occurrence tree."

:Rel **arboreal**
:Inst UnaryRel

:Sig Activity_Occurrence
:rem "(arboreal ?s) is TRUE in an interpretation of the Occurrence Tree Theory if and only if ?s is an element of the occurrence tree."

Functions

:Fun successor
:Inst BinaryFun
:Sig Activity Activity_Occurrence -> Activity_Occurrence
:name "successor"
:rem "(= ?occ2 (successor ?a ?occ1)) is TRUE in an interpretation of the Occurrence Tree Theory if and only if ?occ2 denotes the occurrence of ?a that follows consecutively after the activity occurrence ?occ in the occurrence tree."

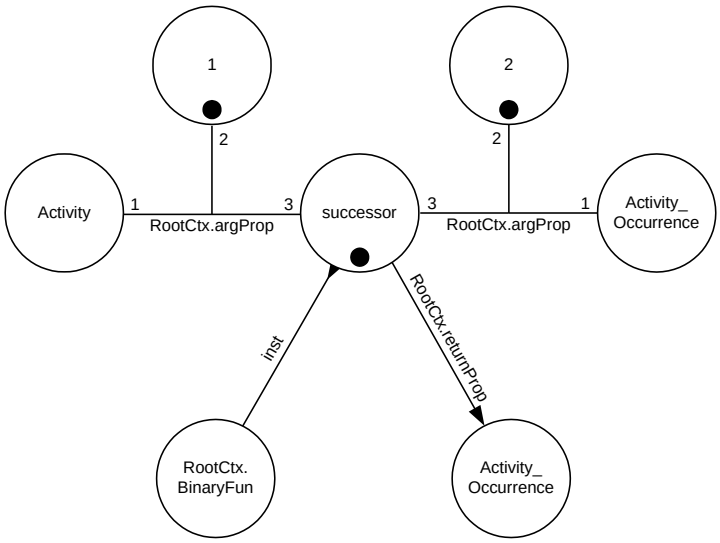


Figure C-7 Theory of Occurrence Trees Functions

Axioms

```
(=> (earlier ?s1 ?s2)
    (and (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)))
```

:IC hard "**The earlier relation is restricted to arboreal activity occurrences (that is, activity occurrences that are elements of the occurrence tree).**"

```
(=> (and (Activity_Occurrence ?s)
         (initial ?s))
    (and (arboreal ?s)
         (not (exists (?sp)
                  (and (Activity_Occurrence ?sp)
                       (earlier ?sp ?s))))))
```

:IC hard "**No occurrence in the occurrence tree is earlier than an initial occurrence.**"

```
(=> (and (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (earlier ?s1 ?s2))
    (exists (?sp)
```

```
        (and (Activity_Occurrence ?sp)
             (initial ?sp)
             (earlierEq ?sp ?s1))))
```
:IC hard "**Every branch of the occurrence tree has an initial occurrence.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s)
         (occurrence_of ?s ?a)
         (generator ?a))
    (arboreal ?s))
```
:IC hard "**There is an initial occurrence of each activity.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (initial ?s1)
         (initial ?s2)
         (occurrence_of ?s1 ?a)
         (occurrence_of ?s2 ?a))
    (= ?s1 ?s2))
```
:IC hard "**No two initial activity occurrences in the occurrence tree are occurrences of the same activity.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?o)
         (Activity_Occurrence (successor ?a ?o))
         (occurrence_of (successor ?a ?o) ?a))
    (and (generator ?a)
         (arboreal ?o)))
```
:IC hard "**The successor of an arboreal activity occurrence is an occurrence of a generator activity.**"

```
(=> (and (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (earlier ?s1 ?s2))
    (exists (?a ?s3)
        (and (Activity ?a)
             (Activity_Occurrence ?s3)
             (generator ?a)
             (= ?s2 (successor ?a ?s3)))))
```
:IC weak "**Every non-initial activity occurrence is the successor of another activity occurrence.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (Activity_Occurrence (successor ?a ?s2))
         (generator ?a)
         (earlierEq ?s1 ?s2))
    (earlier ?s1 (successor ?a ?s2)))
```
:IC hard "**An occurrence ?s1 is earlier than the successor occurrence of ?s2 if and only if the occurrence ?s2 is later than ?s1.**"

```
(=> (and (Activity_Occurrence ?s)
         (legal ?s))
    (arboreal ?s))
```
:IC hard "**The legal relation restricts arboreal activity occurrences.**"

```
(=> (and (Activity_Occurrence ?s1)
```

```
            (Activity_Occurrence ?s2)
            (legal ?s1)
            (earlier ?s2 ?s1))
      (legal ?s2))
```
:IC hard "**If an activity occurrence is legal, all earlier activity occurrences in the occurrence tree are also legal.**"

```
 (=> (and (Activity_Occurrence ?s1)
            (Activity_Occurrence ?s2)
            (earlier ?s1 ?s2))
      (and (Timepoint (beginof ?s2))
            (Timepoint (endof ?s1))
            (Foundation.before (endof ?s1) (beginof ?s2))))
```
:IC hard "**The endof an activity occurrence is before the beginof the successor of the activity occurrence.**"

Definitions

```
 (<= (precedes ?s1 ?s2)
      (and (Activity_Occurrence ?s1)
            (Activity_Occurrence ?s2)
            (earlier ?s1 ?s2)
            (legal ?s2)))
```
:rem "**An activity occurrence ?s1 precedes another activity occurrence ?s2 if and only if ?s1 is earlier than ?s2 in the occurrence tree and ?s2 is legal.**"

```
 (<= (earlierEq ?s1 ?s2)
      (and (Activity_Occurrence ?s1)
            (Activity_Occurrence ?s2)
            (arboreal ?s1)
            (arboreal ?s2)
            (or (earlier ?s1 ?s2)
                (= ?s1 ?s2))))
```
:rem "**An activity occurrence ?s1 is earlierEq than an activity occurrence ?s2 if and only if it is either earlier than ?s2 or it is equal to ?s2.**"

```
(<= (poss ?a ?s)
     (and (Activity ?a)
           (Activity_Occurrence ?s)
           (Activity_Occurrence (successor ?a ?s))
           (legal (successor ?a ?s))))
```
:rem "**An activity is poss at some occurrence if and only if the successor occurrence of the activity is legal.**"

```
 (<= (generator ?a)
      (and (Activity ?a)
            (exists (?s)
               (and (Activity_Occurrence ?s)
                     (initial ?s)
                     (occurrence_of ?s ?a)))))
```
:rem "**An activity is a generator if and only if it has an initial occurrence in the occurrence tree.**"

```
 (<= (arboreal ?s)
      (and (Activity_Occurrence ?s)
            (exists (?sp)
               (and (Activity_Occurrence ?sp)
                     (earlier ?s ?sp)))))
```

:rem "**An activity occurrence is arboreal if and only if it is an element of an occurrence tree.**"

C.1.2.3 Theory of Discrete States

Classes

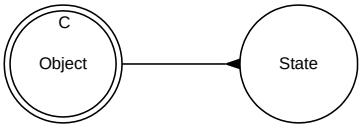


Figure C-8 Theory of Discrete State Classes

:Prop **State**
:Inst Type
:sup Object
:name "State"
:rem "(state ?f) is TRUE in an interpretation of the Foundation Layer if and only if ?f is a member of the set of states in the universe of discourse of the interpretation. States are a subcategory of object. They intuitively represent properties and relationships in the domain that can change as the result of the occurrence of activities."

Relations

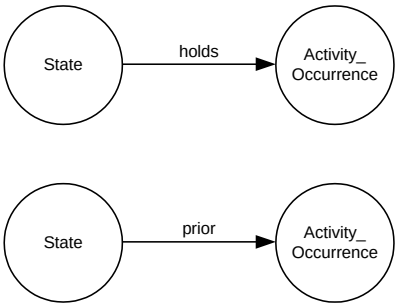


Figure C-9 Theory of Discrete States Relations

:Rel **holds**
:Inst BinaryRel
:Sig State Activity_Occurrence
:name "holds"
:rem "(holds ?f ?occ) is TRUE in an interpretation of the Foundation Layer if and only if the state ?f is true after the activity occurrence ?occ."

:Rel **prior**
:Inst BinaryRel
:Sig State Activity_Occurrence
:name "prior"
:rem "(prior ?f ?occ) is TRUE in an interpretation of the Foundation Layer if and only if the state ?f is true prior to the activity occurrence ?occ."

Axioms

```
 (=> (State ?f)
      (Object ?f))
:IC hard "States are objects."

(=> (holds ?f ?occ)
     (and (State ?f)
          (Activity_Occurrence ?occ)
          (arboreal ?occ)))
:IC hard "The holds relation is only between states and arboreal activity occurrences."

 (=> (prior ?f ?occ)
      (and (State ?f)
           (Activity_Occurrence ?occ)
           (arboreal ?occ)))
:IC hard "The prior relation is only between states and arboreal activity occurrences."

 (=> (and (Activity_Occurrence ?occ1)
          (Activity_Occurrence ?occ2)
          (initial ?occ1)
          (initial ?occ2)
          (State ?f)
          (prior ?f ?occ2))
      (prior ?f ?occ1))
:IC hard "All initial occurrences agree on the states that hold prior to them."

(=> (and (State ?f)
         (Activity_Occurrence ?occ)
         (Activity ?a)
         (Activity_Occurrence (successor ?a ?occ))
         (holds ?f ?occ)
         (generator ?a))
     (prior ?f (successor ?a ?occ)))
:IC hard "A state holds after an arboreal activity occurrence if and only if it holds prior to the successor occurrence."

 (=> (and (State ?f)
          (Activity_Occurrence ?occ1)
          (holds ?f ?occ1))
      (exists (?occ2)
           (and (Activity_Occurrence ?occ2)
                (earlierEq ?occ2 ?occ1)
                (holds ?f ?occ2)
                (or (initial ?occ2)
                     (not (prior ?f ?occ2))))))
:IC hard "If a fluent holds after some activity occurrence, then there exists an earliest activity occurrence along the branch where the fluent holds."

(=> (and (State ?f)
         (Activity_Occurrence ?occ1)
         (holds ?f ?occ1)
         (exists (?occ2)
              (and (Activity_Occurrence ?occ2)
                   (earlierEq ?occ2 ?occ1)
                   (holds ?f ?occ2)
                   (or (initial ?occ2)
                   (not (prior ?f ?occ2)))))
                   (Activity_Occurrence ?occ3)
                   (earlier ?occ2 ?occ3)
```

```
                    (earlier ?occ3 ?occ1))
      (holds ?f ?occ3))
```
:IC hard "**If a fluent holds after some activity occurrence, then there exists an earliest activity occurrence along the branch where the fluent holds.**"

```
 (=> (and (State ?f)
          (Activity_Occurrence ?occ1)
          (arboreal ?occ1)
          (not (holds ?f ?occ1)))
      (exists (?occ2)
          (and (Activity_Occurrence ?occ2)
               (earlierEq ?occ2 ?occ1)
               (not (holds ?f ?occ2))
               (or (initial ?occ2)
                   (prior ?f ?occ2)))))
```
:IC hard "**If a fluent does not hold after some activity occurrence, then there exists an earliest activity occurrence along the branch where the fluent does not hold.**"

```
(=> (and (State ?f)
          (Activity_Occurrence ?occ1)
          (arboreal ?occ1)
          (not (holds ?f ?occ1))
          (exists (?occ2)
                  (and (Activity_Occurrence ?occ2)
                       (earlierEq ?occ2 ?occ1)
                       (not (holds ?f ?occ2))
                       (or (initial ?occ2)
                           (prior ?f ?occ2))))
                       (Activity_Occurrence ?occ3)
                       (earlier ?occ2 ?occ3)
                       (earlier ?occ3 ?occ1))
       (holds ?f ?occ3))
```
:IC hard "**If a fluent does not hold after some activity occurrence, then there exists an earliest activity occurrence along the branch where the fluent does not hold.**"

```
 (=> (and (State ?f)
          (Activity_Occurrence ?s1)
          (holds ?f ?s1))
      (exists (?s2)
          (and (Activity_Occurrence ?s2)
               (holds ?f ?s2)
               (earlierEq ?s2 ?s1))))
```
:IC hard "**If a fluent holds, there exists an earliest activity occurrence where it holds.**"

```
(=> (and (State ?f)
          (Activity_Occurrence ?s1)
          (holds ?f ?s1)
          (Activity_Occurrence ?s2)
          (holds ?f ?s2)
          (earlierEq ?s2 ?s1)
          (Activity_Occurrence ?s3)
          (holds ?f ?s3))
      (not (earlier ?s3 ?s2)))
```
:IC hard "**If a fluent holds, there exists an earliest activity occurrence where it holds**."

C.1.2.4 Theory of Atomic Activities

Relations

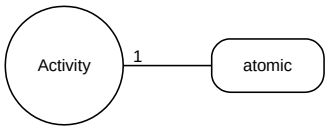


Figure C-10 Theory of Atomic Activities Relations

:Rel **atomic**
:Inst UnaryRel
:Sig Activity
:name "atomic"
:rem "(atomic ?a) is TRUE in an interpretation of the Foundation Layer if and only if either ?a is primitive or it is the concurrent superposition of a set of primitive activities."

Functions

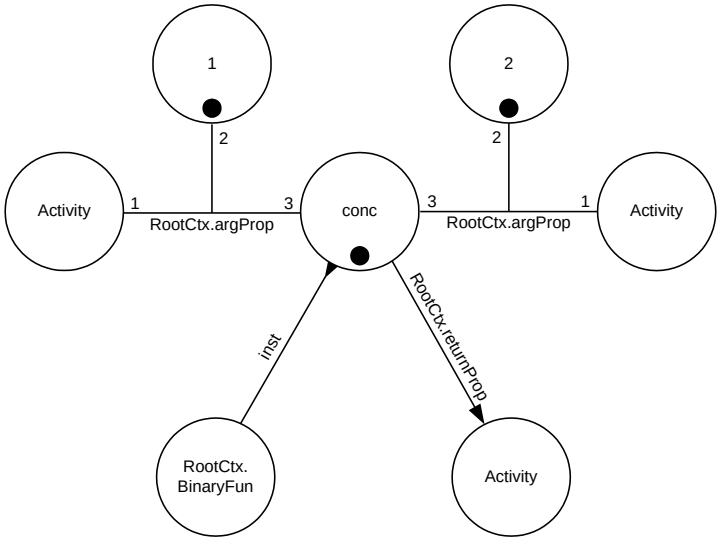


Figure C-11 Theory of Atomic Activities Functions

:Fun **conc**
:Inst BinaryFun
:Sig Activity Activity -> Activity
:name "conc"
:rem "(= ?a3 (conc ?a1 ?a2)) is TRUE in an interpretation of the Foundation Layer if and only if ?a3 is the atomic activity that is the concurrent superposition of the two atomic activities ?a1 and ?a2."

Axioms

```
 (=> (and (Activity ?a)
          (primitive ?a))
     (atomic ?a))
```
:IC hard "**Primitive activities are atomic.**"

```
(=> (and (Activity ?a)
         (Activity ?a1)
         (Activity (conc ?a ?a))
         (= ?a1 (conc ?a ?a)))
    (= ?a ?a1)) ;;; Work Around
:IC weak "The function conc is idempotent."

(= (conc ?a1 ?a2) (conc ?a2 ?a1))
:IC weak "The function conc is commutative."

(and (= ?a4 (conc ?a2 ?a3))
     (= ?a5 (conc ?a1 ?a2))
     (= (conc ?a1 ?a4) (conc ?a5 ?a3))) ;;; Work Around
:IC weak "The function conc is associative."

(=> (and (Activity ?a1)
         (Activity ?a2)
         (Activity ?a3)
         (atomic ?a1)
         (atomic ?a2)
         (= ?a3 (conc ?a1 ?a2))
         (= ?a2 ?a3))
    (subactivity ?a1 ?a2)) ;;; Work Around
:IC hard "An atomic activity ?a1 is a subactivity of an atomic activity ?a2 if and only if ?a2 is an idempotent for ?a1."

(=> (and (Activity ?a1)
         (Activity ?a2)
         (atomic ?a2)
         (subactivity ?a1 ?a2)
         (/= ?a1 ?a2))
    (exists (?a3)
        (and (Activity ?a3)
             (atomic ?a3)
             (= ?a2 (conc ?a1 ?a3))
             (not (exists (?a4)
                      (and (Activity ?a4)
                           (atomic ?a4)
                           (subactivity ?a4 ?a1)
                           (subactivity ?a4 ?a3)))))))
:IC hard "An atomic action has a proper subactivity if and only if there exists another atomic activity which can be concurrently aggregated."

(=> (and (Activity ?a)
         (Activity ?b0)
         (Activity ?b1)
         (atomic ?a)
         (atomic ?b0)
         (atomic ?b1)
         (subactivity ?a (conc ?b0 ?b1))
         (not (primitive ?a)))
    (exists (?a0 ?a1)
        (and (Activity ?a0)
             (Activity ?a1)
             (subactivity ?a0 ?a)
             (subactivity ?a1 ?a)
             (= ?a (conc ?a0 ?a1)))))
:IC hard "The semilattice of atomic activities is distributive."
```

(=> (and (Activity ?a)
(generator ?a))
(atomic ?a))
:IC hard "**Only atomic activities can be generator activities. Equivalently, only occurrences of atomic activities can be elements of an occurrence tree.**"

(=> (atomic ?a)
(Activity ?a))
:IC hard "**Atomic activities are activities.**"

C.1.2.5 Theory of Complex Activities

Relations

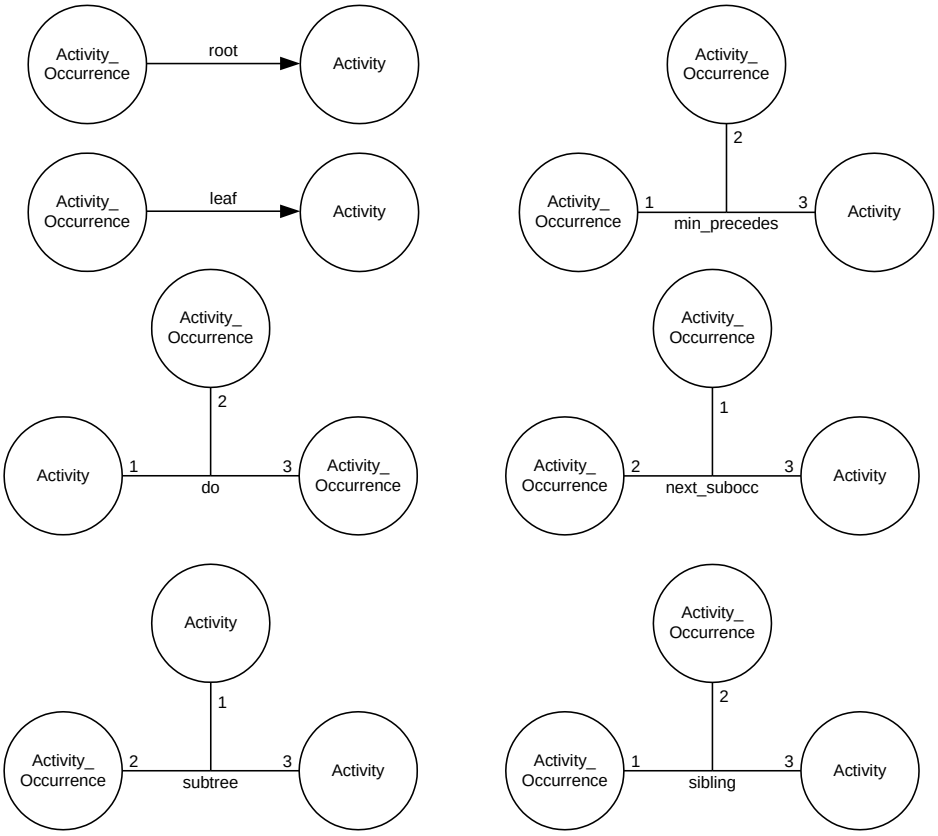


Figure C-12 Theory of Complex Activities Relations

:Rel **min_precedes**
:Inst TernaryRel
:Sig Activity_Occurrence Activity_Occurrence Activity
:name "min_precedes"
:rem "(min_precedes ?s1 ?s2 ?a) is TRUE in an interpretation of the Foundation Layer if and only if ?s1 and ?s2 are subactivity occurrences in the activity tree for ?a, and ?s1 precedes ?s2 in the subtree. Any occurrence of an activity ?a corresponds to an activity tree (which is a subtree of the occurrence tree). The activity occurrences within this subtree are the subactivity occurrences of the occurrence of ?a."

:Rel **root**
:Inst BinaryRel
:Sig Activity_Occurrence Activity
:name "root"

:rem "(root ?s ?a) is TRUE in an interpretation of the Foundation Layer if and only if the activity occurrence ?s is the root of an activity tree for ?a."

:Rel **do**
:Inst TernaryRel
:Sig Activity Activity_Occurrence Activity_Occurrence
:name "do"
:rem "(do ?a ?s1 ?s2) is TRUE in an interpretation of the Foundation Layer if and only if ?s1 is the root of an activity tree and ?s2 is a leaf of the same activity tree such that both activity occurrences are elements of the same branch of the activity tree."

:Rel **leaf**
:Inst BinaryRel
:Sig Activity_Occurrence Activity
:name "leaf"
:rem "(leaf ?s ?a) is TRUE in an interpretation of the Foundation Layer if and only if the activity occurrence ?s is the leaf of an activity tree for ?a."

:Rel **next_subocc**
:Inst TernaryRel
:Sig Activity_Occurrence Activity_Occurrence Activity
:name "next_subocc"
:rem "(next_subocc ?s1 ?s2 ?a) is TRUE in an interpretation of the Foundation Layer if and only if ?s1 precedes ?s2 in the tree and there does not exist a subactivity occurrence that is between them in the tree."

:Rel **subtree**
:Inst TernaryRel
:Sig Activity_Occurrence Activity Activity
:name "subtree"
:rem "(subtree ?s ?a1 ?a2) is TRUE in an interpretation of the Foundation Layer if and only if every atomic subactivity occurrence in the activity tree for ?a1 is an element of the activity tree for ?a2."

:Rel **sibling**
:Inst TernaryRel
:Sig Activity_Occurrence Activity_Occurrence Activity
:name "sibling"
:rem "(sibling ?s1 ?s2 ?a) is TRUE in an interpretation of the Foundation Layer if and only if the atomic subactivity occurrences ?s1 and ?s2 are siblings in an activity tree for ?a where they either have a common predecessor in the activity tree or they are both roots of activity trees that have a common predecessor in the occurrence tree."

Axioms

```
(=> (and (Activity ?a)
 (Activity_Occurrence ?s1)
          (Activity_Occurrence ?s2)
          (min_precedes ?s1 ?s2 ?a))
     (exists (?a1 ?a2)
          (and (Activity ?a1)
               (Activity ?a2)
               (atomic ?a2)
               (subactivity ?a1 ?a)
               (subactivity ?a1 ?a2)
               (occurrence_of ?s1 ?a2))))
```

:IC hard "**Occurrences in the activity tree for an activity correspond to atomic subactivity occurrences of the activity.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (min_precedes ?s1 ?s2 ?a))
    (exists (?a2 ?a3)
        (and (Activity ?a2)
             (Activity ?a3)
             (atomic ?a3)
             (subactivity ?a2 ?a)
             (subactivity ?a2 ?a3)
             (occurrence_of ?s2 ?a3))))
```
:IC hard "**Occurrences in the activity tree for an activity correspond to atomic subactivity occurrences of the activity.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s)
         (root ?s ?a))
    (exists (?a2 ?a3)
        (and (Activity ?a2)
             (Activity ?a3)
             (atomic ?a3)
             (subactivity ?a2 ?a3)
             (subactivity ?a2 ?a)
             (occurrence_of ?s ?a3))))
```
:IC hard "**Root occurrences in the activity tree correspond to atomic subactivity occurrences of the activity.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (min_precedes ?s1 ?s2 ?a))
    (exists (?s3)
        (and (Activity_Occurrence ?s3)
             (root ?s3 ?a)
             (min_precedes ?s3 ?s2 ?a))))
```
:IC hard "**All activity trees have a root subactivity occurrence.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (min_precedes ?s1 ?s2 ?a))
    (not (root ?s2 ?a)))
```
:IC hard "**No subactivity occurrences in an activity tree occur earlier than the root subactivity occurrence.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (min_precedes ?s1 ?s2 ?a))
    (precedes ?s1 ?s2))
```
:IC hard "**An activity tree is a subtree of the occurrence tree.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s)
         (root ?s ?a))
    (legal ?s))
```
:IC hard "**Root occurrences are elements of the occurrence tree.**"

```
(=> (and (Activity ?a1)
         (Activity ?a2)
```

```
            (atomic ?a1)
            (subactivity ?a2 ?a1)
            (/= ?a2 ?a1)
            (Activity_Occurrence ?s)
            (occurrence_of ?s ?a1)
            (legal ?s))
      (root ?s ?a2))
```
:IC hard "**Every legal atomic activity occurrence is an activity tree containing only one occurrence.**"

```
 (=> (and (Activity ?a)
            (Activity_Occurrence ?s1)
            (Activity_Occurrence ?s2)
            (min_precedes ?s1 ?s2 ?a))
      (exists (?s3)
          (and (Activity_Occurrence ?s3)
               (next_subocc ?s1 ?s3 ?a))))
```
:IC hard "**Activity trees are discrete.**"

```
(=> (and (Activity ?a)
            (Activity_Occurrence ?s1)
            (Activity_Occurrence ?s2)
            (Activity_Occurrence ?s3)
            (min_precedes ?s1 ?s2 ?a)
            (min_precedes ?s1 ?s3 ?a)
            (precedes ?s2 ?s3))
      (min_precedes ?s2 ?s3 ?a))
```
:IC hard "**Subactivity occurrences on the same branch of the occurrence tree are on the same branch of the activity tree.**"

```
(=> (and (Activity ?a1)
            (Activity ?a1)
            (Activity_Occurrence ?s)
            (subtree ?s ?a1 ?a2))
     (not (subactivity ?a2 ?a1)))
```
:IC hard "**The activity tree for a complex subactivity occurrence is a subtree of the activity tree for the activity occurrence.**"

```
 (=> (and (Activity ?a)
            (Activity_Occurrence ?s1)
            (Activity_Occurrence ?s2)
            (min_precedes ?s1 ?s2 ?a))
      (not (atomic ?a)))
```
:IC hard "**Only complex activities can be arguments to the min_precedes relation.**"

```
 (=> (and (Activity ?a)
            (Activity_Occurrence ?s1)
            (Activity_Occurrence ?s2)
            (Activity_Occurrence ?s3)
            (min_precedes ?s2 ?s1 ?a)
            (min_precedes ?s3 ?s1 ?a)
            (precedes ?s2 ?s3))
      (min_precedes ?s2 ?s3 ?a))
```
:IC hard "**Subactivity occurrences on the same branch of the activity tree are linearly ordered by the min_precedes relation.**"

Definitions

```
(<= (leaf ?s ?a)
    (and (Activity ?a)
         (Activity_Occurrence ?s)
         (or (root ?s ?a)
         (exists (?s1)
            (and (Activity_Occurrence ?s1)
                 (min_precedes ?s1 ?s ?a)
                 (not (exists (?s2)
                 (and (Activity_Occurrence ?s2)
                      (min_precedes ?s ?s2 ?a)))))))))
```

:rem "**An occurrence is the leaf of an activity tree if and only if there exists an earlier atomic subactivity occurrence but there does not exist a later atomic subactivity occurrence.**"

```
(<= (do ?a ?s1 ?s2)
    (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (root ?s1 ?a)
         (leaf ?s2 ?a)
         (or (min_precedes ?s1 ?s2 ?a)
            (= ?s1 ?s2))))
```

:rem "**The do relation specifies the initial and final atomic subactivity occurrences of an occurrence of an activity.**"

```
(<= (next_subocc ?s1 ?s2 ?a)
    (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (min_precedes ?s1 ?s2 ?a)
         (not (exists (?s3)
              (and (Activity_Occurrence ?s3)
                   (min_precedes ?s1 ?s3 ?a)
                   (min_precedes ?s3 ?s2 ?a))))))
```

:rem "**An activity occurrence ?s2 is the next subactivity occurrence after ?s1 in an activity tree for ?a if and only of ?s1 precedes ?s2 in the tree and there does not exist a subactivity occurrence that is between them in the tree.**"

```
(<= (subtree ?s1 ?a1 ?a2)
    (and (Activity ?a1)
         (Activity_Occurrence ?s1)
         (root ?s1 ?a1)
         (exists (?s2 ?s3)
            (and (Activity_Occurrence ?s2)
                 (Activity_Occurrence ?s3)
                 (root ?s2 ?a2)
                 (min_precedes ?s1 ?s2 ?a1)
                 (min_precedes ?s1 ?s3 ?a1)
                 (not (min_precedes ?s2 ?s3 ?a2))))))
```

:rem "**The activity tree for ?a1 with root occurrence ?s1 contains an activity tree for ?a2 as a subtree if and only if every atomic subactivity occurrence in the activity tree for ?a2 is an element of the activity tree for ?a1, and there is an atomic subactivity occurrence in the activity tree for ?a1 that is not in the activity tree for ?a2.**"

```
(<= (sibling ?s1 ?s2 ?a)
    (and (Activity ?a)
         (Activity_Occurrence ?s1)
```

```
(Activity_Occurrence ?s2)
(or (exists (?s3)
(and (Activity_Occurrence ?s3)
      (next_subocc ?s3 ?s1 ?a)
      (next_subocc ?s3 ?s2 ?a)))
      (and (root ?s1 ?a)
            (root ?s2 ?a)
            (or (and (initial ?s1)
                     (initial ?s2))
            (exists (?s4 ?a1 ?a2)
                  (and (Activity ?a1)
                        (Activity ?a2)
                        (Activity_Occurrence ?s4)
                        (= ?s1 (successor ?a1 ?s4))
                        (= ?s2 (successor ?a2 ?s4)))))))))
```

:rem "**The atomic subactivity occurrences ?s1 and ?s2 are siblings in an activity tree for ?a if and only if they either have a common predecessor in the activity tree or they are both roots of activity trees that have a common predecessor in the occurrence tree.**"

C.1.2.6 Theory of Activity Occurrences

Relations

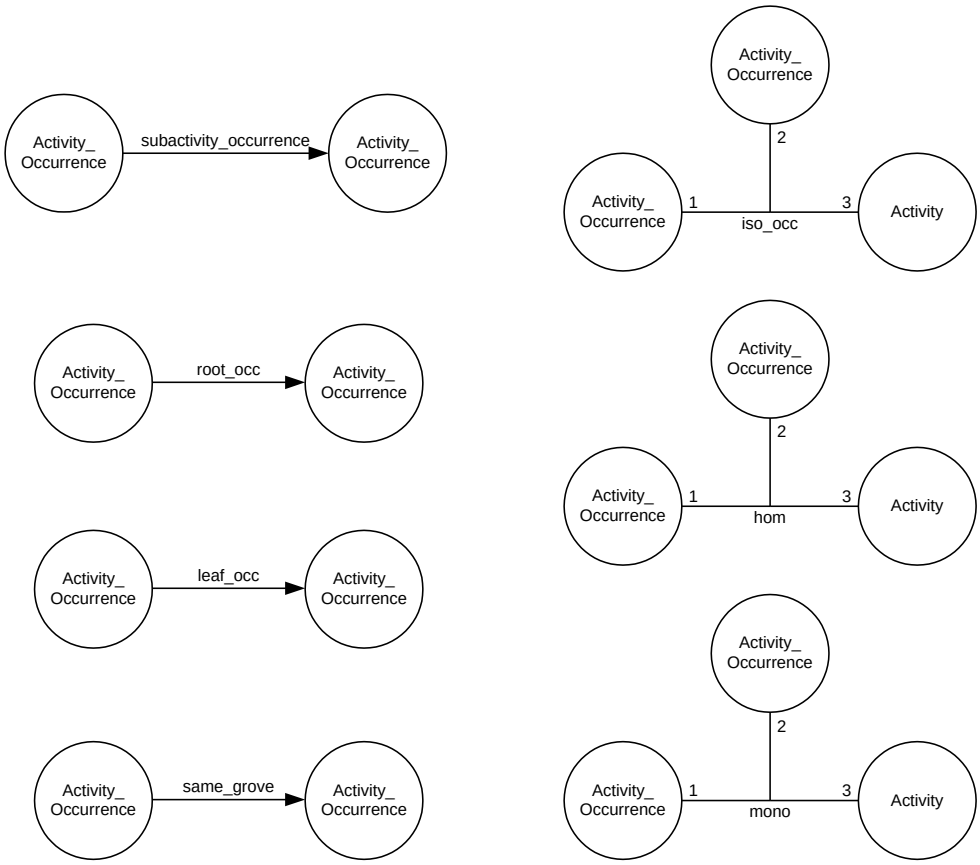


Figure C-13 Theory of Activity Occurrences Relations

:Rel **subactivity_occurrence**
:Inst BinaryRel
:Inst TransitiveBR ;;; Axiom 9
:Sig Activity_Occurrence Activity_Occurrence
:name "subactivity_occurrence"
:rem "(subactivity_occurrence ?occ1 ?occ2) is TRUE in an interpretation of the Foundation Layer if and only if the branch corresponding to the activity occurrence ?occ1 is a subset of the branch corresponding to activity occurrence ?occ2."

:Rel **mono**
:Inst TernaryRel
:Sig Activity_Occurrence Activity_Occurrence Activity
:name "mono"
:rem "(mono ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Foundation Layer if and only if there is a one-to-one mapping between branches of an activity tree for ?a that maps the atomic subactivity occurrence ?s1 to the atomic subactivity occurrence ?s2."

:Rel **iso_occ**
:Inst TernaryRel
:Sig Activity_Occurrence Activity_Occurrence Activity
:name "iso_occ"
:rem "(iso_occ ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Foundation Layer if and only if both ?occ1 and ?occ2 are occurrences of an atomic activity that contain a common subactivity."

:Rel **hom**
:Inst TernaryRel
:Sig Activity_Occurrence Activity_Occurrence Activity
:name "hom"
:rem "(hom ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Foundation Layer if and only if there is a mapping between branches of an activity tree for ?a that maps the atomic subactivity occurrence ?s1 to the atomic subactivity occurrence ?s2."

:Rel **root_occ**
:Inst BinaryRel
:Sig Activity_Occurrence Activity_Occurrence
:name "root_occ"
:rem "(root_occ ?occ1 ?occ2) is TRUE in an interpretation of the Foundation Layer if and only if activity occurrence ?occ1 is the root occurrence in the branch of the activity tree for ?a corresponding to the activity occurrence ?occ2."

:Rel **leaf_occ**
:Inst BinaryRel
:Sig Activity_Occurrence Activity_Occurrence
:name "leaf_occ"
:rem "(leaf_occ ?occ1 ?occ2) is TRUE in an interpretation of the Foundation Layer if and only if activity occurrence ?occ1 is the leaf occurrence in the branch of the activity tree for ?a corresponding to the activity occurrence ?occ2."

:Rel **same_grove**
:Inst BinaryRel
:Sig Activity_Occurrence Activity_Occurrence
:name "same_grove"
:rem "(same_grove ?occ1 ?occ2) is TRUE in an interpretation of the Foundation Layer if and only if activity occurrences ?occ1 and ?occ2 of ?a correspond to branches in the same activity tree for ?a."

Axioms

```
(=> (subactivity_occurrence ?o1 ?o2)
    (and (Activity_Occurrence ?o1)
         (Activity_Occurrence ?o2)))
```
:IC hard "**The subactivity_occurrence relation is between activity occurrences.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (min_precedes ?s1 ?s2 ?a))
    (exists (?occ)
        (and (Activity_Occurrence ?occ)
             (occurrence_of ?occ ?a)
             (subactivity_occurrence ?s1 ?occ)
             (subactivity_occurrence ?s2 ?occ))))
```
:IC hard "**There exists an occurrence of an activity ?a for every branch of an activity tree for ?a. All atomic subactivity occurrences on the branch are subactivity occurrences of the occurrence of ?a.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s)
         (root ?s ?a)
         (not (atomic ?a)))
    (exists (?occ)
        (and (Activity_Occurrence ?occ)
             (occurrence_of ?occ ?a)
             (subactivity_occurrence ?s ?occ))))
```
:IC hard "**There exists an occurrence of an activity ?a for every branch of an activity tree for ?a. All root subactivity occurrences on the branch are subactivity occurrences of the occurrence of ?a.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?occ)
         (occurrence_of ?occ ?a)
         (not (atomic ?a)))
    (exists (?s)
        (and (Activity_Occurrence ?s)
             (root ?s ?a)
             (subactivity_occurrence ?s ?occ))))
```
:IC hard "**Every occurrence of a complex activity ?a contains an atomic subactivity occurrence that is the root of an activity tree for ?a.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?occ1)
         (Activity_Occurrence ?occ2)
         (occurrence_of ?occ1 ?a)
         (occurrence_of ?occ2 ?a)
         (/= ?occ1 ?occ2)
         (not (atomic ?a)))
    (exists (?s)
        (and (Activity_Occurrence ?s)
             (arboreal ?s)
             (subactivity_occurrence ?s ?occ1)
             (not (subactivity_occurrence ?s ?occ2)))))
```
:IC hard "**Distinct occurrences of an activity correspond to distinct branches of an activity tree.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?occ)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (occurrence_of ?occ ?a)
         (arboreal ?s1)
         (arboreal ?s2)
         (subactivity_occurrence ?s1 ?occ)
         (subactivity_occurrence ?s2 ?occ))
    (or (min_precedes ?s1 ?s2 ?a)
        (min_precedes ?s2 ?s1 ?a)
        (= ?s1 ?s2)))
```
:IC weak "**All atomic subactivity occurrences of a complex activity occurrence are elements of the same branch of the activity tree.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (Activity_Occurrence ?occ)
         (min_precedes ?s1 ?s2 ?a)
         (occurrence_of ?occ ?a)
         (subactivity_occurrence ?s2 ?occ))
    (subactivity_occurrence ?s1 ?occ))
```
:IC hard "**All elements of the same branch of an activity tree are atomic subactivity occurrences of the same activity occurrences.**"

```
(=> (and (Activity ?a1)
         (Activity ?a2)
         (Activity_Occurrence ?occ1)
         (Activity_Occurrence ?occ2)
         (occurrence_of ?occ1 ?a1)
         (occurrence_of ?occ2 ?a2)
         (not (atomic ?a1))
         (subactivity_occurrence ?occ1 ?occ2))
    (subactivity ?a1 ?a2))
```
:IC hard "The **subactivity_occurrence relation preserves the subactivity relation.**"

```
(=> (and (Activity ?a1)
         (Activity ?a2)
         (Activity_Occurrence ?occ1)
         (Activity_Occurrence ?occ2)
         (occurrence_of ?occ1 ?a1)
         (occurrence_of ?occ2 ?a2)
         (subactivity ?a1 ?a2)
         (/= ?a1 ?a2)
         (not (subactivity_occurrence ?occ1 ?occ2)))
    (exists (?s)
        (and (Activity_Occurrence ?s)
             (subactivity_occurrence ?s ?occ2)
             (not (subactivity_occurrence ?s ?occ1)))))
```
:IC hard "**Occurrences of subactivities are subactivity occurrences if the occurrences satisfy branch containment.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (mono ?s1 ?s2 ?a))
    (hom ?s1 ?s2 ?a))
```
:IC hard "**The mono relation is a branch homomorphism.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (hom ?s1 ?s2 ?a)
         (not (mono ?s1 ?s2 ?a)))
    (exists (?s3)
        (and (Activity_Occurrence ?s3)
             (or (and (min_precedes ?s3 ?s2 ?a)
                      (mono ?s1 ?s3 ?a))
                 (and (min_precedes ?s3 ?s1 ?a)
                      (mono ?s2 ?s3 ?a))))))
```
:IC hard "**If an atomic subactivity occurrence is mapped in a branch homomorphism, then there exists another atomic subactivity occurrence that is mono with it.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (Activity_Occurrence ?s3)
         (mono ?s1 ?s2 ?a)
         (mono ?s3 ?s2 ?a))
    (not (or (min_precedes ?s1 ?s3 ?a)
             (min_precedes ?s3 ?s1 ?a))))
```
:IC hard "**The mono relation is restricted to one-to-one homomorphisms between different branches of the activity tree.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (mono ?s1 ?s2 ?a))
     (mono ?s2 ?s1 ?a))
```
:IC soft "**The mono relation is symmetric on activity occurrences.**"

```
(=> (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (Activity_Occurrence ?s3)
         (mono ?s1 ?s2 ?a)
         (mono ?s2 ?s3 ?a))
    (mono ?s1 ?s3 ?a))
```
:IC soft "**The mono relation is transitive on activity occurrences.**"

Definitions

```
(<= (iso_occ ?s1 ?s2 ?a)
    (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (exists (?a1 ?a2 ?a3)
             (and (Activity ?a1)
                  (Activity ?a2)
                  (Activity ?a3)
                  (atomic ?a1)
                  (atomic ?a2)
                  (atomic ?a3)
                  (subactivity ?a3 ?a)
                  (occurrence_of ?s1 (conc ?a1 ?a3))
                  (occurrence_of ?s2 (conc ?a2 ?a3))))))
```

:rem "**Two activity occurrences are occurrence isomorphic if and only if they are occurrences of atomic activities that have a common subactivity with the complex activity ?a.**"

```
(<= (iso_occ ?s1 ?s2 ?a)
    (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (exists (?a1 ?a2 ?a3)
            (and (Activity ?a1)
                 (Activity ?a2)
                 (Activity ?a3)
                 (atomic ?a1)
                 (atomic ?a2)
                 (atomic ?a3)
                 (subactivity ?a3 ?a)
                 (occurrence_of ?s1 (conc ?a1 ?a3))
                 (occurrence_of ?s2 (conc ?a2 ?a3))))
                 (Activity ?a4)
                 (subactivity ?a4 (conc ?a3 ?a1))
                 (subactivity ?a4 (conc ?a3 ?a2))
                 (subactivity ?a4 ?a)
                 (not (subactivity ?a3 ?a4))))
```

:rem "**Two activity occurrences are occurrence isomorphic if and only if they are occurrences of atomic activities that have a common subactivity with the complex activity ?a.**"

```
(<= (hom ?s1 ?s2 ?a)
    (and (Activity ?a)
         (Activity_Occurrence ?s1)
         (Activity_Occurrence ?s2)
         (exists (?occ1 ?occ2)
            (and (Activity_Occurrence ?occ1)
                 (Activity_Occurrence ?occ2)
                 (iso_occ ?s1 ?s2 ?a)
                 (not (min_precedes ?s1 ?s2 ?a))
                 (not (min_precedes ?s2 ?s1 ?a))
                 (subactivity_occurrence ?s1 ?occ1)
                 (subactivity_occurrence ?s2 ?occ2)
                 (occurrence_of ?occ1 ?a)
                 (occurrence_of ?occ2 ?a)))))
```

:rem "**For every two occurrences of the same activity on different branches of an activity tree, there exist homomorphic occurrences on those branches.**"

```
(<= (root_occ ?s ?occ)
    (and (Activity_Occurrence ?s)
         (Activity_Occurrence ?occ)
         (exists (?a)
            (and (Activity ?a)
                 (occurrence_of ?occ ?a)
                 (subactivity_occurrence ?s ?occ)
                 (root ?s ?a)))))
```

:rem "**An occurrence ?occ is the root occurrence of an occurrence of ?a if and only if it is a subactivity occurrence and it is the root of an activity tree for ?a.**"

```
(<= (leaf_occ ?s ?occ)
    (and (Activity_Occurrence ?s)
         (Activity_Occurrence ?occ)
         (exists (?a)
            (and (Activity ?a)
                 (occurrence_of ?occ ?a)
```

```
                (subactivity_occurrence ?s ?occ)
                (leaf ?s ?a)))))
```
:rem "**An occurrence ?occ is the leaf occurrence of an occurrence of ?a if and only if it is a subactivity occurrence and it is the leaf of an activity tree for ?a.**"

```
 (<= (same_grove ?occ1 ?occ2)
     (and (Activity_Occurrence ?occ1)
          (Activity_Occurrence ?occ1)
          (Activity_Occurrence ?s1)
          (Activity_Occurrence ?s2)
          (exists (?a)
             (and (Activity ?a)
                  (occurrence_of ?occ1 ?a)
                  (occurrence_of ?occ2 ?a)
                  (root_occ ?s1 ?occ1)
                  (root_occ ?s2 ?occ2)
                  (or (and (initial ?s1)
                           (initial ?s2))
                           (exists (?s4 ?a1 ?a2)
                              (and (= ?s1 (successor ?a1 ?s4))
                                   (= ?s2 (successor ?a2 ?s4)))))))))
```
:rem "**Two complex activity occurrences are in the same grove if and only if they are occurrences of the same activity and their root occurrences are siblings.**"

C.2 Entity Information Semantics

C.2.1 Core Entities and Core Properties

Classes

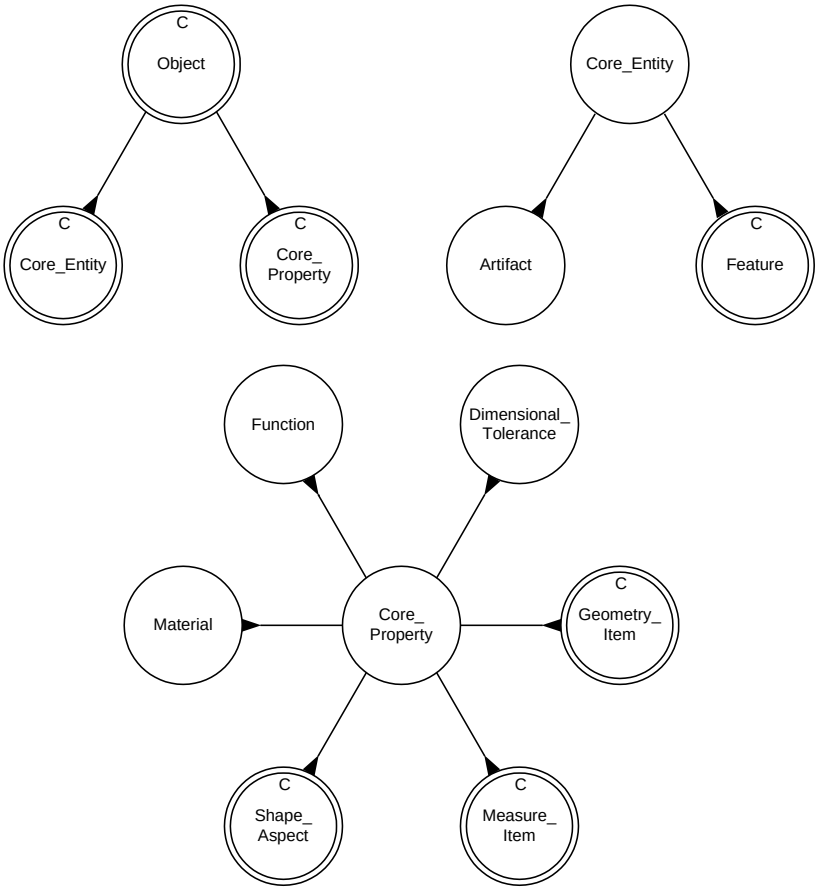


Figure C-14 Core_Entity and Core_Property Classes

:Prop **Core_Entity**
:Inst Property
:sup Object
:name "Core Entity"
:rem "(Core_Entity ?coreEnt) is TRUE in an interpretation of the Foundation Layer if and only if Core_Entity is an abstract kind of object from which the Artifact and Feature classes are specialised."

:Prop **Core_Property**
:Inst Property
:sup Object
:name "Core Property"
:rem "(Core_Property ?coreProp) is TRUE in an interpretation of the the Foundation Layer if and only if Core_Property is an abstract kind of object from which a number of subclasses are specialised."

:Prop **Material**
:Inst Property
:sup Core_Property

:name "Material"
:rem "(Material ?m) is TRUE in an interpretation of the Foundation Layer if and only if ?m is a member of a set of materials. Materials describe the internal composition of features with positive geometry and the internal composition of artifacts."

:Prop **Function**
:Inst Property
:sup Core_Property
:name "Function"
:rem "(Function ?func) is TRUE in an interpretation of the Foundation Layer if and only if ?func is a member of a set of functions. Functions are intended behaviours that represent aspects of what features and artifacts are supposed to do."

Relations

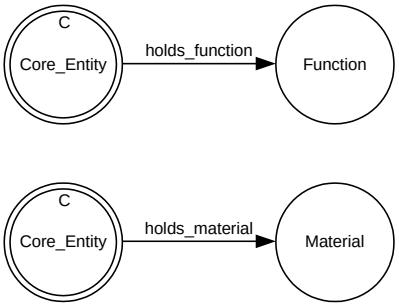


Figure C-15 Core_Entity and Core_Property Relations

:Rel **holds_material**
:Inst BinaryRel
:Sig Core_Entity Material
:name "holds_material"
:rem "(holds_material ?coreEnt ?m) is TRUE in an interpretation of the Foundation Layer if and only if the core entity ?coreEnt is composed of the material ?m."

:Rel **holds_function**
:Inst BinaryRel
:Sig Core_Entity Function
:name "holds_function"
:rem "(holds_function ?coreEnt ?func) is TRUE in an interpretation of the Foundation Layer if and only if the core entity ?coreEnt has an intended function ?func."

Axioms

(=> (Core_Entity ?coreEnt)
    (Object ?coreEnt))
:IC hard "**Core entities are objects.**"

(=> (Core_Property ?coreProp)
    (Object ?coreProp))
:IC hard "**Core properties are objects.**"

(=> (Material ?m)
    (Core_Property ?m))
:IC hard "**Materials are core properties.**"

```
 (=> (Function ?func)
      (Core_Property ?func))
:IC hard "Functions are core properties."

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

C.2.2 Geometry and Measure Items

Classes

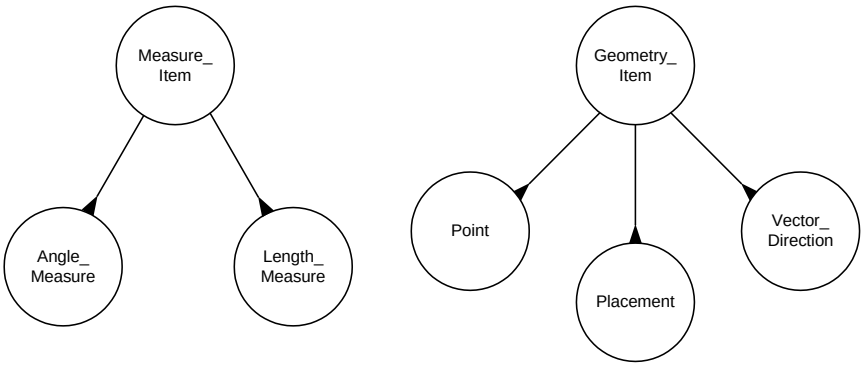


Figure C-16 Geometry_Item and Measure_Item Classes

:Prop **Geometry_Item**
:Inst Property
:sup Core_Property
:name "Geometry Item"
:rem "(Geometry_Item ?geo) is TRUE in an interpretation of the Foundation Layer if and only if Geometry_Item is an abstract kind of Core_Property where an instance of Geometry_Item ?geo can only exist as an instance of one of the instantiable subclasses of Geometry_Item."

:Prop **Point**
:Inst Property
:sup Geometry_Item
:name "Point"
:rem "(Point ?pt) is TRUE in an interpretation of the Foundation Layer if and only if ?pt is a member of a set of points described in terms of a position in space relative to the X, Y and Z Cartesian axes."

:Prop **Vector_Direction**
:Inst Property
:sup Geometry_Item
:name "Vector Direction"
:rem "(Vector_Direction ?v) is TRUE in an interpretation of the Foundation Layer if and only if ?v is a member of a set of vector directions stated in terms of a position in space relative to the X, Y and Z Cartesian axes. Vector directions are unitless."

:Prop **Placement**
:Inst Property
:sup Geometry_Item
:name "Placement"

:rem "(Placement ?p) is TRUE in an interpretation of the Foundation Layer if and only if ?p is a member of a set of placements. A placement is the direction and location of the basic shape of a part, feature on a part or of the components of a feature which are profile objects and path objects."

:Prop **Measure_Item**
:Inst Property
:sup Core_Property
:name "Measure Item"
:rem "(Measure_Item ?mea) is TRUE in an interpretation of the Foundation Layer if and only if Measure_Item is an abstract kind of Core_Property where an instance of Measure_Item ?mea can only exist as an instance of one of the instantiable subclasses of Measure_Item."

:Prop **Length_Measure**
:Inst Property
:sup Measure_Item
:name "Length Measure"
:rem "(Length_Measure ?length) is TRUE in an interpretation of the Foundation Layer if and only if ?length is a member of a set of length measures."

:Prop **Angle_Measure**
:Inst Property
:sup Measure_Item
:name "Angle Measure"
:rem "(Angle_Measure ?angle) is TRUE in an interpretation of the Foundation Layer if and only if ?angle is a member of a set of angle measures."

Relations

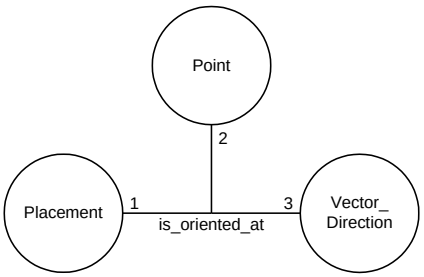


Figure C-17 Geometry_Item and Measure_Item Relations

:Rel **is_oriented_at**
:Inst TernaryRel
:Sig Placement Point Vector_Direction
:name "is_oriented_at"
:rem "(is_oriented_at ?p ?pt ?v) is TRUE in an interpretation of the Foundation Layer if and only if the placement ?p is specified relative to a point ?pt in space which is the origin of the vector direction ?v."

Functions

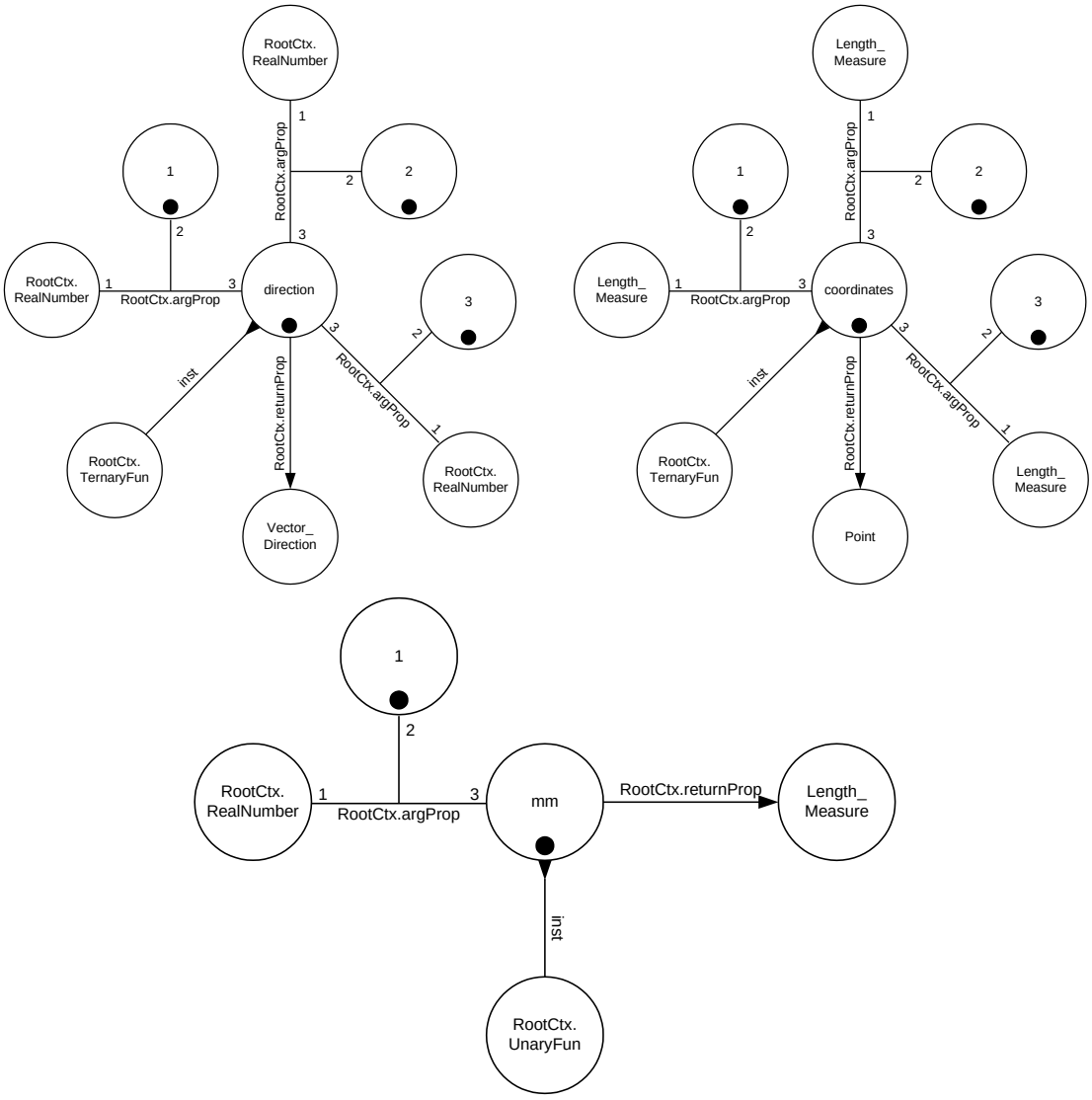
:Fun **coordinates**
:Inst TernaryFun
:Sig Length_Measure Length_Measure Length_Measure -> Point
:name "coordinates"
:rem "(= ?pt (coordinates ?length1 ?length2 ?length3)) is TRUE in an interpretation of the Foundation Layer if and only if ?pt is the point whose coordinates are given by length

measures ?length1, ?length2 and ?length3 relative to the X, Y and Z Cartesian axes respectively."

:Fun **direction**
:Inst TernaryFun
:Sig RealNumber RealNumber RealNumber -> Vector_Direction
:name "direction"
:rem "(= ?v (direction ?real1 ?real2 ?real3)) is TRUE in an interpretation of the Foundation Layer if and only if ?v is the vector direction whose direction is given by real numbers ?real1, ?real2 and ?real3 relative to the X, Y and Z Cartesian axes respectively."

:Fun **mm**
:Inst UnaryFun
:Sig RealNumber -> Length_Measure
:name "millimetre"
:rem "(= ?length (mm ?real)) is TRUE in an interpretation of the Foundation Layer if and only if ?length is a length measure whose value in millimeters is given by a real number ?real."

:Fun **degree**
:Inst UnaryFun
:Sig RealNumber -> Angle_Measure
:name "degree"
:rem "(= ?angle (degree ?real)) is TRUE in an interpretation of the Foundation Layer if and only if ?angle is angle measure whose value in degrees is given by a real number ?real."



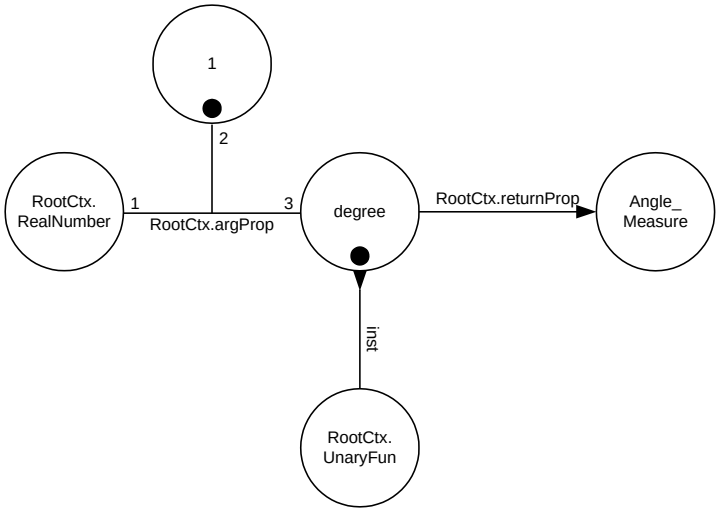


Figure C-18 Geometry_Item and Measure_Item Functions

Axioms

(=> (Geometry_Item ?geo)
(Core_Property ?geo))
:IC hard "**Geometry items are core properties.**"

(=> (Geometry_Item ?geo)
(exists (?class)
(and (RootCtx.sup ?class Geometry_Item)
(RootCtx.inst ?geo ?class)
(/= ?class Geometry_Item))))
:IC hard "**Any instance of geometry item can only be an instance of one of its subclasses.**"

(and (=> (Point ?geo)
(not (or (Vector_Direction ?geo) (Placement ?geo))))
(=> (Vector_Direction ?geo)
(not (Placement ?geo))))
:IC hard "**Points, vector directions and placements are all distinct kinds of things.**"

(=> (is_oriented_at ?p ?pt ?v)
(and (Placement ?p)
(Point ?pt)
(Vector_Direction ?v)))
:IC hard "**The orientation relation only holds between placements, points and vector directions.**"

(=> (Measure_Item ?mea)
(Core_Property ?mea))
:IC hard "**Measure items are core properties.**"

(=> (Measure_Item ?mea)
(exists (?class)
(and (RootCtx.sup ?class Measure_Item)
(RootCtx.inst ?mea ?class)
(/= ?class Measure_Item))))
:IC hard "**Any instance of measure item can only be an instance of one of its subclasses.**"

```
(=> (Length_Measure ?mea)
    (not (Angle_Measure ?mea)))
:IC hard "Length measures and angle measures are all distinct kinds of things."

(=> (Length_Measure ?length)
    (exists (?real)
        (and (RootCtx.RealNumber ?real)
            (= ?length (Foundation.mm ?real)))))
:IC hard "Every length measure is given by some unit of measurement and some real value."

(=> (Angle_Measure ?angle)
    (exists (?real)
        (and (RootCtx.RealNumber ?real)
            (= ?angle (Foundation.degree ?real)))))
:IC hard "Every angle measure is given by some unit of measurement and some real value."

(=> (Point ?pt)
    (exists (?length1 ?length2 ?length3)
        (and (Length_Measure ?length1)
            (Length_Measure ?length2)
            (Length_Measure ?length3)
            (= ?pt (coordinates ?length1 ?length2 ?length3)))))
:IC hard "Every point is given by some x, y and z coordinates."

(=> (Vector_Direction ?v)
    (exists (?real1 ?real2 ?real3)
        (and (RootCtx.RealNumber ?real1)
            (RootCtx.RealNumber ?real2)
            (RootCtx.RealNumber ?real3)
            (= ?v (direction ?real1 ?real2 ?real3)))))
:IC hard "Every vector direction is given by some x, y and z direction ratio."

(=> (and (is_oriented_at ?p ?pt1 ?v1)
         (is_oriented_at ?p ?pt2 ?v2))
    (and (= ?pt1 ?pt2)
         (= ?v1 ?v2)))
:IC hard "A placement is associated with a unique point and a unique vector direction."

(=> (Placement ?p)
    (exists (?pt ?v)
        (and (Point ?pt)
            (Vector_Direction ?v)
            (is_oriented_at ?p ?pt ?v))))
:IC hard "Every placement is oriented at some point and vector direction."
```

C.2.3 Shape Aspects

Classes

```
:Prop Shape_Aspect
:Inst Property
:sup Core_Property
:name "Shape Aspect"
```

:rem "(Shape_Aspect ?sa) is TRUE in an interpretation of the Foundation Layer if and only if Shape_Aspect is an abstract kind of Core_Property where an instance of Shape_Aspect ?sa can only exist as an instance of one of the instantiable subclasses of Shape_Aspect."

:Prop **Circular_Closed_Profile**
:Inst Property
:sup Shape_Aspect
:name "Circular Closed Profile"
:rem "(Circular_Closed_Profile ?ccp) is TRUE in an interpretation of the Foundation Layer if and only if ?ccp is a member of a set of circular closed profiles. A circular closed profile is an enclosed 2D area which is defined according to its diameter. The orientation is at the centre of the circular arc."

:Prop **Rectangular_Closed_Profile**
:Inst Property
:sup Shape_Aspect
:name "Rectangular Closed Profile"
:rem "(Rectangular_Closed_Profile ?rcp) is TRUE in an interpretation of the Foundation Layer if and only if ?rcp is a member of a set of rectangular closed profiles. A rectangular closed profile is an enclosed area bounded by four sides with opposite sides equal in length and corners at 90 degrees. The orientation is at the centre of the rectangle."

:Prop **Linear_Path**
:Inst Property
:sup Shape_Aspect
:name "Linear Path"
:rem "(Linear_Path ?lin) is TRUE in an interpretation of the Foundation Layer if and only if ?lin is a member of a set of linear paths. A linear path defines a direction of travel along a line and is defined according to the length of the path."

:Prop **Linear_Profile**
:Inst Property
:sup Shape_Aspect
:name "Linear Profile"
:rem "(Linear_Profile ?lp) is TRUE in an interpretation of the Foundation Layer if and only if ?lp is a member of a set of linear profiles. A linear profile can be regarded as being an open profile that involves exactly two connected points in a straight line with a specified length."

:Prop **Taper**
:Inst Property
:sup Shape_Aspect
:name "Taper"
:rem "(Taper ?tap) is TRUE in an interpretation of the Foundation Layer if and only if ?tap is a member of a set of tapers. A taper is a type of shape aspect which represents a constant change in shape of a feature or a part. A taper starts at the location of placement of a feature and is applied to the entire feature."

:Prop **Transition_Feature**
:Inst Property
:sup Shape_Aspect
:name "Transition Feature"
:rem "(Transition_Feature ?tf) is TRUE in an interpretation of the Foundation Layer if and only if Transition_Feature is an abstract kind of Shape_Aspect where an instance of Transition_Feature ?tf can only exist as an instance of one of the instantiable subclasses of Transition_Feature."

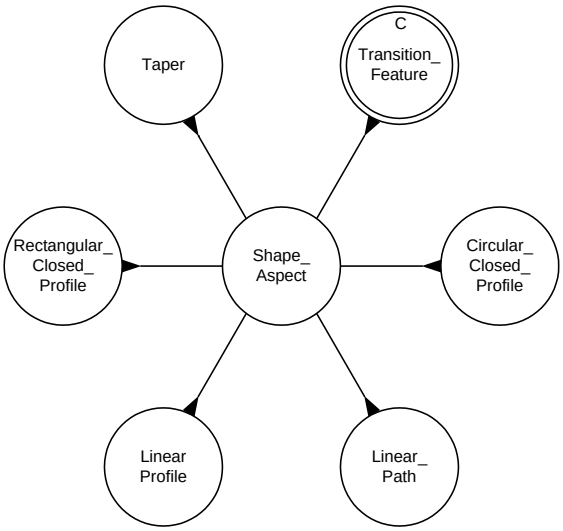


Figure C-19 Shape_Aspect Classes

Relations

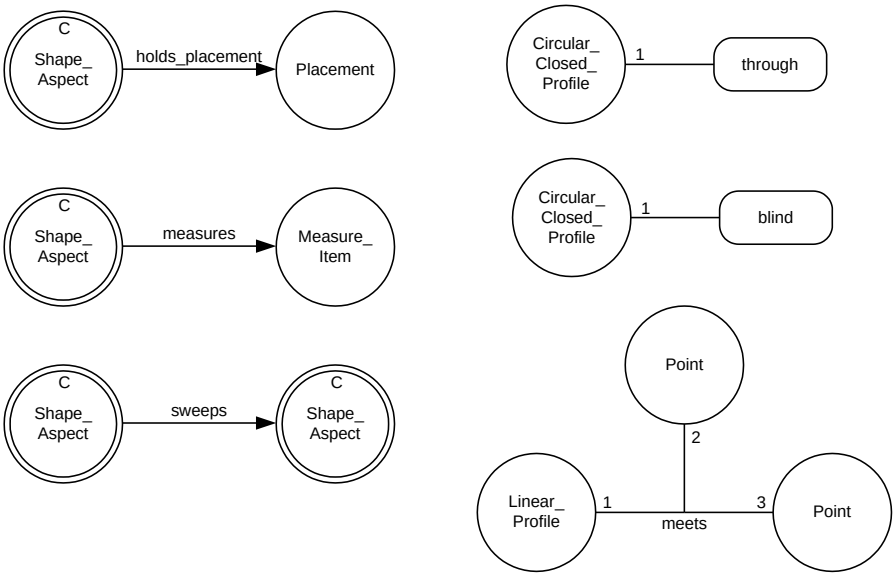


Figure C-20 Shape_Aspect Relations

:Rel **holds_placement**
:Inst BinaryRel
:Sig Shape_Aspect Placement
:name "holds_placement"
:rem "(holds_placement ?sa ?p) is TRUE in an interpretation of the Foundation Layer if and only if the shape aspect ?sa holds a placement ?p. A shape aspect may have one and only one placement."

:Rel **measures**
:Inst BinaryRel
:Sig Shape_Aspect Measure_Item
:name "measures"
:rem "(measures ?sa ?measure) is TRUE in an interpretation of the Foundation Layer if and only if the shape aspect ?sa has ?measure as its measure representation item."

:Rel **sweeps**
:Inst BinaryRel
:Sig Shape_Aspect Shape_Aspect
:name "sweeps"
:rem "(sweeps ?sa1 ?sa2) is TRUE in an interpretation of the Foundation Layer if and only if the shape aspect ?sa1 is a linear path or taper that sweeps another existing shape aspect ?sa2 to produce a 3D feature."

:Rel **meets**
:Inst TernaryRel
:Sig Linear_Profile Point Point
:name "meets"
:rem "(meets ?lp ?pt1 ?pt2) is TRUE in an interpretation of the Foundation Layer if and only if the linear profile ?lp meets the points ?pt1 and ?pt2 forming a straight line."

:Rel **blind**
:Inst UnaryRel
:Sig Circular_Closed_Profile
:name "blind"
:rem "(blind ?ccp) is TRUE in an interpretation of the Foundation Layer if and only if the circular closed profile ?ccp describes a blind hole bottom condition of a round hole feature. A blind hole bottom condition is one where the hole feature does not go through the material completely."

:Rel **through**
:Inst UnaryRel
:Sig Circular_Closed_Profile
:name "through"
:rem "(through ?ccp) is TRUE in an interpretation of the Foundation Layer if and only if the circular closed profile ?ccp describes a through hole bottom condition of a round hole feature. A through hole bottom condition is one where the hole feature goes through the material completely."

Axioms

```
(=> (Shape_Aspect ?sa)
    (Core_Property ?sa))
```
:IC hard "**Shape aspects are core properties.**"

```
(=> (Shape_Aspect ?sa)
    (exists (?class)
        (and (RootCtx.sup ?class Shape_Aspect)
             (RootCtx.inst ?sa ?class)
             (/= ?class Shape_Aspect))))
```
:IC hard "**Any instance of shape aspect can only be an instance of one of its subclasses.**"

```
(and (=> (Circular_Closed_Profile ?sa)
         (not (or (Rectangular_Closed_Profile ?sa) (Linear_Path ?sa) (Linear_Profile ?sa) (Taper ?sa) (Transition_Feature ?sa))))
     (=> (Rectangular_Closed_Profile ?sa)
         (not (or (Linear_Path ?sa) (Linear_Profile ?sa) (Taper ?sa) (Transition_Feature ?sa))))
     (=> (Linear_Path ?sa)
         (not (or (Linear_Profile ?sa) (Taper ?sa) (Transition_Feature ?sa))))
     (=> (Linear_Profile ?sa)
         (not (or (Taper ?sa) (Transition_Feature ?sa))))
     (=> (Taper ?sa)
```

```
        (not (Transition_Feature ?sa))))
```
:IC hard "**Circular closed profiles, rectangular closed profiles, linear paths, linear profiles, tapers and transition features are all distinct kinds of things.**"

```
(=> (holds_placement ?sa ?p)
    (and (Shape_Aspect ?sa)
         (Placement ?p)))
```
:IC hard "**The relation holds_placement only holds between shape aspects and placements.**"

```
(=> (and (holds_placement ?sa ?p1)
         (holds_placement ?sa ?p2))
    (= ?p1 ?p2))
```
:IC hard "**A shape aspect is associated with a unique placement.**"

```
(=> (meets ?lp ?pt1 ?pt2)
    (and (Linear_Profile ?lp)
         (Point ?pt1)
         (Point ?pt2)
         (/= ?pt1 ?pt2)))
```
:IC hard "**The relation meets only holds between linear profiles and two distinct points.**"

```
(=> (meets ?lp ?pt1 ?pt2)
    (meets ?lp ?pt2 ?pt1))
```
:IC soft "**The relation meets is symmetric over linear profiles and points.**"

```
(=> (and (Linear_Profile ?lp)
         (Point ?pt))
    (not (meets ?lp ?pt ?pt)))
```
:IC hard "**The relation meets is irreflexive on points.**"

```
(=> (measures ?sa ?mea)
    (and (Shape_Aspect ?sa)
         (Measure_Item ?mea)))
```
:IC hard "**The relation measures only holds between shape aspects and measure items.**"

```
(=> (sweeps ?sa1 ?sa2)
    (and (or (Linear_Path ?sa1)
             (Taper ?sa1))
         (Shape_Aspect ?sa2)
          (not (or (Linear_Path ?sa2)
                   (Taper ?sa2)))))
```
:IC hard "**The relation sweeps holds over shape aspects that are linear paths or tapers and other shape aspects.**"

```
(=> (blind ?ccp)
    (Circular_Closed_Profile ?ccp))
```
:IC hard "**The relation blind only holds for circular closed profiles.**"

```
(=> (through ?ccp)
    (Circular_Closed_Profile ?ccp))
```
:IC hard "**The relation through only holds for circular closed profiles.**"

```
(=> (Circular_Closed_Profile ?ccp)
    (exists (?length)
        (and (Length_Measure ?length)
             (measures ?ccp ?length))))
```
:IC hard "**Every circular closed profile has an associated length measure which represents the diameter of the profile.**"

```
(=> (Circular_Closed_Profile ?ccp)
    (exists (?p)
        (and (Placement ?p)
            (holds_placement ?ccp ?p))))
```
:IC hard "**Every circular closed profile has an associated placement.**"

```
(=> (Rectangular_Closed_Profile ?rcp)
    (exists (?length1 ?length2)
        (and (Length_Measure ?length1)
            (Length_Measure ?length2)
            (measures ?rcp ?length1)
            (measures ?rcp ?length2))))
```
:IC hard "**Every rectangular closed profile has two associated length measures which represent the width and breadth of the profile.**"

```
(=> (Rectangular_Closed_Profile ?rcp)
    (exists (?p)
        (and (Placement ?p)
            (holds_placement ?rcp ?p))))
```
:IC hard "**Every rectangular closed profile has an associated placement.**"

```
(=> (Linear_Path ?lin)
    (exists (?length)
        (and (Length_Measure ?length)
            (measures ?lin ?length))))
```
:IC hard "**Every linear path has an associated length measure which represents the distance of the linear path.**"

```
(=> (Linear_Path ?lin)
    (exists (?p)
        (and (Placement ?p)
            (holds_placement ?lin ?p))))
```
:IC hard "**Every linear path has an associated placement.**"

```
(=> (Linear_Path ?lin)
    (exists (?sa)
        (and (Shape_Aspect ?sa)
            (sweeps ?lin ?sa))))
```
:IC hard "**Every linear path has an associated shape aspect that the linear path sweeps.**"

```
(=> (Linear_Profile ?lp)
    (exists (?length)
        (and (Length_Measure ?length)
            (measures ?lp ?length))))
```
:IC hard "**Every linear profile has an associated length measure which represents the length of the profile.**"

```
(=> (Linear_Profile ?lp)
    (exists (?pt1 ?pt2)
        (and (Point ?pt1)
            (Point ?pt2)
            (meets ?lp ?pt1 ?pt2))))
```
:IC hard "**Every linear profile meets two distinct points.**"

```
(=> (Taper ?tap)
    (exists (?angle)
        (and (Angle_Measure ?angle)
            (measures ?tap ?angle))))
```
:IC hard "**Every taper has an associated angle measure which represents the taper angle.**"

```
 (=> (Taper ?tap)
     (exists (?p)
          (and (Placement ?p)
               (holds_placement ?tap ?p))))
:IC hard "Every taper has an associated placement."

 (=> (Taper ?tap)
     (exists (?sa)
          (and (Shape_Aspect ?sa)
               (sweeps ?tap ?sa))))
:IC hard "Every taper has an associated shape aspect that the taper sweeps."
```

C.2.4 Features and Artifacts

Classes

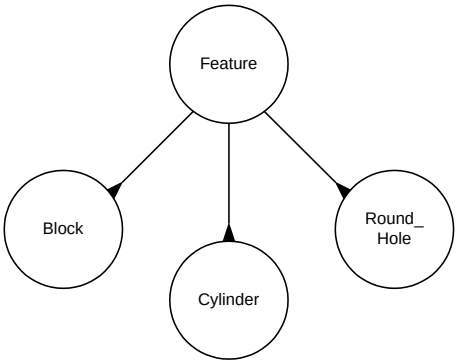


Figure C-21 Feature Classes

:Prop **Artifact**
:Inst Property
:sup Core_Entity
:name "Artifact"
:rem "(Artifact ?art) is TRUE in an interpretation of the Foundation Layer if and only if ?art is a member of a set of artifacts in the universe of discourse of the interpretation. Intuitively, artifacts represent a distinct entity in a product whether that entity is a component, part, subassembly or assembly. Artifacts can involve intuitions about part families."

:Prop **Feature**
:Inst Property
:sup Core_Entity
:name "Feature"
:rem "(Feature ?f) is TRUE in an interpretation of the Foundation Layer if and only if ?f is a member of a set of features. A feature represents a portion or element of interest of an artifact's form."

:Prop **Round_Hole**
:Inst Property
:sup Feature
:name "Round Hole"
:rem "(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."

:Prop **Block**
:Inst Property
:sup Feature
:name "Block"
:rem "(Block ?b) is TRUE in an interpretation of the Foundation Layer if and only if ?b is a member of a set of blocks. A block specifies the representation of a feature that is a rectangular volume defined as a rectangular closed profile swept along a linear path. The orientation of a block is at the centre point of the rectangular closed profile with the direction pointing out of the block along the axis of the rectangular closed profile."

:Prop **Cylinder**
:Inst Property
:sup Feature
:name "Cylinder"
:rem "(Cylinder ?c) is TRUE in an interpretation of the Foundation Layer if and only if ?c is a member of a set of cylinders. A cylinder specifies the representation of a feature that is a cylindrical volume defined as a circular closed profile swept along a linear path. The orientation of the cylinder is at the centre point of the circular closed profile with the direction pointing out of the cylinder along the axis of the circular closed profile."

Relations

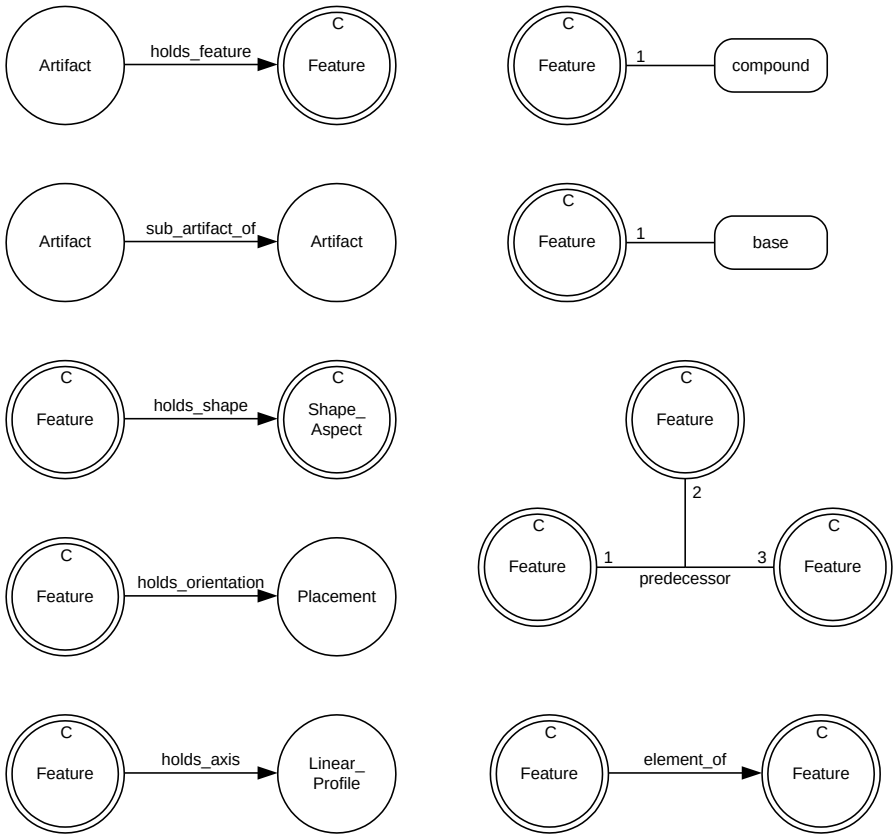


Figure C-22 Feature and Artifact Relations

:Rel **holds_shape**
:Inst BinaryRel
:Sig Feature Shape_Aspect
:name "holds_shape"

:rem "(holds_shape ?f ?sa) is TRUE in an interpretation of the Foundation Layer if and only if the feature ?f has a related shape aspect ?sa that is used towards the definition of the feature ?f."

:Rel **compound**
:Inst UnaryRel
:Sig Feature
:name "compound"
:rem "(compound ?f) is TRUE in an interpretation of the Foundation Layer if and only if the feature ?f is a compound feature that is the union of more than one feature to create a more complex feature definition."

:Rel **element_of**
:Inst BinaryRel
:Inst ReflexiveBR
:Sig Feature Feature
:name "element_of"
:rem "(element_of ?f1 ?f) is TRUE in an interpretation of the Foundation Layer if and only if the feature ?f1 is an element of a compound feature ?f."

:Rel **base**
:Inst UnaryRel
:Sig Feature
:name "base"
:rem "(base ?f) is TRUE in an interpretation of the Foundation Layer if and only if the feature ?f is an element of a compound feature such that ?f is the base feature from which other element features are aggregated."

:Rel **predecessor**
:Inst TernaryRel
:Sig Feature Feature Feature
:name "predecessor"
:rem "(predecessor_of ?f1 ?f2 ?f) is TRUE in an interpretation of the Foundation Layer if and only if the feature ?f1 is an element of a compound feature ?f, the latter having the highest precedence over ?f2, a second element of the compound feature ?f."

:Rel **holds_feature**
:Inst BinaryRel
:Sig Artifact Feature
:name "holds_feature"
:rem "(holds_feature ?art ?f) is TRUE in an interpretation of the Foundation Layer if and only if the artifact ?art holds a given feature ?f. This is the basic relation between artifacts and features."

:Rel **sub_artifact_of**
:Inst BinaryRel
:Inst IrreflexiveBR
:Inst TransitiveBR
:Sig Artifact Artifact
:name "sub_artifact_of"
:rem "(sub_artifact_of ?sub ?art) is TRUE in an interpretation of the Foundation Layer if and only if the artifact ?sub is a sub-artifact of the artifact ?art."

:Rel **holds_orientation**
:Inst BinaryRel
:Sig Feature Placement
:name "holds_orientation"
:rem "(holds_orientation ?f ?p) is TRUE in an interpretation of the Foundation Layer if and only if the feature ?f holds an orientation given by a placement ?p. The orientation of a feature corresponds to the placement of one of the shape aspects that make up the feature."

:Rel **holds_axis**
:Inst BinaryRel
:Sig Feature Linear_Profile
:name "holds_axis"
:rem "(holds_axis ?f ?lp) is TRUE in an interpretation of the Foundation Layer if and only if the feature ?f holds an axis given by the linear profile ?lp."

Axioms

```
 (=> (and (Feature ?f)
          (Artifact ?art))
     (and (Core_Entity ?f)
          (Core_Entity ?art)))
:IC hard "Features and artifacts are core entities."

 (and (=> (Round_Hole ?f)
          (not (or (Block ?f) (Cylinder ?f))))
      (=> (Block ?f)
          (not (Cylinder ?f))))
:IC hard "Round holes, blocks and cylinders are all distinct kinds of things."

 (=> (holds_shape ?f ?sa)
     (and (Feature ?f)
          (Shape_Aspect ?sa)))
:IC hard "The relation holds_shape only holds between features and shape aspects."

 (=> (compound ?f)
     (Feature ?f))
:IC hard "The relation compound only holds for features."

 (=> (element_of ?f1 ?f)
     (and (Feature ?f1)
          (Feature ?f)))
:IC hard "The element_of relation only holds between features."

 (=> (base ?f)
     (Feature ?f))
:IC hard "The relation base only holds for features."

 (=> (predecessor ?f1 ?f2 ?f)
      (and (Feature ?f1)
           (Feature ?f2)
           (Feature ?f)
           (compound ?f)))
:IC hard "The relation predecessor only holds between features."

 (=> (and (Feature ?f1)
          (Feature ?f))
     (not (predecessor ?f1 ?f1 ?f)))
:IC hard "The relation predecessor is irreflexive."

 (=> (and (predecessor ?f1 ?f2 ?f)
          (predecessor ?f2 ?f3 ?f))
     (predecessor ?f1 ?f3 ?f))
:IC soft "The relation predecessor is transitive on compound features."
```

```
(=> (holds_feature ?art ?f)
    (and (Feature ?f)
         (Artifact ?art)))
:IC hard "The holds_feature relation only holds between artifacts and features."

(=> (sub_artifact_of ?sub ?art)
    (and (Artifact ?sub)
         (Artifact ?art)))
:IC hard "The sub_artifact_of relation only holds between artifacts."

(=> (holds_orientation ?f ?p)
    (and (Feature ?f)
         (Placement ?p)))
:IC hard "The relation holds_orientation only holds between features and placements."

(=> (and (Feature ?f)
         (Placement ?p1)
         (Placement ?p2)
         (holds_orientation ?f ?p1)
         (holds_orientation ?f ?p2))
    (= ?p1 ?p2))
:IC hard "A feature is associated with a unique placement."

(=> (holds_axis ?f ?lp)
    (and (Feature ?f)
         (Linear_Profile ?lp)))
:IC hard "The relation holds_axis on holds between features and linear profiles."

(=> (compound ?f)
    (exists (?f1 ?f2)
        (and (Feature ?f1)
             (Feature ?f2)
             (/= ?f1 ?f2)
             (element_of ?f1 ?f)
             (element_of ?f2 ?f))))
:IC hard "If a feature is compound, then there should exist any two features that are elements of the compound feature."

(=> (compound ?f)
    (exists (?f1)
        (and (Feature ?f1)
             (base ?f1)
             (element_of ?f1 ?f))))
:IC hard "If a feature is compound, then there exists a base feature that is an element of the compound feature."

(=> (and (compound ?f)
         (base ?f1)
         (element_of ?f1 ?f))
    (not (exists (?f2)
           (and (Feature ?f2)
                (predecessor ?f2 ?f1 ?f)))))
:IC hard "No feature can be a predecessor of a base feature in a compound feature."

(=> (and (compound ?f)
         (base ?f1)
         (element_of ?f1 ?f))
    (exists (?f2)
        (and (Feature ?f2)
             (element_of ?f2 ?f)
```

```
            (predecessor ?f1 ?f2 ?f))))
```
:IC hard "**Every feature stated to be a base feature implies the existence of another feature which the base feature precedes on a compound feature.**"

```
(=> (and (Artifact ?art)
         (Round_Hole ?hole)
         (holds_feature ?art ?hole))
    (exists (?f)
        (and (Feature ?f)
             (holds_feature ?art ?f)
             (not (Round_Hole ?f)))))
```
:IC hard "**An artifact can only hold a round hole provided it holds another feature that is not a round hole i.e. a round hole cannot be the sole feature describing an artifact.**"

```
(=> (Artifact ?art)
    (exists (?m)
        (and (Material ?m)
             (holds_material ?art ?m))))
```
:IC soft "**Every artifact holds some material that describes its internal composition.**"

```
(=> (and (Cylinder ?c)
         (Circular_Closed_Profile ?ccp)
         (Linear_Path ?lin)
         (holds_shape ?c ?ccp)
         (holds_shape ?c ?lin)
         (holds_placement ?ccp ?p1)
         (sweeps ?lin ?ccp))
    (exists (?p2)
        (and (holds_orientation ?c ?p2)
             (= ?p1 ?p2))))
```
:IC hard "**The orientation of a cylinder corresponds to the placement of its circular closed profile that its linear path sweeps.**"

```
(=> (and (Cylinder ?c)
         (Circular_Closed_Profile ?ccp1)
         (Linear_Path ?lin)
         (Length_Measure ?length1)
         (holds_shape ?c ?ccp1)
         (holds_shape ?c ?lin)
         (sweeps ?lin ?ccp1)
         (measures ?ccp1 ?length1))
    (exists (?ccp2 ?length2)
        (and (Circular_Closed_Profile ?ccp2)
             (Length_Measure ?length2)
             (holds_shape ?c ?ccp2)
             (measures ?ccp2 ?length2)
             (= ?length1 ?length2))))
```
:IC hard "**For any cylinder the diameter of its circular closed profile that its linear path sweeps is the same as the diameter of its other existing circular closed profile.**"

```
(=> (and (Block ?b)
         (Rectangular_Closed_Profile ?rcp)
         (Linear_Path ?lin)
         (holds_shape ?b ?rcp)
         (holds_shape ?b ?lin)
         (holds_placement ?rcp ?p1)
         (sweeps ?lin ?rcp))
    (exists (?p2)
        (and (holds_orientation ?b ?p2)
             (= ?p1 ?p2))))
```

:IC hard "**The orientation of a block corresponds to the placement of its rectangular closed profile that its linear path sweeps.**"

```
(=> (Cylinder ?c)
     (exists (?ccp1 ?ccp2)
          (and (Circular_Closed_Profile ?ccp1)
               (Circular_Closed_Profile ?ccp2)
               (/= ?ccp1 ?ccp2)
               (holds_shape ?c ?ccp1)
               (holds_shape ?c ?ccp2))))
```

:IC hard "**Every cylinder holds exactly two circular closed profiles.**"

```
(=> (Cylinder ?c)
     (exists (?lin)
          (and (Linear_Path ?lin)
               (holds_shape ?c ?lin))))
```

:IC hard "**Every cylinder holds exactly one linear path.**"

```
(=> (and (Cylinder ?c)
          (Circular_Closed_Profile ?ccp1)
          (Circular_Closed_Profile ?ccp2)
          (Point ?pt1)
          (Point ?pt2)
          (Vector_Direction ?v1)
          (Vector_Direction ?v2)
          (holds_shape ?c ?ccp1)
          (holds_shape ?c ?ccp2)
          (holds_placement ?ccp1 ?p1)
          (holds_placement ?ccp2 ?p2)
          (is_oriented_at ?p1 ?pt1 ?v1)
          (is_oriented_at ?p2 ?pt2 ?v2))
     (exists (?lp)
          (and (Linear_Profile ?lp)
               (holds_axis ?c ?lp)
               (meets ?lp ?pt1 ?pt2))))
```

:IC soft "**Every cylinder may hold an axis which meets the centre points of the two circular closed profiles of the cylinder.**"

```
(=> (Block ?b)
    (exists (?rcp1 ?rcp2  ?rcp3 ?rcp4 ?rcp5 ?rcp6)
          (and (Rectangular_Closed_Profile ?rcp1)
               (Rectangular_Closed_Profile ?rcp2)
               (Rectangular_Closed_Profile ?rcp3)
               (Rectangular_Closed_Profile ?rcp4)
               (Rectangular_Closed_Profile ?rcp5)
               (Rectangular_Closed_Profile ?rcp6)
               (holds_shape ?b ?rcp1)
               (holds_shape ?b ?rcp2)
               (holds_shape ?b ?rcp3)
               (holds_shape ?b ?rcp4)
               (holds_shape ?b ?rcp5)
               (holds_shape ?b ?rcp6))))
```

:IC hard "**Every block holds six rectangular closed profiles.**"

```
(=> (Block ?b)
     (exists (?lin)
(and (Linear_Path ?lin)
               (holds_shape ?b ?lin))))
```

:IC hard "**Every block holds exactly one linear path.**"

```
(=> (and (Round_Hole ?hole)
         (Circular_Closed_Profile ?ccp)
         (Linear_Path ?lin)
         (holds_shape ?hole ?ccp)
         (holds_shape ?hole ?lin)
         (sweeps ?lin ?ccp))
    (exists (?p)
        (and (Placement ?p)
             (holds_placement ?ccp ?p)
             (holds_placement ?lin ?p))))
```
:IC hard "**For a given round hole, the placement of the linear path is the same as the placement of one of the circular closed profiles that its linear path sweeps.**"

```
(=> (and (Round_Hole ?hole)
         (Placement ?p1)
         (holds_orientation ?hole ?p1)
         (Circular_Closed_Profile ?ccp)
         (or (blind ?ccp)
             (through ?ccp))
         (holds_shape ?hole ?ccp)
         (holds_placement ?ccp ?p2))
    (= ?p1 ?p2))
```
:IC hard "**The orientation of a round hole corresponds to the placement of either the blind or through circular closed profile of the hole.**"

```
(=> (and (Round_Hole ?hole)
         (Circular_Closed_Profile ?ccp1)
         (Circular_Closed_Profile ?ccp2)
         (holds_shape ?hole ?ccp1)
         (holds_shape ?hole ?ccp2)
         (or (blind ?ccp2)
             (through ?ccp2))
         (RealNumber ?real1)
         (RealNumber ?real2)
         (measures ?ccp1 (mm ?real1))
         (measures ?ccp2 (mm ?real2))
         (ltNum ?real2 ?real1))
    (exists (?tap)
        (and (Taper ?tap)
             (sweeps ?tap ?ccp1))))
```
:IC soft "**If the nominal diameter of a blind or through circular closed profile of a round hole is less than that of the diameter of the other circular closed profile for the same hole, then a taper parameter that sweeps the non-blind or non-through circular closed profile may be specified.**"

```
(=> (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.**"

```
(=> (and (Round_Hole ?hole)
         (Circular_Closed_Profile ?ccp1)
         (Circular_Closed_Profile ?ccp2)
         (Point ?pt1)
         (Point ?pt2)
         (Vector_Direction ?v1)
         (Vector_Direction ?v2)
         (holds_shape ?hole ?ccp1)
         (holds_shape ?hole ?ccp2)
         (holds_placement ?ccp1 ?p1)
         (holds_placement ?ccp2 ?p2)
         (is_oriented_at ?p1 ?pt1 ?v1)
         (is_oriented_at ?p2 ?pt2 ?v2))
    (exists (?lp)
       (and (Linear_Profile ?lp)
            (holds_axis ?hole ?lp)
            (meets ?lp ?pt1 ?pt2))))
```
:IC soft "**Every round hole feature may hold an axis which meets the centre points of the two circular closed profiles of the hole feature.**"

Definitions

```
(<= (holds_function ?art ?func)
    (and (Feature ?f)
         (Artifact ?art)
         (Function ?func)
         (holds_feature ?art ?f)
         (holds_function ?f ?func)))
```
:rem "**An artifact ?art can hold some function ?func that derives from its feature ?f that holds the function ?func.**"

C.2.5 Transition Features

Classes

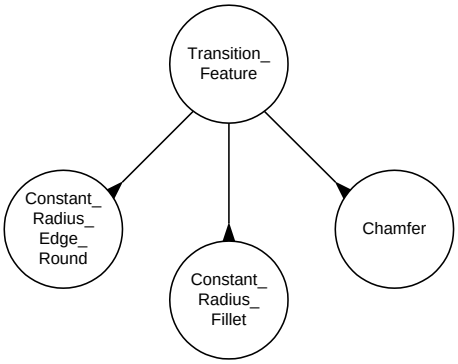


Figure C-23 Transition Feature Classes

:Prop **Constant_Radius_Edge_Round**
:Inst Property
:sup Transition_Feature
:name "Constant Radius Edge Round"
:rem "(Constant_Radius_Edge_Round ?edge) is TRUE in an interpretation of the Foundation Layer if and only if ?edge is a member of a set of constant radius edge rounds. A constant radius edge round intuitively is a type of transition feature that is a convex circular arc transition of constant radius between two intersecting surfaces where the blend surface produced is tangent to both of the adjacent surface edges."

:Prop **Constant_Radius_Fillet**
:Inst Property
:sup Transition_Feature
:name "Constant Radius Fillet"
:rem "(Constant_Radius_Fillet ?fill) is TRUE in an interpretation of the Foundation Layer if and only if ?fill is a member of a set of constant radius fillets. A constant radius fillet intuitively is a type of transition feature that is a concave circular arc transition of constant radius between two intersecting surfaces. The blend surface may be tangent to both of the adjacent surfaces edges."

:Prop **Chamfer**
:Inst Property
:sup Transition_Feature
:name "Chamfer"
:rem "(Chamfer ?chf) is TRUE in an interpretation of the Foundation Layer if and only if ?chf is a member of a set of chamfers. A chamfer intuitively is a type of transition feature that is a transition between two joining non-coplanar surfaces, having a flat orthogonal cross-section. A chamfer description requires an offset length from one face and an offset length from a second face, which forms an angle with respect to the first face."

Relations

:Rel **is_offset_at**
:Inst TernaryRel
:Sig Transition_Feature Length_Measure Shape_Aspect
:name "is_offset_at"

:rem "(is_offset_at ?tf ?length ?sa) is TRUE in an interpretation of the Foundation Layer if and only if the transition feature ?tf is offset at a given length measure ?length with respect to a given shape aspect ?sa."

:Rel **is_angled_at**
:Inst TernaryRel
:Sig Transition_Feature Angle_Measure Shape_Aspect
:name "is_angled_at"
:rem "(is_angled_at ?tf ?angle ?sa) is TRUE in an interpretation of the Foundation Layer if and only if the transition feature ?tf is angled at a given angle measure ?angle with respect to a given shape asepct ?sa."

:Rel **blends**
:Inst BinaryRel
:Sig Transition_Feature Shape_Aspect
:name "blends"
:rem "(blends ?tf ?sa) is TRUE in an interpretation of the Foundation Layer if and only if the transition feature ?tf blends the shape aspects ?sa."

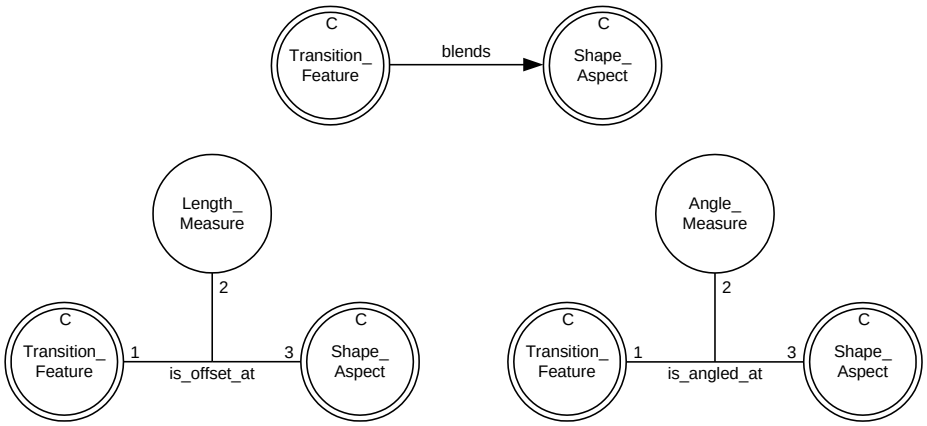


Figure C-24 Transition_Feature Relations

Axioms

```
(=> (Transition_Feature ?tf)
    (Shape_Aspect ?tf))
```
:IC hard "**Transition features are shape aspects.**"

```
(=> (Transition_Feature ?tf)
    (exists (?class)
        (and (RootCtx.sup ?class Transition_Feature)
             (RootCtx.inst ?tf ?class)
             (/= ?class Transition_Feature))))
```
:IC hard "**Any instance of transition feature can only be an instance of one of its subclasses.**"

```
(and (=> (Constant_Radius_Edge_Round ?tf)
         (not (or (Constant_Radius_Fillet ?tf) (Chamfer ?tf))))
     (=> (Constant_Radius_Fillet ?tf)
         (not (Chamfer ?tf))))
```
:IC hard "**Constant radius edge rounds, constant radius fillets and chamfers are all distinct kinds of things.**"

```
(=> (is_offset_at ?tf ?length ?sa)
    (and (Transition_Feature ?tf)
         (Length_Measure ?length)
         (Shape_Aspect ?sa)))
```
:IC hard "**The relation is_offset_at only holds between transition features, length measures and shape aspects.**"

```
(=> (is_angled_at ?tf ?angle ?sa)
    (and (Transition_Feature ?tf)
         (Angle_Measure ?angle)
         (Shape_Aspect ?sa)))
```
:IC hard "**The relation is_angled_at only holds between transition features, angle measures and shape aspects.**"

```
(=> (blends ?tf ?sa)
    (and (Transition_Feature ?tf)
         (Shape_Aspect ?sa)))
```
:IC hard " **The relation blends only holds between transition features and shape aspects.**"

```
(=> (Transition_Feature ?tf)
    (not (exists (?p)
           (and (Placement ?p)
                (holds_placement ?tf ?p)))))
```
:IC hard "**Transition features are shape aspects that do not have placements.**"

```
(=> (Constant_Radius_Edge_Round ?edge)
    (exists (?length)
        (and (Length_Measure ?length)
             (measures ?edge ?length))))
```
:IC hard "**Every constant radius edge round is a transition feature with exactly one length measure which represents the radius of curvature of the transition area.**"

```
(=> (Constant_Radius_Edge_Round ?edge)
    (exists (?sa1 ?sa2)
        (and (Shape_Aspect ?sa1)
             (Shape_Aspect ?sa2)
             (/= ?sa1 ?sa2)
             (blends ?edge ?sa1)
             (blends ?edge ?sa2))))
```
:IC hard "**Every constant radius edge round is a transition feature with exactly two blended shape aspects.**"

```
(=> (Constant_Radius_Fillet ?fill)
    (exists (?length)
        (and (Length_Measure ?length)
             (measures ?fill ?length))))
```
:IC hard "**Every constant radius fillet is a transition feature with exactly one length measure which represents the radius of curvature of the transition area.**"

```
(=> (Constant_Radius_Fillet ?fill)
    (exists (?sa1 ?sa2)
        (and (Shape_Aspect ?sa1)
             (Shape_Aspect ?sa2)
             (/= ?sa1 ?sa2)
             (blends ?fill ?sa1)
             (blends ?fill ?sa2))))
```
:IC hard "**Every constant radius fillet is a transition feature with exactly two blended shape aspects.**"

```
(=> (Constant_Radius_Fillet ?fill)
    (exists (?sa ?length)
        (and (Shape_Aspect ?sa)
             (Length_Measure ?length)
             (blends ?fill ?sa)
             (is_offset_at ?fill ?length ?sa))))
```
:IC hard "**Every constant radius fillet has an offset dimension specification from the shape aspect that it blends.**"

```
(=> (Chamfer ?chf)
    (exists (?sa1 ?sa2)
        (and (Shape_Aspect ?sa1)
             (Shape_Aspect ?sa2)
             (/= ?sa1 ?sa2)
             (blends ?chf ?sa1)
             (blends ?chf ?sa2))))
```
:IC hard "**Every chamfer is a transition feature with exactly two blended shape aspects.**"

```
(=> (Chamfer ?chf)
    (exists (?sa ?length)
        (and (Shape_Aspect ?sa)
             (Length_Measure ?length)
     (blends ?chf ?sa)
             (is_offset_at ?chf ?length ?sa))))
```
:IC hard "**Every chamfer has an offset dimension specification from the shape aspect that it blends.**"

```
(=> (Chamfer ?chf)
    (exists (?sa ?angle)
        (and (Shape_Aspect ?sa)
             (Angle_Measure ?angle)
             (blends ?chf ?sa)
             (is_angled_at ?chf ?angle ?sa))))
```
:IC soft "**Every chamfer may have an angle measure specification from the shape aspect that the chamfer blends.**"

```
(=> (and (Round_Hole ?hole)
         (Circular_Closed_Profile ?ccp)
         (Linear_Path ?lin)
         (holds_shape ?hole ?ccp)
         (holds_shape ?hole ?lin)
         (blind ?ccp)
         (Transition_Feature ?tf)
         (blends ?tf ?ccp)
         (blends ?tf ?lin))
    (or (Chamfer ?tf)
        (Constant_Radius_Fillet ?tf)))
```
:IC hard "**Only chamfers and fillets can be transition features that blend the blind circular closed profile and the linear path of a round hole.**"

```
(=> (and (Round_Hole ?hole)
         (Linear_Path ?lin)
         (holds_shape ?hole ?lin)
         (Feature ?f)
         (Shape_Aspect ?sa)
         (holds_shape ?f ?sa)
         (/= ?hole ?f)
         (Transition_Feature ?tf)
         (blends ?tf ?lin)
         (blends ?tf ?sa))
```

(or (Chamfer ?tf)
(Constant_Radius_Edge_Round ?tf)))
:IC hard "**Only chamfers and edge rounds can be transition features that blend the linear path of a round hole and some other shape aspect from another feature.**"

(=> (and (Cylinder ?c)
(Circular_Closed_Profile ?ccp)
(Linear_Path ?lin)
(holds_shape ?c ?ccp)
(holds_shape ?c ?lin)
(Transition_Feature ?tf)
(blends ?tf ?ccp)
(blends ?tf ?lin))
(or (Chamfer ?tf)
(Constant_Radius_Edge_Round ?tf)))
:IC hard "**Only chamfers and edge rounds can be transition features that blend the linear path and the circular closed profile of a cylinder.**"

(=> (and (Block ?b)
(Rectangular_Closed_Profile ?rcp1)
(Rectangular_Closed_Profile ?rcp2)
(holds_shape ?b ?rcp1)
(holds_shape ?b ?rcp2)
(Transition_Feature ?tf)
(blends ?tf ?rcp1)
(blends ?tf ?rcp2))
(or (Chamfer ?tf)
(Constant_Radius_Edge_Round ?tf)))
:IC hard "**Only chamfers and edge rounds can be transition features that blend two of the rectangular closed profiles of a block.**"

C.2.6 Dimensional Tolerances

Classes

:Prop **Dimensional_Tolerance**
:Inst Property
:sup Core_Property
:name "Dimensional Tolerance"
:rem "(Dimensional_Tolerance ?dtol) is TRUE in an interpretation of the Foundation Layer if and only if ?dtol is a member of a set of dimensional tolerances."

Relations

:Rel **holds_size_tolerance**
:Inst TernaryRel
:Sig Shape_Aspect Dimensional_Tolerance Measure_Item
:name "holds_size_tolerance"
:rem "(holds_size_tolerance ?sa ?dtol ?mea) is TRUE in an interpretation of the Foundation Layer if and only if the shape aspect ?sa holds a given dimensional size tolerance ?dtol with respect to the toleranced measure item ?mea of ?sa."

:Rel **holds_location_tolerance**
:Inst QuaternaryRel
:Sig Feature Dimensional_Tolerance Measure_Item Feature

:name "holds_location_tolerance"
:rem "(holds_location_tolerance ?f1 ?dtol ?mea ?f2) is TRUE in an interpretation of the Foundation Layer if and only if the feature ?f1 holds a given dimensional location tolerance ?dtol with respect to the toleranced measure item ?mea that separates feature ?f1 from another feature ?f2."

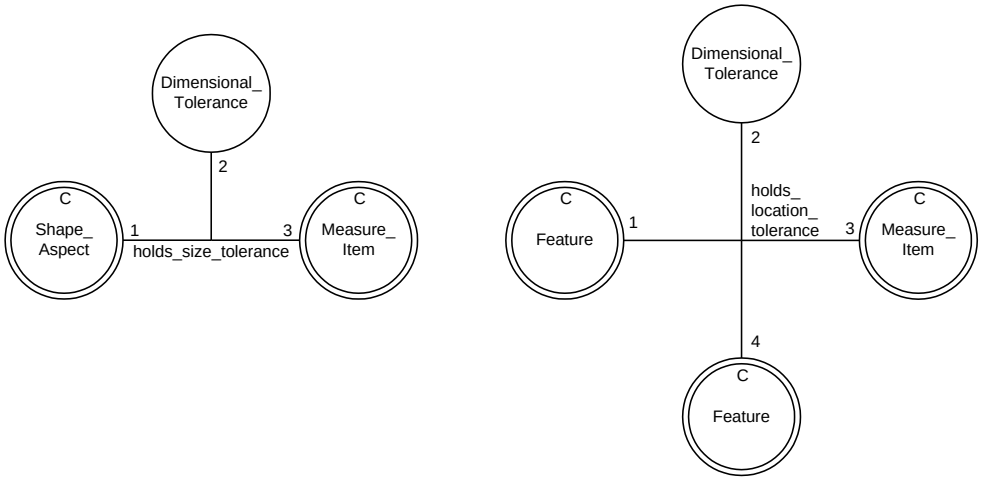


Figure C-25 Dimensional_Tolerance Relations

Functions

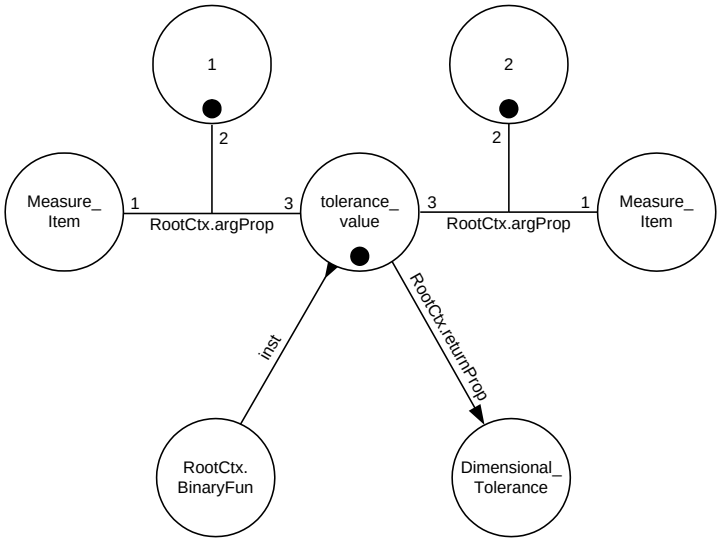


Figure C-26 Dimensional_Tolerance Functions

:Fun **tolerance_value**
:Inst BinaryFun
:Sig Measure_Item Measure_Item -> Dimensional_Tolerance
:name "tolerance_value"
:rem "(= ?dtol (tolerance_value ?mea1 ?mea2)) is TRUE in an interpretation of the Foundation Layer if and only if ?dtol is the dimensional tolerance whose lower-bound value (or minimum value) is given by the measure item ?mea1 and whose upper-bound value (or maximum value) is given by the measure item ?mea2."

Axioms

```
(=> (Dimensional_Tolerance ?dtol)
    (Core_Property ?dtol))
```
:IC hard "**Dimensional tolerances are core properties.**"

```
 (=> (holds_size_tolerance ?sa ?dtol ?mea)
     (and (Shape_Aspect ?sa)
          (Dimensional_Tolerance ?dtol)
          (Measure_Item ?mea)))
```
:IC hard "**The holds_size_tolerance relation only holds between shape aspects, dimensional tolerances and measure items.**"

```
 (=> (holds_location_tolerance ?f1 ?dtol ?mea ?f2)
     (and (Feature ?f1)
          (Feature ?f2)
          (Dimensional_Tolerance ?dtol)
          (Measure_Item ?mea)))
```
:IC hard "**The holds_location_tolerance relation only holds between features, dimensional tolerances and measure items.**"

```
 (=> (and (Dimensional_Tolerance ?dtol)
          (RootCtx.RealNumber ?real1)
          (RootCtx.RealNumber ?real2)
          (or (= ?dtol (tolerance_value (Foundation.mm ?real1) (Foundation.mm ?real2)))
              (= ?dtol (tolerance_value (Foundation.degree ?real1) (Foundation.degree
?real2)))))
      (ltNum ?real1 ?real2))
```
:IC hard "**The lowerbound value of a dimensional tolerance is always numerically less than that of its upperbound value.**"

```
 (=> (and (Dimensional_Tolerance ?dtol)
          (RootCtx.RealNumber ?real1)
          (RootCtx.RealNumber ?real2))
     (not (or (= ?dtol (tolerance_value (Foundation.mm ?real1) (Foundation.degree ?real2)))
              (= ?dtol (tolerance_value (Foundation.degree ?real1) (Foundation.mm ?real2))))))
```
:IC hard "**Both the lowerbound value and upperbound value of a dimensional tolerance have the same unit of measurement function.**"

```
 (=> (Dimensional_Tolerance ?dtol)
     (exists (?mea1 ?mea2)
         (and (Measure_Item ?mea1)
              (Measure_Item ?mea2)
              (= ?dtol (tolerance_value ?mea1 ?mea2)))))
```
:IC hard "**Every dimensional tolerance is given by some lowerbound and upperbound measure value.**"

```
 (=> (and (Circular_Closed_Profile ?ccp)
          (Length_Measure ?length)
          (measures ?ccp ?length)
          (Dimensional_Tolerance ?stol1)
          (Dimensional_Tolerance ?stol2)
          (holds_size_tolerance ?ccp ?stol1 ?length)
          (holds_size_tolerance ?ccp ?stol2 ?length))
     (= ?stol1 ?stol2))
```
:IC hard "**A circular closed profile can only hold a unique size tolerance.**"

```
(=> (and (Rectangular_Closed_Profile ?rcp)
         (Length_Measure ?length1)
         (Length_Measure ?length2)
         (measures ?rcp ?length1)
         (measures ?rcp ?length2)
         (Dimensional_Tolerance ?stol1)
         (Dimensional_Tolerance ?stol2)
         (holds_size_tolerance ?rcp ?stol1 ?length1)
         (holds_size_tolerance ?rcp ?stol2 ?length2))
    (or (= ?stol1 ?stol2)
        (/= ?stol1 ?stol2)))
:IC hard "A rectangular closed profile can hold only two size tolerances."

(=> (and (Linear_Path ?lin)
         (Length_Measure ?length)
         (measures ?lin ?length)
         (Dimensional_Tolerance ?stol1)
         (Dimensional_Tolerance ?stol2)
         (holds_size_tolerance ?lin ?stol1 ?length)
         (holds_size_tolerance ?lin ?stol2 ?length))
    (= ?stol1 ?stol2))
:IC hard "A linear path can only hold a unique size tolerance."

(=> (and (Taper ?tap)
         (Angle_Measure ?angle)
         (measures ?tap ?angle)
         (Dimensional_Tolerance ?stol1)
         (Dimensional_Tolerance ?stol2)
         (holds_size_tolerance ?tap ?stol1 ?angle)
         (holds_size_tolerance ?tap ?stol2 ?angle))
    (= ?stol1 ?stol2))
:IC hard "A taper can only hold a unique size tolerance."

(=> (and (Linear_Profile ?lp)
         (Length_Measure ?length)
         (measures ?lp ?length)
         (Dimensional_Tolerance ?stol1)
         (Dimensional_Tolerance ?stol2)
         (holds_size_tolerance ?lp ?stol1 ?length)
         (holds_size_tolerance ?lp ?stol2 ?length))
    (= ?stol1 ?stol2))
:IC hard "A linear profile can only hold a unique size tolerance."

(=> (and (Round_Hole ?hole)
         (Circular_Closed_Profile ?ccp1)
         (Circular_Closed_Profile ?ccp2)
         (Dimensional_Tolerance ?stol1)
         (Length_Measure ?length1)
         (Length_Measure ?length2)
         (holds_shape ?hole ?ccp1)
         (holds_shape ?hole ?ccp2)
         (measures ?ccp1 ?length1)
         (measures ?ccp2 ?length2)
         (holds_size_tolerance ?ccp1 ?stol1 ?length1)
         (not (exists (?stol2)
                      (and (Dimensional_Tolerance ?stol2)
                           (holds_size_tolerance ?ccp2 ?stol2 ?length2)
                           (/= ?stol1 ?stol2)))))
    (holds_size_tolerance ?ccp2 ?stol1 ?length2))
```

:IC soft "**If one of the circular closed profiles of a round hole has a size tolerance while the other does not, then the same size tolerance may apply to the non-toleranced circular closed profile of the hole.**"

```
(=> (and (Cylinder ?c)
         (Circular_Closed_Profile ?ccp1)
         (Circular_Closed_Profile ?ccp2)
         (Dimensional_Tolerance ?stol1)
         (Length_Measure ?length1)
         (Length_Measure ?length2)
         (holds_shape ?c ?ccp1)
         (holds_shape ?c ?ccp2)
         (measures ?ccp1 ?length1)
         (measures ?ccp2 ?length2)
         (holds_size_tolerance ?ccp1 ?stol1 ?length1)
         (not (exists (?stol2)
                  (and (Dimensional_Tolerance ?stol2)
                       (holds_size_tolerance ?ccp2 ?stol2 ?length2)
                       (/= ?stol1 ?stol2)))))
    (holds_size_tolerance ?ccp2 ?stol1 ?length2))
```

:IC soft "**If one of the circular closed profiles of a cylinder has a size tolerance while the other does not, then the same size tolerance may apply to the non-toleranced circular closed profile of the cylinder.**"

C.3 Flow Objects

Relations

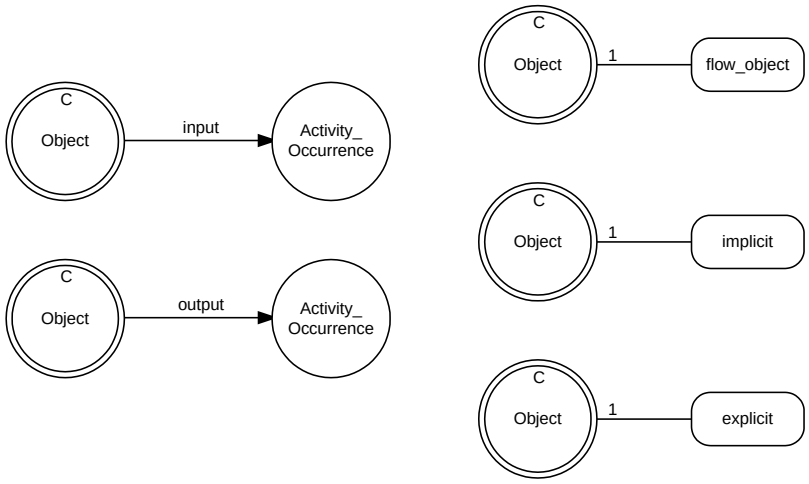


Figure C-27 Flow_Object Relations

:Rel **flow_object**
:Inst UnaryRel
:Sig Object
:name "flow_object"
:rem "(flow_object ?flow) is TRUE in an interpretation of the Foundation Layer if and only if ?flow is an object that participates as a precondition and/or postcondition on activity occurrences."

:Rel **explicit**
:Inst UnaryRel
:Sig Object
:name "explicit"
:rem "(explicit ?flow) is TRUE in an interpretation of the Foundation Layer if and only if the flow object ?flow has been explicitly defined using the relevant necessary conditions. The ?flow object must be an explicitly defined shape aspect, feature or artifact that the user asserts."

:Rel **implicit**
:Inst UnaryRel
:Sig Object
:name "implicit"
:rem "(implicit ?flow) is TRUE in an interpretation of the Foundation Layer if and only if the flow object ?flow has not been explicitly defined using the relevant necessary conditions. The ?flow object is not an explicitly defined shape aspect, feature or artifact that the user asserts."

:Rel **input**
:Inst BinaryRel
:Sig Object Activity_Occurrence
:name "input"
:rem "(input ?flow ?occ) is TRUE in an interpretation of the Foundation Layer if and only if the flow object ?flow is a precondition to an activity occurrence ?occ, which demands that the flow object ?flow is made available to the activity occurrence ?occ in a given way."

:Rel **output**
:Inst BinaryRel
:Sig Object Activity_Occurrence

:name "output"
:rem "(output ?flow ?occ) is TRUE in an interpretation of the Foundation Layer if and only if the flow object ?flow is a postcondition from an activity occurrence ?occ, where the flow object ?flow can participate in other activity occurrences."

Axioms

```
 (=> (flow_object ?flow)
     (Object ?flow))
```
:IC hard "**The relation flow_object only holds for objects.**"

```
 (=> (input ?flow ?occ)
     (and (Object ?flow)
          (flow_object ?flow)
          (Activity_Occurrence ?occ)))
```
:IC hard "**The input relation only holds between flow objects and activity occurrences.**"

```
(=> (output ?flow ?occ)
    (and (Object ?flow)
         (flow_object ?flow)
         (Activity_Occurrence ?occ)))
```
:IC hard "**The output relation only holds between flow objects and activity occurrences.**"

```
 (=> (explicit ?flow)
     (and (Object ?flow)
          (flow_object ?flow)))
```
:IC hard "**The relation explicit only holds for flow objects.**"

```
 (=> (implicit ?flow)
     (and (Object ?flow)
          (flow_object ?flow)))
```
:IC hard "**The relation implicit only holds for flow objects.**"

```
 (=> (and (Object ?flow)
          (flow_object ?flow)
          (Activity_Occurrence ?occ2)
          (input ?flow ?occ2))
     (or (exists (?occ1 ?a)
           (and (Activity_Occurrence ?occ1)
                (Activity ?a)
                (output ?flow ?occ1)
                (or (min_precedes ?occ1 ?occ2 ?a)
                    (next_subocc ?occ1 ?occ2 ?a))
                (/= ?occ1 ?occ2)))
         (exists (?occ)
           (and (Activity_Occurrence ?occ)
                (input ?flow ?occ)
                (subactivity_occurrence ?occ2 ?occ)
                (/= ?occ2 ?occ)))))
```
:IC hard "**An activity occurrence that depends on an input flow object must be either executed after another activity occurrence has provided the input as an output flow object or participate in a complex activity occurrence that requires the flow object as an input.**"

```
 (=> (and (Object ?flow)
          (flow_object ?flow))
     (or (Shape_Aspect ?flow)
         (Feature ?flow)
```

```
        (Artifact ?flow)))
:IC hard "A flow object is a shape aspect, feature or artifact."

(=> (and (Object ?flow)
         (flow_object ?flow))
    (or (explicit ?flow)
        (implicit ?flow)))
:IC hard "A flow object is either an explicitly or implicitly defined object."

(=> (and (Object ?flow)
         (flow_object ?flow)
         (implicit ?flow))
    (not (explicit ?flow)))
:IC hard "An implicit flow object cannot be an explicitly defined object."

(=> (and (Object ?flow)
         (flow_object ?flow)
         (explicit ?flow))
    (not (implicit ?flow)))
:IC hard "An explicit flow object cannot be an implicitly defined object."
```

C.4 Controlled Specialisation Approach

Relations

:Rel **holdsArg**
:Inst TernaryRel
:Sig Relation PosInt Property
:name "holdsArg"
:rem "(holdsArg ?rel ?posInt ?prop) applies if and only if the relation ?rel is an applicable relation that holds for a given argument position ?posInt the argument ?prop. holdsArg is a system relation that binds semantic mapping relations to cross-domain arguments in the order that they appear."

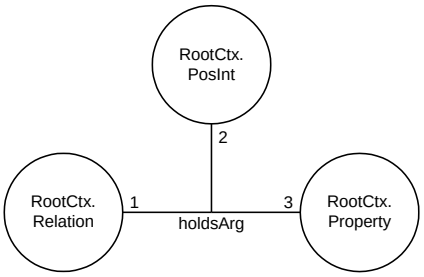


Figure C-28 holdsArg Ternary Relation

Axioms

```
(=> (and (RootCtx.Relation ?rel)
         (RootCtx.withinContext ?rel Foundation)
         (not (RootCtx.Property ?rel)))
    (not (exists (?subrel)
           (and (RootCtx.Relation ?subrel)
                (RootCtx.supRel ?subrel ?rel)))))
```

:IC hard "**Subsumptions involving foundation relations are not permitted.**"

D Domain Ontology Layer

D.1 Machining Hole Feature Ontology A

Context Declaration

:Ctx machiningHoleFeatureOntologyA
:Inst UserContext
:supCtx TopUserCtx
:name "Context for the Machining Hole Feature Ontology A"
:rem "This context explores the integrity-driven domain ontology development for hole features defined from a machining process viewpoint using the semantics from the Foundation Layer."

:Use machiningHoleFeatureOntologyA

Classes

:Prop **Housing_Part_Family**
:Inst Property
:sup Foundation.**Artifact**
:name "Housing_Part_Family"
:rem "A housing part family is a type of artifact which is manufactured through a series of turning and hole making machining processes."

:Prop **Centre_Drilled_Hole**
:Inst Property
:sup Foundation.**Round_Hole**
:name "Centre_Drilled_Hole"
:rem "A centre drilled hole is a round hole feature which is machined using a centre drilling process."

:Prop **Counterbore**
:Inst Property
:sup Foundation.**Round_Hole**
:name "Counterbore"
:rem "A counterbore is a round hole feature which is machined using a counterboring process."

:Prop **Counterbore_Hole**
:Inst Property
:sup Foundation.**Feature**
:name "Counterbore_Hole"
:rem "A counterbore hole is a compound hole feature which is machined using a sequence of centre-drilling, followed by drilling, followed by counterboring processes."

:Prop **Drilled_Hole**
:Inst Property
:sup Foundation.**Round_Hole**
:name "Drilled_Hole"
:rem "A drilled hole is a round hole feature which is machined using a sequence of centre-drilling, followed by drilling processes."

:Prop **Turned_Boss**
:Inst Property
:sup Foundation.**Cylinder**
:name "Turned_Boss"
:rem "A turned boss is a cylindrical feature which makes up a housing and is machined using turning processes."

:Prop **Turned_Flange**
:Inst Property
:sup Foundation.**Cylinder**
:name "Turned_Flange"
:rem "A turned flange is a cylindrical feature which makes up a housing and is machined using turning processes."

:Prop **Reamed_Hole**
:Inst Property
:sup Foundation.**Round_Hole**
:name "Reamed_Hole"
:rem "A reamed hole is a round hole feature which is machined using a sequence of centre-drilling, followed by drilling, followed by reaming processes."

:Prop **Drilled_Hole_Depth**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Drilled_Hole_Depth"
:rem "A drilled hole depth is the length measure for the overall depth of a drilled hole."

:Prop **Drilled_Hole_Diameter**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Drilled_Hole_Diameter"
:rem "A drilled hole diameter is the length measure for the diameter of a drilled hole."

:Prop **Counterbore_Depth**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Counterbore_Depth"
:rem "A counterbore depth is the length measure for the depth of a counterbore."

:Prop **Counterbore_Diameter**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Counterbore_Diameter"
:rem "A counterbore diameter is the length measure for the diameter of a counterbore."

:Prop **Reamed_Hole_Depth**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Reamed_Hole_Depth"
:rem "A reamed hole depth is the length measure for the overall depth of a reamed hole."

:Prop **Reamed_Hole_Diameter**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Reamed_Hole_Diameter"
:rem "A reamed hole diameter is the length measure for the diameter of a reamed hole."

:Prop **Centre_Drilling**
:Inst Property
:sup Foundation.**Activity**

:name "Centre_Drilling"
:rem "A centre drilling activity is a reusable process behaviour whose occurrences produce centre-drilled holes as outputs. An occurrence of a centre drilling activity, for which a centre drilled hole is output, is an atomic activity occurrence."

:Prop **Counterboring**
:Inst Property
:sup Foundation.Activity
:name "Counterboring"
:rem "A counterboring activity is a reusable process behaviour whose occurrences produce centre-drilled holes as outputs. An occurrence of a counterboring activity, for which a counterbore is output, is an atomic activity occurrence."

:Prop **Drilling**
:Inst Property
:sup Foundation.**Activity**
:name "Drilling"
:rem "A drilling activity is a reusable process behaviour whose occurrences produce drilled holes as outputs. An occurrence of a drilling activity, for which a drilled hole is output, is an atomic activity occurrence."

:Prop **Reaming**
:Inst Property
:sup Foundation.**Activity**
:name "Reaming"
:rem "A reaming activity is a reusable process behaviour whose occurrences produce reamed holes as outputs. An occurrence of a reaming activity, for which a reamed hole is output, is an atomic activity occurrence."

:Prop **Counterbore_Hole_Making**
:Inst Property
:sup Foundation.**Activity**
:name "Counterbore_Hole_Making"
:rem "A counterbore hole making activity is a reusable process behaviour whose occurrences produce counterbore holes as outputs. An occurrence of a counterbore hole making activity, for which a counterbore 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 counterboring."

:Prop **Reamed_Hole_Making**
:Inst Property
:sup Foundation.**Activity**
:name "Reamed_Hole_Making"
:rem "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."

:Prop **Aluminium**
:Inst Property
:sup Foundation.**Material**
:name "Aluminium"
:rem "Aluminium is a material that represents the chemical element aluminium, which is a silvery ductile metallic element found primarily in bauxite."

Functions

:Fun **inch**
:Inst UnaryFun
:Sig **RealNumber** -> Foundation.**Length_Measure**
:name "inch"
:rem "(= ?length (inch ?real)) is used to denote the value of a length measure in inches."

Axioms

```
(=> (Housing_Part_Family ?house)
    (exists (?flange ?boss ?cbore ?dhole ?rhole)
        (and (Turned_Flange ?flange)
             (Turned_Boss ?boss)
             (Counterbore_Hole ?cbore)
             (Drilled_Hole ?dhole)
             (Reamed_Hole ?rhole)
             (Foundation.holds_feature ?house ?flange)
             (Foundation.holds_feature ?house ?boss)
             (Foundation.holds_feature ?house ?cbore)
             (Foundation.holds_feature ?house ?dhole)
             (Foundation.holds_feature ?house ?rhole))))
```

:IC hard "**Every housing has some compulsory turned flange, turned boss, counterbore hole, drilled hole and reamed hole as features present on the housing.**"

```
(=> (Housing_Part_Family ?house)
    (exists (?al)
        (and (Aluminium ?al)
             (Foundation.holds_material ?house ?al))))
```

:IC hard "**Every housing is made up of some aluminium material.**"

```
(=> (and (Centre_Drilled_Hole ?cDrillHole)
         (Foundation.flow_object ?cDrillHole))
    (exists (?cDrill ?cDrillOcc)
        (and (Centre_Drilling ?cDrill)
             (Foundation.Activity_Occurrence ?cDrillOcc)
             (Foundation.occurrence_of ?cDrillOcc ?cDrill)
             (Foundation.output ?cDrillHole ?cDrillOcc))))
```

:IC soft "**Every centre drilled hole that is a flow object is an output from a potential occurrence of a centre drilling activity.**"

```
(=> (Counterbore ?chole)
    (exists (?ccp1 ?ccp2 ?cdia1 ?cdia2)
        (and (Foundation.Circular_Closed_Profile ?ccp1)
             (Foundation.Circular_Closed_Profile ?ccp2)
             (Foundation.holds_shape ?chole ?ccp1)
             (Foundation.holds_shape ?chole ?ccp2)
             (Counterbore_Diameter ?cdia1)
             (Counterbore_Diameter ?cdia2)
             (Foundation.measures ?ccp1 ?cdia1)
             (Foundation.measures ?ccp2 ?cdia2)
             (= ?cdia1 ?cdia2))))
```

:IC hard "**Every counterbore holds exactly two circular closed profiles of identical counterbore diameter.**"

```
(=> (Counterbore ?chole)
    (exists (?lin ?cdepth)
        (and (Foundation.Linear_Path ?lin)
            (Foundation.holds_shape ?chole ?lin)
            (Counterbore_Depth ?cdepth)
            (Foundation.measures ?lin ?cdepth))))
```
:IC hard "**Every counterbore holds exactly one linear path of counterbore depth.**"

```
(=> (and (Counterbore ?chole)
         (Foundation.flow_object ?chole))
    (exists (?cbore ?cboreOcc)
        (and (Counterboring ?cbore)
            (Foundation.Activity_Occurrence ?cboreOcc)
            (Foundation.occurrence_of ?cboreOcc ?cbore)
            (Foundation.output ?chole ?cboreOcc))))
```
:IC soft "**Every counterbore that is a flow object is an output from a potential occurrence of a counterboring activity.**"

```
(=> (Counterbore_Hole ?cbhole)
    (Foundation.compound ?cbhole))
```
:IC hard "**A counterbore hole is a compound feature.**"

```
(=> (Counterbore_Hole ?cbhole)
    (exists (?dhole ?chole)
        (and (Drilled_Hole ?dhole)
            (Counterbore ?chole)
            (Foundation.element_of ?dhole ?cbhole)
            (Foundation.element_of ?chole ?cbhole))))
```
:IC hard "**Every counterbore hole involves a drilled hole and a counterbore which are elements of the counterbore hole.**"

```
(=> (and (Counterbore_Hole ?cbhole)
         (Drilled_Hole ?dhole)
         (Foundation.element_of ?dhole ?cbhole))
    (Foundation.base ?dhole))
```
:IC hard "**The drilled hole of a counterbore hole is the base feature of the counterbore hole.**"

```
(=> (and (Counterbore_Hole ?cbhole)
         (Drilled_Hole ?dhole)
         (Counterbore ?chole)
         (Foundation.element_of ?dhole ?cbhole)
         (Foundation.element_of ?chole ?cbhole)
         (Foundation.Circular_Closed_Profile ?ccp1)
         (Foundation.Circular_Closed_Profile ?ccp2)
         (Foundation.holds_shape ?dhole ?ccp1)
         (Foundation.holds_shape ?chole ?ccp2))
    (exists (?real1 ?real2)
        (and (RootCtx.RealNumber ?real1)
            (RootCtx.RealNumber ?real2)
            (Foundation.measures ?ccp1 (Foundation.mm ?real1))
            (Foundation.measures ?ccp2 (Foundation.mm ?real2))
            (/= ?real1 ?real2)
            (gtNum ?real2 ?real1))))
```
:IC hard "**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.**"

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

```
        (Foundation.element_of ?dhole ?cbhole)
        (Foundation.element_of ?chole ?cbhole)
        (Foundation.Linear_Path ?lin1)
        (Foundation.Linear_Path ?lin2)
        (Foundation.holds_shape ?dhole ?lin1)
        (Foundation.holds_shape ?chole ?lin2))
    (exists (?real1 ?real2)
        (and (RootCtx.RealNumber ?real1)
            (RootCtx.RealNumber ?real2)
            (Foundation.measures ?lin1 (Foundation.mm ?real1))
            (Foundation.measures ?lin2 (Foundation.mm ?real2))
            (/= ?real1 ?real2)
            (gtNum ?real1 ?real2))))
```

:IC hard "**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.**"

```
(=> (and (Counterbore_Hole ?cbhole)
        (Foundation.flow_object ?cbhole))
    (exists (?cboreMake ?cboreMakeOcc)
        (and (Counterbore_Hole_Making ?cboreMake)
            (Foundation.Activity_Occurrence ?cboreMakeOcc)
            (Foundation.occurrence_of ?cboreMakeOcc ?cboreMake)
            (Foundation.output ?cbhole ?cboreMakeOcc))))
```

:IC soft "**Every compound counterbore hole that is a flow object is an output from a potential occurrence of a complex counterbore hole making activity.**"

```
(=> (Drilled_Hole ?dhole)
    (exists (?ccp1 ?ccp2 ?dhdia1 ?dhdia2)
        (and (Foundation.Circular_Closed_Profile ?ccp1)
            (Foundation.Circular_Closed_Profile ?ccp2)
            (Foundation.holds_shape ?dhole ?ccp1)
            (Foundation.holds_shape ?dhole ?ccp2)
            (Drilled_Hole_Diameter ?dhdia1)
            (Drilled_Hole_Diameter ?dhdia2)
            (Foundation.measures ?ccp1 ?dhdia1)
            (Foundation.measures ?ccp2 ?dhdia2)
            (= ?dhdia1 ?dhdia2))))
```

:IC hard "**Every drilled hole holds exactly two circular closed profiles of identical drilled hole diameter.**"

```
(=> (Drilled_Hole ?dhole)
    (exists (?lin ?dhdepth)
        (and (Foundation.Linear_Path ?lin)
            (Foundation.holds_shape ?dhole ?lin)
            (Drilled_Hole_Depth ?dhdepth)
            (Foundation.measures ?lin ?dhdepth))))
```

:IC hard "**Every drilled hole holds exactly one linear path of drilled hole depth.**"

```
(=> (and (Drilled_Hole ?dhole)
        (Foundation.flow_object ?dhole))
    (exists (?drill ?drillOcc)
        (and (Drilling ?drill)
            (Foundation.Activity_Occurrence ?drillOcc)
            (Foundation.occurrence_of ?drillOcc ?drill)
            (Foundation.output ?dhole ?drillOcc))))
```

:IC soft "**Every drilled hole that is a flow object is an output from a potential occurrence of a drilling activity.**"

```
(=> (Reamed_Hole ?rhole)
    (exists (?ccp1 ?ccp2 ?rhdia1 ?rhdia2)
        (and (Foundation.Circular_Closed_Profile ?ccp1)
             (Foundation.Circular_Closed_Profile ?ccp2)
             (Foundation.holds_shape ?rhole ?ccp1)
             (Foundation.holds_shape ?rhole ?ccp2)
             (Reamed_Hole_Diameter ?rhdia1)
             (Reamed_Hole_Diameter ?rhdia2)
             (Foundation.measures ?ccp1 ?rhdia1)
             (Foundation.measures ?ccp2 ?rhdia2)
             (= ?rhdia1 ?rhdia2))))
```
:IC hard "**Every reamed hole holds exactly two circular closed profiles of identical reamed hole diameter.**"

```
(=> (Reamed_Hole ?rhole)
    (exists (?lin ?rhdepth)
        (and (Foundation.Linear_Path ?lin)
             (Foundation.holds_shape ?rhole ?lin)
             (Reamed_Hole_Depth ?rhdepth)
             (Foundation.measures ?lin ?rhdepth))))
```
:IC hard "**Every reamed hole holds exactly one linear path of reamed hole depth.**"

```
(=> (and (Reamed_Hole ?rhole)
         (Foundation.Linear_Path ?lin)
         (Foundation.Circular_Closed_Profile ?ccp)
         (Foundation.through ?ccp)
         (Foundation.holds_shape ?rhole ?lin)
         (Foundation.holds_shape ?rhole ?ccp))
    (exists (?chf1 ?chf2)
        (and (Foundation.Chamfer ?chf1)
             (Foundation.Chamfer ?chf2)
             (Foundation.blends ?chf1 ?lin)
             (Foundation.blends ?chf2 ?lin))))
```
:IC hard "**Every reamed hole that has a through hole bottom condition requires two chamfers that blend the linear path of the reamed hole.**"

```
(=> (and (Reamed_Hole ?rhole)
         (Foundation.flow_object ?rhole))
    (exists (?rholeMake ?rholeMakeOcc ?ream ?reamOcc)
        (and (Reamed_Hole_Making ?rholeMake)
             (Reaming ?ream)
             (Foundation.Activity_Occurrence ?rholeMakeOcc)
             (Foundation.Activity_Occurrence ?reamOcc)
             (Foundation.occurrence_of ?rholeMakeOcc ?rholeMake)
             (Foundation.occurrence_of ?reamOcc ?ream)
             (Foundation.output ?rhole ?rholeMakeOcc)
             (Foundation.output ?rhole ?reamOcc))))
```
:IC soft "**Every reamed hole that is a flow object is both an output from a potential occurrence of a complex reamed hole making activity and an output from a potential occurrence of an atomic reaming activity.**"

```
(=> (and (Centre_Drilling ?cdrill)
         (Foundation.Activity_Occurrence ?cdrillOcc)
         (Foundation.occurrence_of ?cdrillOcc ?cdrill)
         (Foundation.legal ?cdrillOcc)
         (Drilling ?drill))
    (Foundation.legal (Foundation.successor ?drill ?cdrillOcc)))
```
:IC soft "**If an occurrence of centre drilling is allowed, then an occurrence of drilling immediately after it may be allowed.**"

```
(=> (and (Drilling ?drill)
         (Foundation.Activity_Occurrence ?drillOcc)
         (Foundation.occurrence_of ?drillOcc ?drill)
         (Foundation.legal ?drillOcc)
         (Counterboring ?cboring))
    (Foundation.legal (Foundation.successor ?cboring ?drillOcc)))
```
:IC soft "**If an occurrence of drilling is allowed, then an occurrence of counterboring immediately after it may be allowed.**"

```
(=> (and (Drilling ?drill)
         (Foundation.Activity_Occurrence ?drillOcc)
         (Foundation.occurrence_of ?drillOcc ?drill)
         (Foundation.legal ?drillOcc)
         (Reaming ?ream))
    (Foundation.legal (Foundation.successor ?ream ?drillOcc)))
```
:IC soft "**If an occurrence of drilling is allowed, then an occurrence of reaming immediately after it may be allowed.**"

```
(=> (and (Counterbore_Hole_Making ?cboreMake)
         (Foundation.Activity_Occurrence ?cboreMakeOcc)
         (Foundation.occurrence_of ?cboreMakeOcc ?cboreMake))
    (exists (?cDrill ?drill ?cDrillOcc ?drillOcc)
        (and (Centre_Drilling ?cDrill)
             (Drilling ?drill)
             (Foundation.Activity_Occurrence ?cDrillOcc)
             (Foundation.Activity_Occurrence ?drillOcc)
             (Foundation.occurrence_of ?cDrillOcc ?cDrill)
             (Foundation.occurrence_of ?drillOcc ?drill)
             (Foundation.min_precedes ?cDrillOcc ?drillOcc ?cboreMake))))
```
:IC hard "**An occurrence of centre drilling must precede an occurrence of drilling under a complex occurrence of counterbore hole making. Other behaviours under the complex counterbore hole making activity may occur in between.**"

```
(=> (and (Counterbore_Hole_Making ?cboreMake)
         (Foundation.Activity_Occurrence ?cboreMakeOcc)
         (Foundation.occurrence_of ?cboreMakeOcc ?cboreMake))
    (exists (?drill ?cbore ?drillOcc ?cboreOcc)
        (and (Drilling ?drill)
             (Counterboring ?cbore)
             (Foundation.Activity_Occurrence ?drillOcc)
             (Foundation.Activity_Occurrence ?cboreOcc)
             (Foundation.occurrence_of ?drillOcc ?drill)
             (Foundation.occurrence_of ?cboreOcc ?cbore)
             (Foundation.min_precedes ?drillOcc ?cboreOcc ?cboreMake))))
```
:IC hard "**An occurrence of drilling must precede an occurrence of counterboring under a complex occurrence of counterbore hole making. Other behaviours under the complex counterbore hole making activity may occur in between.**"

```
(=> (and (Counterbore_Hole_Making ?cboreMake)
         (Foundation.Activity_Occurrence ?cboreMakeOcc)
         (Foundation.occurrence_of ?cboreMakeOcc ?cboreMake))
    (exists (?cDrill ?cDrillOcc)
        (and (Centre_Drilling ?cDrill)
             (Foundation.subactivity ?cDrill ?cboreMake)
             (Foundation.Activity_Occurrence ?cDrillOcc)
             (Foundation.occurrence_of ?cDrillOcc ?cDrill)
             (Foundation.subactivity_occurrence ?cDrillOcc ?cboreMakeOcc)
             (Foundation.root_occ ?cDrillOcc ?cboreMakeOcc))))
```
:IC hard "**An occurrence of centre drilling under a complex occurrence of counterbore hole making must be at the extreme beginning of the complex occurrence.**"

```
(=> (and (Counterbore_Hole_Making ?cboreMake)
         (Foundation.Activity_Occurrence ?cboreMakeOcc)
         (Foundation.occurrence_of ?cboreMakeOcc ?cboreMake))
    (exists (?cbore ?cboreOcc)
        (and (Counterboring ?cbore)
             (Foundation.subactivity ?cbore ?cboreMake)
             (Foundation.Activity_Occurrence ?cboreOcc)
             (Foundation.occurrence_of ?cboreOcc ?cbore)
             (Foundation.subactivity_occurrence ?cboreOcc ?cboreMakeOcc)
             (Foundation.leaf_occ ?cboreOcc ?cboreMakeOcc))))
```
:IC hard "**An occurrence of counterboring under a complex occurrence of counterbore hole making must be at the extreme end of the complex occurrence.**"

```
(=> (and (Reamed_Hole_Making ?rholeMake)
         (Foundation.Activity_Occurrence ?rholeMakeOcc)
         (Foundation.occurrence_of ?rholeMakeOcc ?rholeMake))
    (exists (?cDrill ?drill ?cDrillOcc ?drillOcc)
        (and (Centre_Drilling ?cDrill)
             (Drilling ?drill)
             (Foundation.Activity_Occurrence ?cDrillOcc)
             (Foundation.Activity_Occurrence ?drillOcc)
             (Foundation.occurrence_of ?cDrillOcc ?cDrill)
             (Foundation.occurrence_of ?drillOcc ?drill)
             (Foundation.min_precedes ?cDrillOcc ?drillOcc ?rholeMake))))
```
:IC hard "**An occurrence of centre drilling must precede an occurrence of drilling under a complex occurrence of reamed hole making. Other behaviours under the complex reamed hole making activity may occur in between.**"

```
(=> (and (Reamed_Hole_Making ?rholeMake)
         (Foundation.Activity_Occurrence ?rholeMakeOcc)
         (Foundation.occurrence_of ?rholeMakeOcc ?rholeMake))
    (exists (?drill ?ream ?drillOcc ?reamOcc)
        (and (Drilling ?drill)
             (Reaming ?ream)
             (Foundation.Activity_Occurrence ?drillOcc)
             (Foundation.Activity_Occurrence ?reamOcc)
             (Foundation.occurrence_of ?drillOcc ?drill)
             (Foundation.occurrence_of ?reamOcc ?ream)
             (Foundation.min_precedes ?drillOcc ?reamOcc ?rholeMake))))
```
:IC hard "**An occurrence of drilling must precede an occurrence of reaming under a complex occurrence of reamed hole making. Other behaviours under the complex reamed hole making activity may occur in between.**"

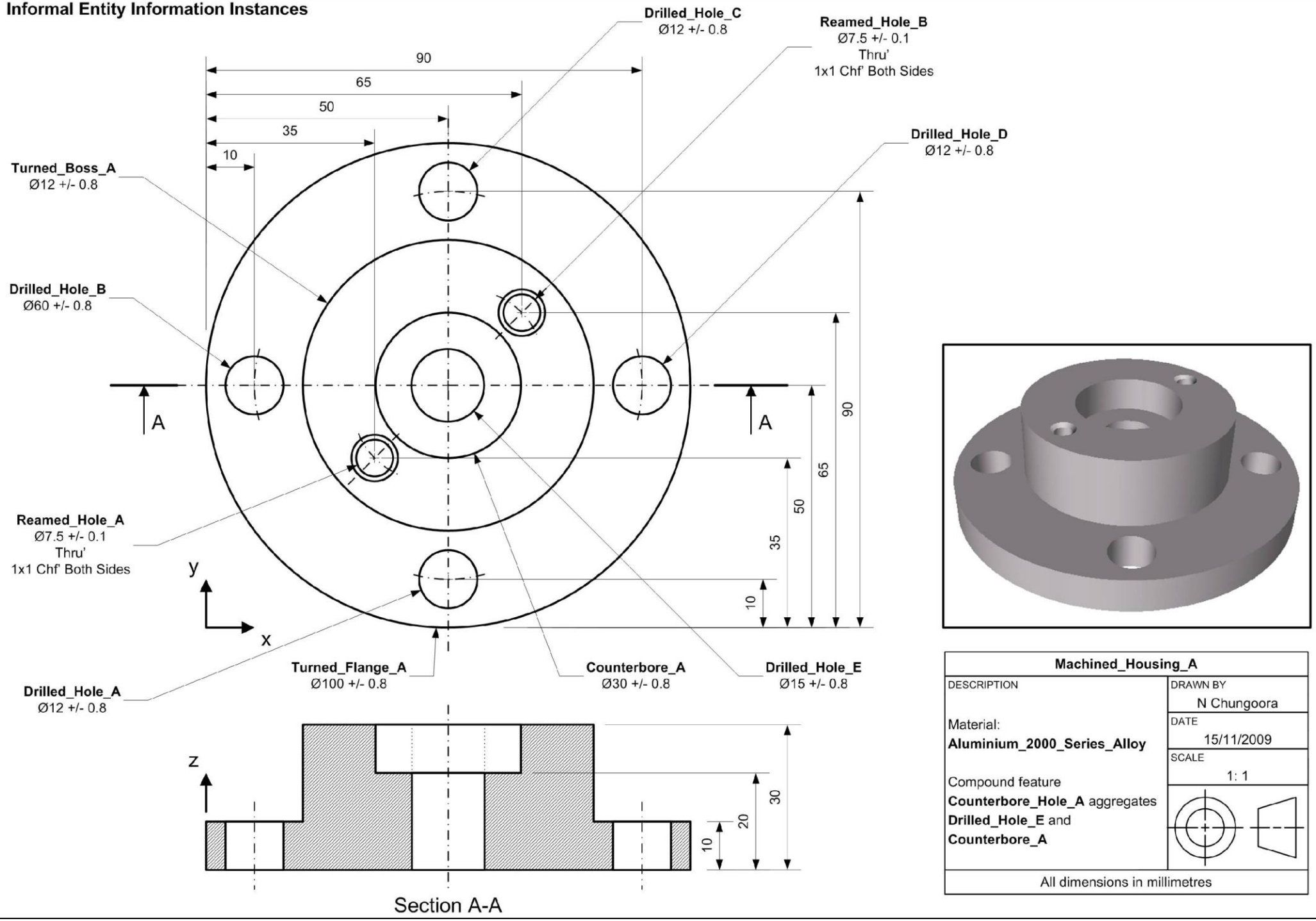
```
(=> (and (Reamed_Hole_Making ?rholeMake)
         (Foundation.Activity_Occurrence ?rholeMakeOcc)
         (Foundation.occurrence_of ?rholeMakeOcc ?rholeMake))
    (exists (?cDrill ?cDrillOcc)
        (and (Centre_Drilling ?cDrill)
             (Foundation.subactivity ?cDrill ?rholeMake)
             (Foundation.Activity_Occurrence ?cDrillOcc)
             (Foundation.occurrence_of ?cDrillOcc ?cDrill)
             (Foundation.subactivity_occurrence ?cDrillOcc ?rholeMakeOcc)
             (Foundation.root_occ ?cDrillOcc ?rholeMakeOcc))))
```
:IC hard "**An occurrence of centre drilling under a complex occurrence of reamed hole making must be at the extreme beginning of the complex occurrence.**"

```
(=> (and (Reamed_Hole_Making ?rholeMake)
         (Foundation.Activity_Occurrence ?rholeMakeOcc)
         (Foundation.occurrence_of ?rholeMakeOcc ?rholeMake))
    (exists (?ream ?reamOcc)
```

```
    (and (Reaming ?ream)
        (Foundation.subactivity ?ream ?rholeMake)
        (Foundation.Activity_Occurrence ?reamOcc)
        (Foundation.occurrence_of ?reamOcc ?ream)
        (Foundation.subactivity_occurrence ?reamOcc ?rholeMakeOcc)
        (Foundation.leaf_occ ?reamOcc ?rholeMakeOcc))))
```

:IC hard "**An occurrence of reaming under a complex occurrence of reamed hole making must be at the extreme end of the complex occurrence.**"

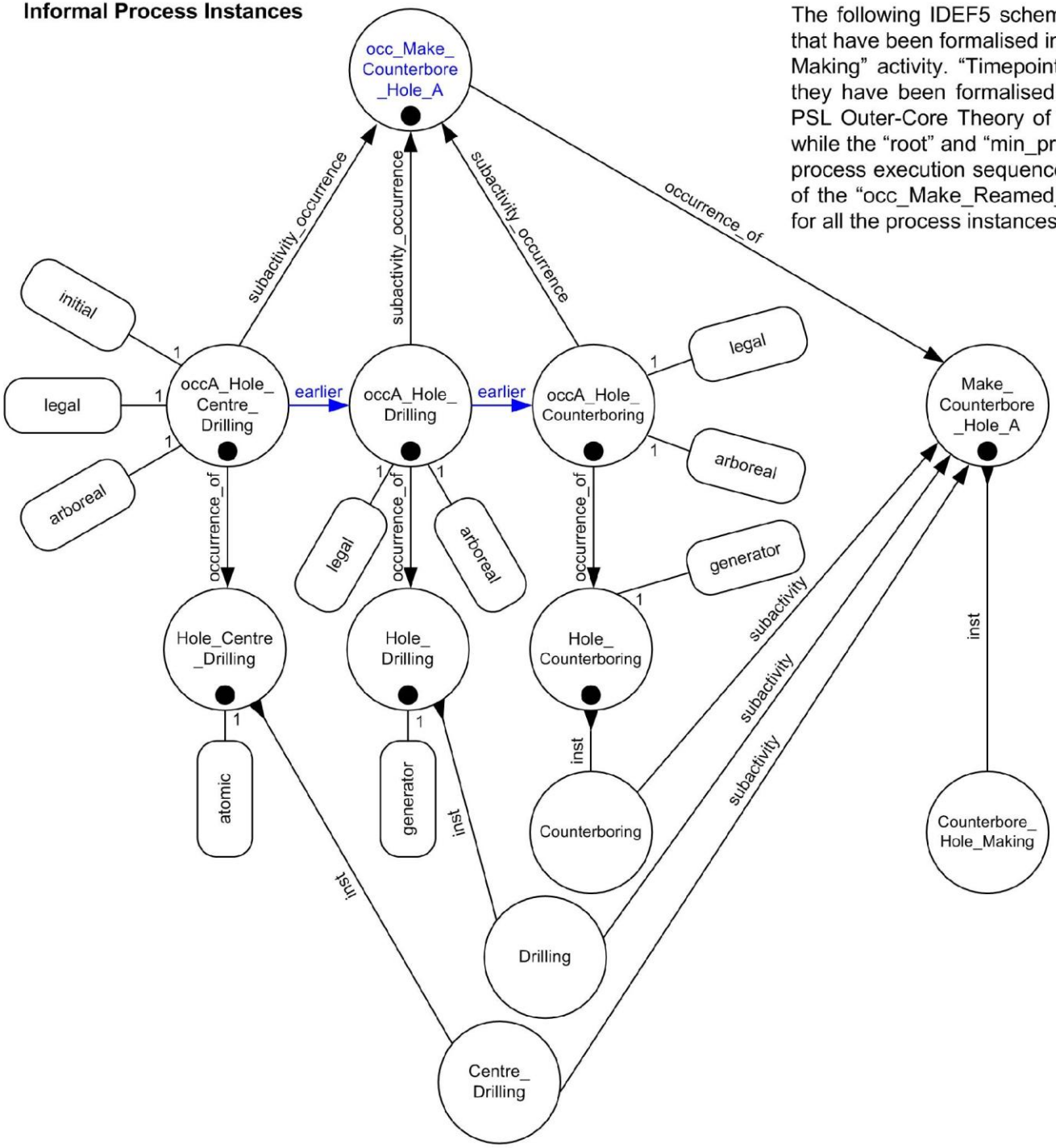
Informal Entity Information Instances

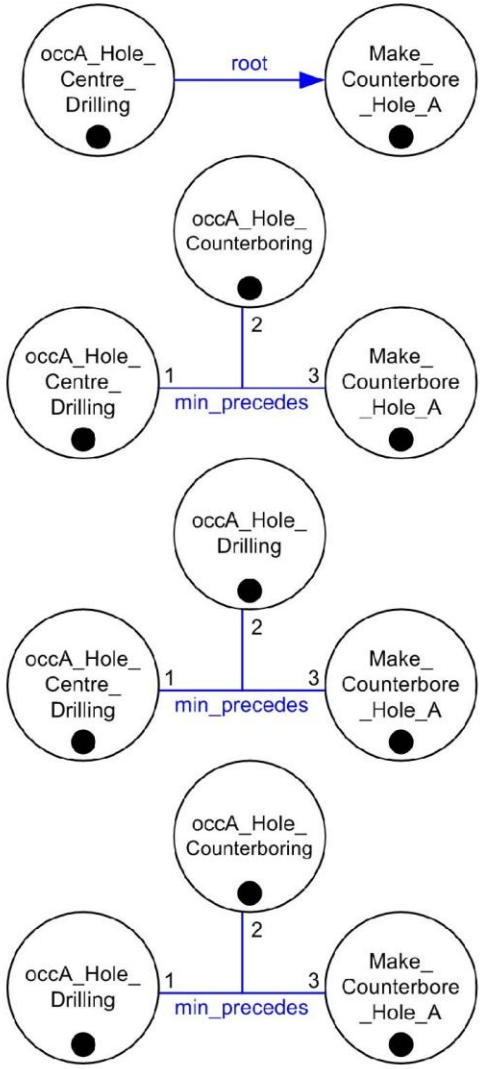


Machined_Housing_A	
DESCRIPTION Material: **Aluminium_2000_Series_Alloy** Compound feature **Counterbore_Hole_A** aggregates **Drilled_Hole_E** and **Counterbore_A**	DRAWN BY N Chungoora DATE 15/11/2009 SCALE 1: 1
All dimensions in millimetres	

Informal Process Instances

The following IDEF5 schematics illustrate some of the important linear ordering semantics that have been formalised in SCL for an occurrence of a complex "Counterbore_Hole_Making" activity. "Timepoints" have not been illustrated in the IDEF5 schematics although they have been formalised at implementation level. The binary relation "earlier", from the PSL Outer-Core Theory of Occurrence Trees, is initially used to provide a partial ordering while the "root" and "min_precedes" relations help support the more rigorous linearly ordered process execution sequence. A similar understanding has been applied for the formalisation of the "occ_Make_Reamed_Holes_AB" in "Machining Hole Feature Ontology A" as well as for all the process instances in "Machining Hole Feature Ontology B"





D.2 Design Hole Feature Ontology A

Context Declaration

:Ctx designHoleFeatureOntologyA
:Inst UserContext
:supCtx TopUserCtx
:name "Context for the Design Hole Feature Ontology A"
:rem "This context captures an ontology for hole features defined from a functional design viewpoint using the semantics from the Foundation Layer."

:Use designHoleFeatureOntologyA

Classes

:Prop **Housing_Part_Family**
:Inst Property
:sup Foundation.**Artifact**
:name "Housing_Part_Family"
:rem "A housing part family is a type of artifact which is manufactured through a series of turning and hole making machining processes."

:Prop **Bolt_Hole**
:Inst Property
:sup Foundation.**Feature**
:name "Bolt_Hole"
:rem "A bolt hole is a compound hole feature which is composed of a plain diameter hole and a secondary hole."

:Prop **Boss**
:Inst Property
:sup Foundation.**Cylinder**
:name "Boss"
:rem "A boss is a cylinder of compound property which is composed of a cylinder and a round hole."

:Prop **External_Flange**
:Inst Property
:sup Foundation.**Cylinder**
:name "External_Flange"
:rem "An external flange is a cylindrical feature which makes up a housing."

:Prop **Locating_Pin_Hole**
:Inst Property
:sup Foundation.**Round_Hole**
:name "Locating_Pin_Hole"
:rem "A locating pin hole is a round hole feature whose function is to provide an accurate positioning of a housing."

:Prop **Plain_Diameter_Hole**
:Inst Property
:sup Foundation.**Round_Hole**
:name "Plain_Diameter_Hole"
:rem "A plain diameter hole is a round hole feature which may be a standalone hole or an element of a bolt hole."

:Prop **Secondary_Hole**
:Inst Property
:sup Foundation.**Round_Hole**
:name "Secondary_Hole"
:rem "A secondary hole is a round hole feature which is an element of a bolt hole."

:Prop **Boss_Height**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Boss_Height"
:rem "A boss height is the length measure for the height of a boss."

:Prop **Boss_Diameter**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Boss_Diameter"
:rem "A boss diameter is the length measure for the diameter of a boss."

:Prop **Flange_Thickness**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Flange_Thickness"
:rem "A flange thickness is the length measure for the height of an external flange."

:Prop **Flange_Diameter**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Flange_Diameter"
:rem "A flange diameter is the length measure for the diameter of an external flange."

:Prop **Primary_Depth**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Primary_Depth"
:rem "A primary depth is the length measure for the overall depth of a plain diameter hole or locating pin hole."

:Prop **Primary_Diameter**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Primary_Diameter"
:rem "A primary diameter is the length measure for the diameter of a plain diameter hole or locating pin hole."

:Prop **Secondary_Depth**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Secondary_Depth"
:rem "A secondary depth is the length measure for the depth of a secondary hole."

:Prop **Secondary_Diameter**
:Inst Property
:sup Foundation.**Length_Measure**
:name "Secondary_Diameter"
:rem "A secondary diameter is the length measure for the diameter of a secondary hole."

:Prop **Aluminium**
:Inst Property
:sup Foundation.**Material**
:name "Aluminium"

:rem "Aluminium is a material that represents the chemical element aluminium, which is a silvery ductile metallic element found primarily in bauxite."

:Prop **Design_Function**
:Inst Property
:sup Foundation.**Function**
:name "Design_Function"
:rem "A design function represents the intended purpose of a core entity defined in the Design Hole Feature Ontology A."

Functions

:Fun **inch**
:Inst UnaryFun
:Sig **RealNumber** -> **Foundation.Length_Measure**
:name "inch"
:rem "(= ?length (inch ?real)) is used to denote the value of a length measure in inches."

Axioms

```
(=> (and (Foundation.Feature ?f)
         (RootCtx.withinContext ?f designHoleFeatureOntologyA)
         (Foundation.Artifact ?art)
         (RootCtx.withinContext ?art designHoleFeatureOntologyA))
    (exists (?func1 ?func2)
        (and (Design_Function ?func1)
             (Design_Function ?func2)
             (Foundation.holds_function ?f ?func1)
             (Foundation.holds_function ?art ?func2))))
```
:IC hard "**Every instance of feature and artifact in the Design Hole Feature Ontology A holds some design function.**"

```
(=> (Housing_Part_Family ?house)
    (exists (?flange ?boss ?bhole ?phole ?lphole)
        (and (External_Flange ?flange)
             (Boss ?boss)
             (Bolt_Hole ?bhole)
             (Plain_Diameter_Hole ?phole)
             (Locating_Pin_Hole ?lphole)
             (Foundation.holds_feature ?house ?flange)
             (Foundation.holds_feature ?house ?boss)
             (Foundation.holds_feature ?house ?bhole)
             (Foundation.holds_feature ?house ?phole)
             (Foundation.holds_feature ?house ?lphole))))
```
:IC hard "**Every housing has some compulsory external flange, boss, bolt hole, plain diameter hole and locating pin hole as features present on the housing.**"

```
(=> (Housing_Part_Family ?house)
    (exists (?al)
        (and (Aluminium ?al)
             (Foundation.holds_material ?house ?al))))
```
:IC hard "**Every housing is made up of some aluminium material.**"

```
(=> (Bolt_Hole ?bhole)
    (Foundation.compound ?bhole))
```
:IC hard "**A bolt hole is a compound feature.**"

```
(=> (Bolt_Hole ?bhole)
    (exists (?phole ?shole)
        (and (Plain_Diameter_Hole ?phole)
            (Secondary_Hole ?shole)
            (Foundation.element_of ?phole ?bhole)
            (Foundation.element_of ?shole ?bhole))))
```
:IC hard "**Every bolt hole involves a plain diameter hole and a secondary hole which are elements of the bolt hole.**"

```
(=> (and (Bolt_Hole ?bhole)
        (Plain_Diameter_Hole ?phole)
        (Foundation.element_of ?phole ?bhole))
    (Foundation.base ?phole))
```
:IC hard "**The plain diameter hole of a bolt hole is the base feature of the bolt hole.**"

```
(=> (and (Bolt_Hole ?bhole)
        (Plain_Diameter_Hole ?phole)
        (Secondary_Hole ?shole)
        (Foundation.element_of ?phole ?bhole)
        (Foundation.element_of ?shole ?bhole)
        (Foundation.Circular_Closed_Profile ?ccp1)
        (Foundation.Circular_Closed_Profile ?ccp2)
        (Foundation.holds_shape ?phole ?ccp1)
        (Foundation.holds_shape ?shole ?ccp2))
    (exists (?real1 ?real2)
        (and (RootCtx.RealNumber ?real1)
            (RootCtx.RealNumber ?real2)
            (Foundation.measures ?ccp1 (Foundation.mm ?real1))
            (Foundation.measures ?ccp2 (Foundation.mm ?real2))
            (/= ?real1 ?real2)
            (gtNum ?real2 ?real1))))
```
:IC hard "**The secondary hole element of a bolt hole has a diameter value which is always greater than that of the plain diameter hole element of the same bolt hole.**"

```
(=> (Boss ?boss)
    (Foundation.compound ?boss))
```
:IC hard "**A boss is a compound feature.**"

```
(=> (Boss ?boss)
    (exists (?c ?rhole)
        (and (Foundation.Cylinder ?c)
            (Foundation.Round_Hole ?rhole)
            (Foundation.element_of ?c ?boss)
            (Foundation.element_of ?rhole ?boss))))
```
:IC hard "**Every boss involves a cylinder and a round hole which are elements of the boss.**"

```
(=> (and (Boss ?boss)
        (Foundation.Cylinder ?c)
        (Foundation.element_of ?c ?boss))
    (Foundation.base ?c))
```
:IC hard "**The cylinder element of a boss is the base feature of the boss.**"

```
(=> (Boss ?boss)
    (exists (?ccp1 ?ccp2 ?bdia1 ?bdia2)
        (and (Foundation.Circular_Closed_Profile ?ccp1)
            (Foundation.Circular_Closed_Profile ?ccp2)
            (Foundation.holds_shape ?boss ?ccp1)
            (Foundation.holds_shape ?boss ?ccp2)
            (Boss_Diameter ?bdia1)
```

```
            (Boss_Diameter ?bdia2)
            (Foundation.measures ?ccp1 ?bdia1)
            (Foundation.measures ?ccp2 ?bdia2)
            (= ?bdia1 ?bdia2))))
```
:IC hard "**Every boss holds exactly two circular closed profiles of identical boss diameter.**"

```
(=> (Boss ?boss)
    (exists (?lin ?bheight)
        (and (Foundation.Linear_Path ?lin)
            (Foundation.holds_shape ?boss ?lin)
            (Boss_Height ?bheight)
            (Foundation.measures ?lin ?bheight))))
```
:IC hard "**Every boss holds exactly one linear path of boss height.**"

```
(=> (External_Flange ?flange)
    (exists (?ccp1 ?ccp2 ?fdia1 ?fdia2)
        (and (Foundation.Circular_Closed_Profile ?ccp1)
            (Foundation.Circular_Closed_Profile ?ccp2)
            (Foundation.holds_shape ?flange ?ccp1)
            (Foundation.holds_shape ?flange ?ccp2)
            (Flange_Diameter ?fdia1)
            (Flange_Diameter ?fdia2)
            (Foundation.measures ?ccp1 ?fdia1)
            (Foundation.measures ?ccp2 ?fdia2)
            (= ?fdia1 ?fdia2))))
```
:IC hard "**Every external flange holds exactly two circular closed profiles of identical flange diameter.**"

```
(=> (External_Flange ?flange)
    (exists (?lin ?fdepth)
        (and (Foundation.Linear_Path ?lin)
            (Foundation.holds_shape ?flange ?lin)
            (Flange_Thickness ?fdepth)
            (Foundation.measures ?lin ?fdepth))))
```
:IC hard "**Every external flange holds exactly one linear path of flange thickness.**"

```
(=> (Locating_Pin_Hole ?lphole)
    (exists (?ccp1 ?ccp2 ?phdia1 ?phdia2)
        (and (Foundation.Circular_Closed_Profile ?ccp1)
            (Foundation.Circular_Closed_Profile ?ccp2)
            (Foundation.holds_shape ?lphole ?ccp1)
            (Foundation.holds_shape ?lphole ?ccp2)
            (Primary_Diameter ?phdia1)
            (Primary_Diameter ?phdia2)
            (Foundation.measures ?ccp1 ?phdia1)
            (Foundation.measures ?ccp2 ?phdia2)
            (= ?phdia1 ?phdia2))))
```
:IC hard "**Every locating pin hole holds exactly two circular closed profiles of identical primary diameter.**"

```
(=> (Locating_Pin_Hole ?lphole)
    (exists (?lin ?phdepth)
        (and (Foundation.Linear_Path ?lin)
            (Foundation.holds_shape ?lphole ?lin)
            (Primary_Depth ?phdepth)
            (Foundation.measures ?lin ?phdepth))))
```
:IC hard "**Every locating pin hole holds exactly one linear path of primary depth.**"

```
(=> (and (Locating_Pin_Hole ?lphole)
         (Foundation.Linear_Path ?lin)
         (Foundation.Circular_Closed_Profile ?ccp)
         (Foundation.through ?ccp)
         (Foundation.holds_shape ?lphole ?lin)
         (Foundation.holds_shape ?lphole ?ccp))
    (exists (?chf1 ?chf2)
        (and (Foundation.Chamfer ?chf1)
             (Foundation.Chamfer ?chf2)
             (Foundation.blends ?chf1 ?lin)
             (Foundation.blends ?chf2 ?lin))))
```
:IC hard "**Every locating pin hole that has a through hole bottom condition requires two chamfers that blend the linear path of the reamed hole.**"

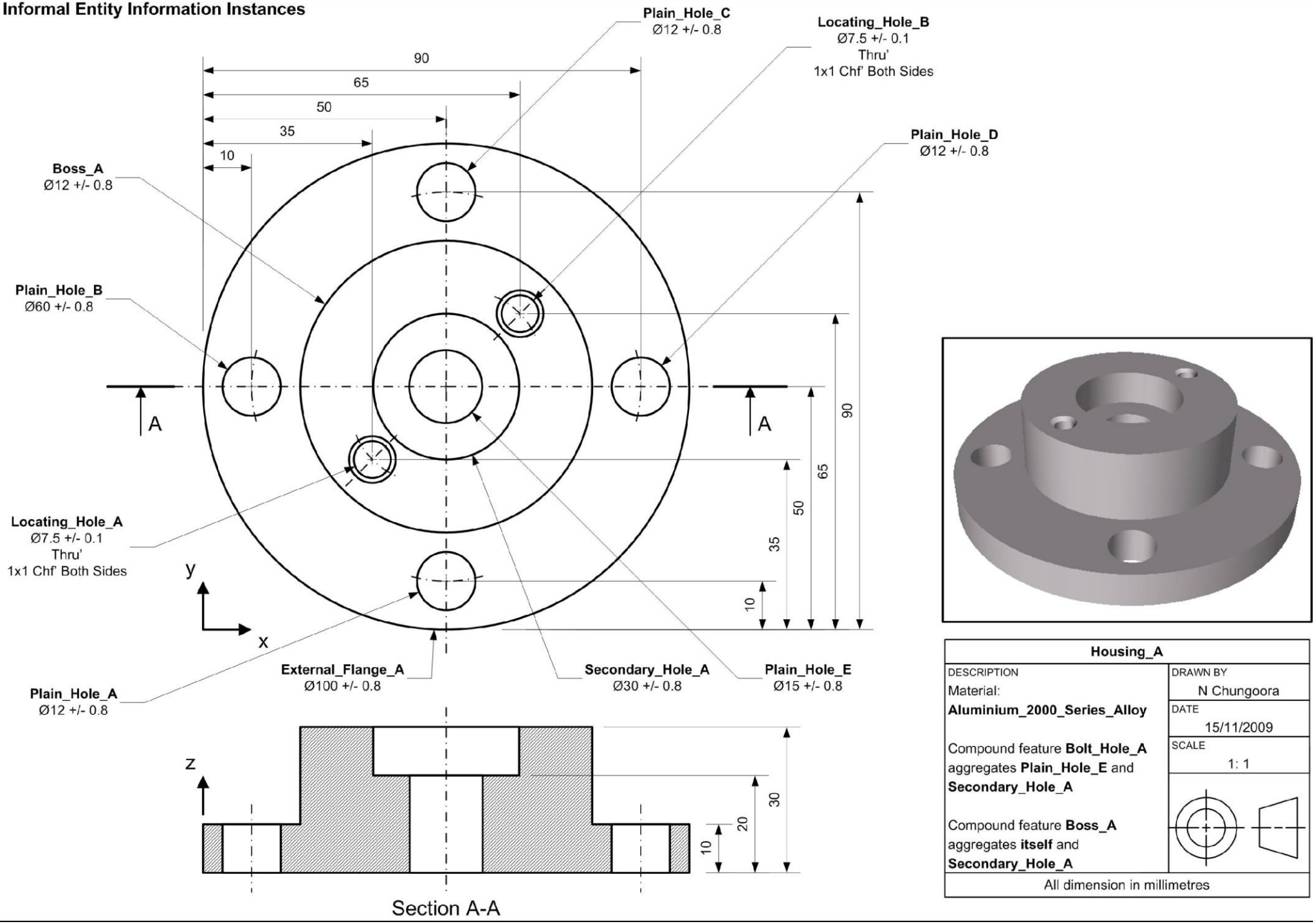
```
(=> (Plain_Diameter_Hole ?phole)
    (exists (?ccp1 ?ccp2 ?phdia1 ?phdia2)
        (and (Foundation.Circular_Closed_Profile ?ccp1)
             (Foundation.Circular_Closed_Profile ?ccp2)
             (Foundation.holds_shape ?phole ?ccp1)
             (Foundation.holds_shape ?phole ?ccp2)
             (Primary_Diameter ?phdia1)
             (Primary_Diameter ?phdia2)
             (Foundation.measures ?ccp1 ?phdia1)
             (Foundation.measures ?ccp2 ?phdia2)
             (= ?phdia1 ?phdia2))))
```
:IC hard "**Every plain diameter hole holds exactly two circular closed profiles of identical primary diameter.**"

```
(=> (Plain_Diameter_Hole ?phole)
    (exists (?lin ?phdepth)
        (and (Foundation.Linear_Path ?lin)
             (Foundation.holds_shape ?phole ?lin)
             (Primary_Depth ?phdepth)
             (Foundation.measures ?lin ?phdepth))))
```
:IC hard "**Every plain diameter hole holds exactly one linear path of primary depth.**"

```
(=> (Secondary_Hole ?shole)
    (exists (?ccp1 ?ccp2 ?sdia1 ?sdia2)
        (and (Foundation.Circular_Closed_Profile ?ccp1)
             (Foundation.Circular_Closed_Profile ?ccp2)
             (Foundation.holds_shape ?shole ?ccp1)
             (Foundation.holds_shape ?shole ?ccp2)
             (Secondary_Diameter ?sdia1)
             (Secondary_Diameter ?sdia2)
             (Foundation.measures ?ccp1 ?sdia1)
             (Foundation.measures ?ccp2 ?sdia2)
             (= ?sdia1 ?sdia2))))
```
:IC hard "**Every secondary hole holds exactly two circular closed profiles of identical secondary diameter.**"

```
(=> (Secondary_Hole ?shole)
    (exists (?lin ?sdepth)
        (and (Foundation.Linear_Path ?lin)
             (Foundation.holds_shape ?shole ?lin)
             (Secondary_Depth ?sdepth)
             (Foundation.measures ?lin ?sdepth))))
```
:IC hard "**Every secondary hole holds exactly one linear path of secondary depth.**"

Informal Entity Information Instances



Housing_A	
DESCRIPTION Material: **Aluminium_2000_Series_Alloy** Compound feature **Bolt_Hole_A** aggregates **Plain_Hole_E** and **Secondary_Hole_A** Compound feature **Boss_A** aggregates **itself** and **Secondary_Hole_A**	DRAWN BY N Chungoora DATE 15/11/2009 SCALE 1: 1
All dimension in millimetres	

D.3 Machining Hole Feature Ontology B

Context Declaration

:Ctx machiningHoleFeatureOntologyB
:Inst UserContext
:supCtx TopUserCtx
:name "Context for the Machining Hole Feature Ontology B"
:rem "This context explores a domain ontology developed for hole features defined from a machining process viewpoint using the semantics from the Foundation Layer."

:Use machiningHoleFeatureOntologyB

Classes

:Prop **Crank_Pulley_Part_Family**
:Inst Property
:sup Foundation.**Artifact**
:name "Crank_Pulley_Part_Family"
:rem "A crank pulley part family is a type of artifact which is is forged and then machined using turning and boring operations."

:Prop **Bored_Hole**
:Inst Property
:sup Foundation.**Round_Hole**
:name "Bored_Hole"
:rem "A bored hole is a round hole feature which is machined using a sequence of rough and finish boring operations."

:Prop **Large_Bored_Hole**
:Inst Property
:sup Foundation.**Round_Hole**
:name "Bored_Hole"
:rem "A large bored hole is a round hole feature which is machined using a sequence of rough and finish boring operations."

:Prop **Pulley_Core_Feature**
:Inst Property
:sup Foundation.**Feature**
:name "Pulley_Core_Feature"
:rem "A pulley core feature is a compound feature which defines the central portion of a crank pulley and is produced using turning and boring operations."

:Prop **Pulley_End_Feature**
:Inst Property
:sup Foundation.**Cylinder**
:name "Pulley_End_Feature"
:rem "A pulley end feature is a cylindrical feature which defines a portion of a crank pulley and is machined using turning operations."

:Prop **Bore_Hole_Making**
:Inst Property
:sup Foundation.**Activity**
:name "Bore_Hole_Making"
:rem "A bore hole making activity is a machining operation whose occurrences may produce both bored and large bored holes as outputs. An occurrence of a bored hole making activity,

which outputs a bored or large bored hole, consists of linear ordering semantics over rough boring followed by finish boring."

:Prop **Face_Turning**
:Inst Property
:sup Foundation.**Activity**
:name "Face_Turning"
:rem "A face turning activity is a machining operation whose occurrences produce faces of on a crank pulley."

:Prop **Finish_Boring**
:Inst Property
:sup Foundation.**Activity**
:name "Finish_Boring"
:rem "A finish boring activity is a machining operation whose occurrences produce bored and larged bored holes."

:Prop **Rough_Boring**
:Inst Property
:sup Foundation.**Activity**
:name "Rough_Boring"
:rem "A rough boring activity is a machining operation whose occurrences always need to precede occurrences of finish boring machining operations."

:Prop **Mild_Steel**
:Inst Property
:sup Foundation.Material
:name "Mild_Steel"
:rem "Mild steel is an alloy that contains between 0.16-0.29% carbon."

Functions

:Fun **inches**
:Inst UnaryFun
:Sig **RealNumber** -> Foundation.**Length_Measure**
:name "inches"
:rem "(= ?length (inches ?real)) is used to denote the value of a length measure in inches."

Axioms

```
(=> (Crank_Pulley_Part_Family ?pull)
    (exists (?core ?end1 ?end2)
        (and (Pulley_Core_Feature ?core)
            (Pulley_End_Feature ?end1)
            (Pulley_End_Feature ?end2)
            (Foundation.holds_feature ?pull ?core)
            (Foundation.holds_feature ?pull ?end1)
            (Foundation.holds_feature ?pull ?end2)
            (/= ?end1 ?end2))))
```

:IC hard "**Every crank pulley consists of some pulley core feature and two distinct pulley end features.**"

```
(=> (Crank_Pulley_Part_Family ?pull)
    (exists (?steel)
        (and (Mild_Steel ?steel)
            (Foundation.holds_material ?pull ?steel))))
```

:IC hard "**Every crank pulley is made up of some mild steel material.**"

```
(=> (Pulley_Core_Feature ?core)
    (Foundation.compound ?core))
```
:IC hard "**A pulley core feature is a compound feature.**"

```
(=> (Pulley_Core_Feature ?core)
    (exists (?lhole1 ?lhole2 ?bhole ?c)
        (and (Large_Bored_Hole ?lhole1)
             (Large_Bored_Hole ?lhole2)
             (/= ?lhole1 ?lhole2)
             (Bored_Hole ?bhole)
             (Foundation.Cylinder ?c)
             (Foundation.element_of ?lhole1 ?core)
             (Foundation.element_of ?lhole2 ?core)
             (Foundation.element_of ?bhole ?core)
             (Foundation.element_of ?c ?core))))
```
:IC hard "**Every pulley core feature consists of two distinct large bored holes, a minimum of one bored hole, and a cylinder which are elements of the pulley core feature.**"

```
(=> (and (Pulley_Core_Feature ?core)
         (Foundation.Cylinder ?c)
         (Foundation.element_of ?c ?core))
    (Foundation.base ?c))
```
:IC hard "**The cylinder element of a pulley core feature is the base feature of the pulley core feature.**"

```
(=> (and (Bored_Hole ?bhole)
         (Foundation.flow_object ?bhole))
    (exists (?fbore ?fboreOcc)
        (and (Finish_Boring ?fbore)
             (Foundation.Activity_Occurrence ?fboreOcc)
             (Foundation.occurrence_of ?fboreOcc ?fbore)
             (Foundation.output ?bhole ?fboreOcc))))
```
:IC hard "**Every bored hole that is a flow object is an output from a finish boring activity occurrence.**"

```
(=> (and (Large_Bored_Hole ?lbhole)
         (Foundation.flow_object ?lbhole))
    (exists (?fbore ?fboreOcc)
        (and (Finish_Boring ?fbore)
             (Foundation.Activity_Occurrence ?fboreOcc)
             (Foundation.occurrence_of ?fboreOcc ?fbore)
             (Foundation.output ?lbhole ?fboreOcc))))
```
:IC hard "**Every large bored hole that is a flow object is an output from a finish boring activity occurrence.**"

```
(=> (and (Foundation.Circular_Closed_Profile ?ccp)
         (Foundation.flow_object ?ccp)
         (RootCtx.withinContext ?ccp machiningHoleFeatureOntologyB))
    (exists (?fturn ?fturnOcc)
        (and (Face_Turning ?fturn)
             (Foundation.Activity_Occurrence ?fturnOcc)
             (Foundation.occurrence_of ?fturnOcc ?fturn)
             (Foundation.output ?ccp ?fturnOcc))))
```
:IC hard "**Every circular closed profile in machining hole feature ontology B that is a flow object is an output from a face turning activity occurrence.**"

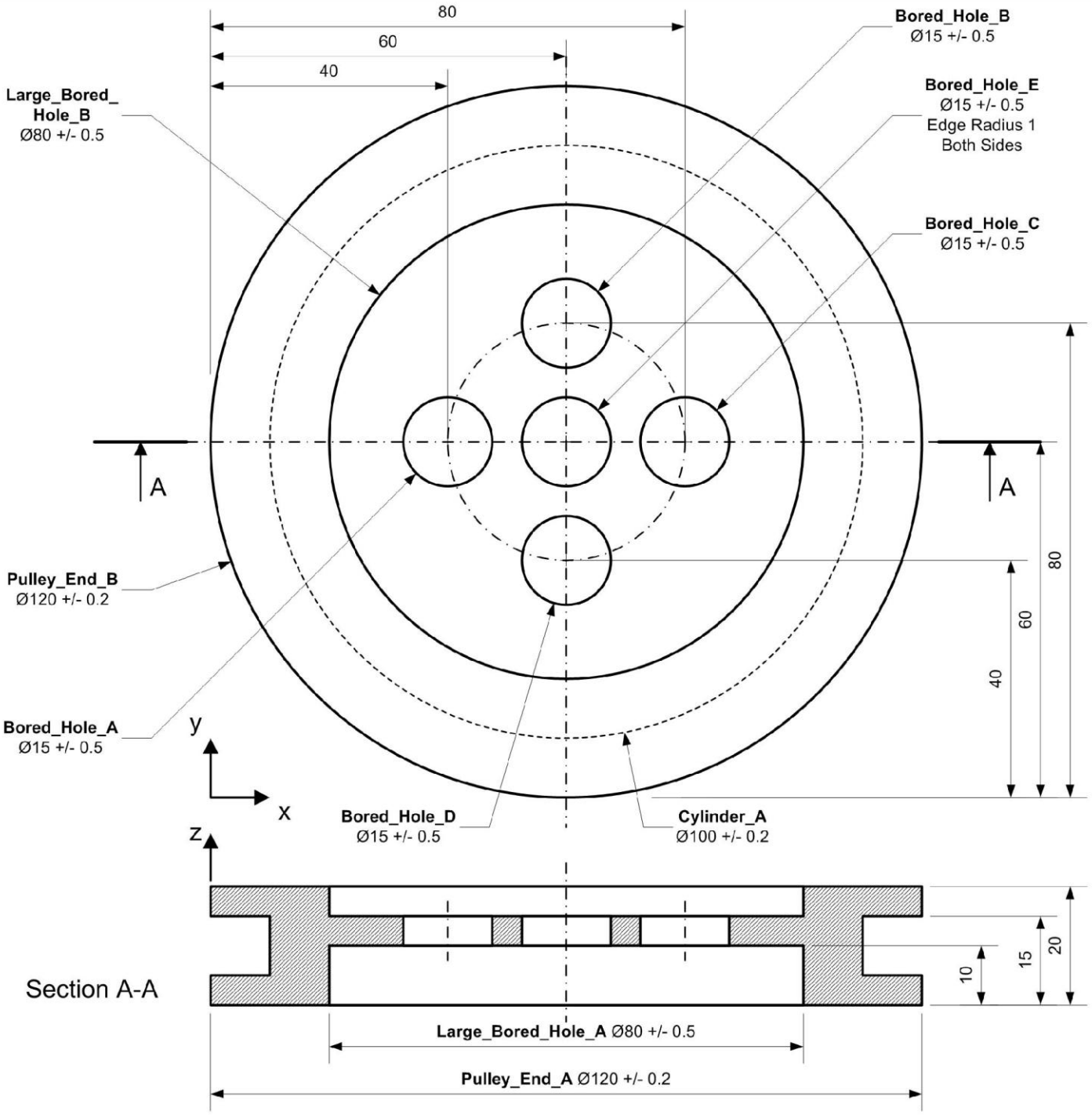
```
(=> (and (Finish_Boring ?fbore)
```

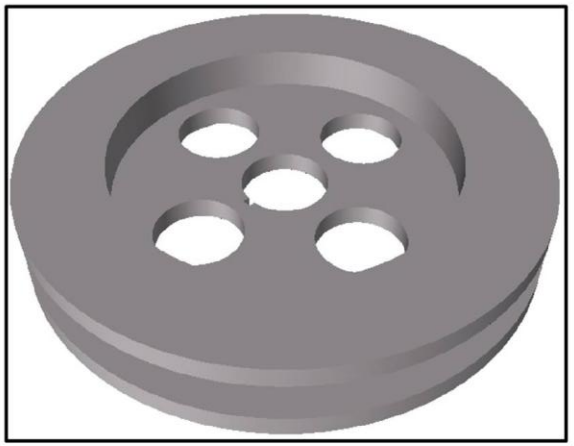
```
        (Foundation.Activity_Occurrence ?fboreOcc)
        (Foundation.occurrence_of ?fboreOcc ?fbore)
        (Foundation.legal ?fboreOcc))
    (exists (?rbore ?rboreOcc)
        (and (Rough_Boring ?rbore)
            (Foundation.Activity_Occurrence ?rboreOcc)
            (Foundation.occurrence_of ?rboreOcc ?rbore)
            (Foundation.legal ?rboreOcc)
            (Foundation.earlier ?rboreOcc ?fboreOcc))))
```
:IC hard "**Every legal finish boring activity occurrence implies the existence of some rough boring activity occurrence that is earlier than the finish boring activity occurrence in the occurrence tree.**"

```
(=> (and (Bore_Hole_Making ?bholeMake)
        (Foundation.Activity_Occurrence ?bholeMakeOcc)
        (Foundation.occurrence_of ?bholeMakeOcc ?bholeMake))
    (exists (?rbore ?rboreOcc ?fbore ?fboreOcc)
        (and (Rough_Boring ?rbore)
            (Finish_Boring ?fbore)
            (Foundation.Activity_Occurrence ?rboreOcc)
            (Foundation.Activity_Occurrence ?fboreOcc)
            (Foundation.occurrence_of ?rboreOcc ?rbore)
            (Foundation.occurrence_of ?fboreOcc ?fbore)
            (Foundation.min_precedes ?rboreOcc ?fboreOcc ?bholeMake))))
```
:IC hard "**An occurrence of rough boring must always precede an occurrence of finish boring under a complex occurrence of bore hole making. Other behaviours under the complex bore hole making activity may occur in between.**"

```
(=> (and (Bore_Hole_Making ?bholeMake)
        (Foundation.Activity_Occurrence ?bholeMakeOcc)
        (Foundation.occurrence_of ?bholeMakeOcc ?bholeMake))
    (exists (?rbore ?rboreOcc)
        (and (Rough_Boring ?rbore)
            (Foundation.subactivity ?rbore ?bholeMake)
            (Foundation.Activity_Occurrence ?rboreOcc)
            (Foundation.occurrence_of ?rboreOcc ?rbore)
            (Foundation.subactivity_occurrence ?rboreOcc ?bholeMakeOcc)
            (Foundation.root_occ ?rboreOcc ?bholeMakeOcc))))
```
:IC hard "**An occurrence of rough boring at the root of the process sequence under a complex occurrence of bore hole making is a precondition to the complex occurrence.**"

```
(=> (and (Bore_Hole_Making ?bholeMake)
        (Foundation.Activity_Occurrence ?bholeMakeOcc)
        (Foundation.occurrence_of ?bholeMakeOcc ?bholeMake))
    (exists (?fbore ?fboreOcc)
        (and (Finish_Boring ?fbore)
            (Foundation.subactivity ?fbore ?bholeMake)
            (Foundation.Activity_Occurrence ?fboreOcc)
            (Foundation.occurrence_of ?fboreOcc ?fbore)
            (Foundation.subactivity_occurrence ?fboreOcc ?bholeMakeOcc)
            (Foundation.leaf_occ ?fboreOcc ?bholeMakeOcc))))
```
:IC hard "**An occurrence of finish boring at the leaf of the process sequence under a complex occurrence of bore hole making is a post-condition to the complex occurrence.**"





Crank_Pulley_Series_01	
DESCRIPTION Material: **Mild_Steel_0.2%_Carbon** Compound feature **Pulley_Core_A** aggregates **Bored_Hole_A**, **Bored_Hole_B**, **Bored_Hole_C**, **Bored_Hole_D**, **Bored_Hole_E**, **Large_Bored_Hole_A**, **Large_Bored_Hole_B** and **Cylinder_A**	DRAWN BY N Chungoora DATE 15/11/2009 SCALE 1: 1
All dimensions in millimetres	

D.4 ISO Tolerance Band Model

Context Declaration

:Ctx isoToleranceBand
:Inst UserContext
:supCtx TopUserCtx
:name "ISO Tolerance Band Domain Context"
:rem "This context may be used to establish potential hole machining processes to produce round holes of known nominal entry diameter and diameter tolerances. This context can also be used to match the conformance of domain-defined hole making activity occurrences with respect to the hole machining processes identified under the ISO Tolerance Band Model."

:Use isoToleranceBand

Relations

:Rel **toleranceBandRelation_01**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_01"
:rem "Based on the entry diameter and entry diameter size tolerance of the queried Foundation.Round_Hole instance, it can be inferred that the feature can be produced using a Honing machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_02**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_02"
:rem "Based on the entry diameter and entry diameter size tolerance of the queried Foundation.Round_Hole instance, it can be inferred that the feature can be produced using an Internal Grinding machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_03**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_03"
:rem "Based on the entry diameter and entry diameter size tolerance of the queried Foundation.Round_Hole instance, it can be inferred that the feature can be produced using an Internal Broaching machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_04**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_04"
:rem "Based on the entry diameter and entry diameter size tolerance of the queried Foundation.Round_Hole instance, it can be inferred that the feature can be produced using a Reaming machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_05**
:Inst UnaryRel

:Sig Property
:name "toleranceBandRelation_05"
:rem "Based on the entry diameter and entry diameter size tolerance of the queried Foundation.Round_Hole instance, it can be inferred that the feature can be produced using a Boring machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_06**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_06"
:rem "Based on the entry diameter and entry diameter size tolerance of the queried Foundation.Round_Hole instance, it can be inferred that the feature can be produced using a Drilling machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_07**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_07"
:rem "The queried Activity_Occurrence instance, which is a hole making occurrence, matches the tolerance range capability of a Honing machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_08**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_08"
:rem "The queried Activity_Occurrence instance, which is a hole making occurrence, matches the tolerance range capability of a Internal Grinding machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_09**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_09"
:rem "The queried Activity_Occurrence instance, which is a hole making occurrence, matches the tolerance range capability of a Internal Broaching machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_10**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_10"
:rem "The queried Activity_Occurrence instance, which is a hole making occurrence, matches the tolerance range capability of a Reaming machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_11**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_11"
:rem "The queried Activity_Occurrence instance, which is a hole making occurrence, matches the tolerance range capability of a Boring machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

:Rel **toleranceBandRelation_12**
:Inst UnaryRel
:Sig Property
:name "toleranceBandRelation_12"

:rem "The queried Activity_Occurrence instance, which is a hole making occurrence, matches the tolerance range capability of a Drilling machining process."
:limitationRem "This criteria is only satisfied under the ISO Tolerance Band domain model."

Definitions

```
(<= (toleranceBandRelation_01 ?hole)
    (and (Foundation.Round_Hole ?hole)
         (Foundation.Circular_Closed_Profile ?ccp)
         (Foundation.holds_shape ?hole ?ccp)
         (not (Foundation.through ?ccp))
         (not (Foundation.blind ?ccp))
         (Foundation.measures ?ccp (Foundation.mm ?real))
         (Foundation.holds_size_tolerance ?ccp (Foundation.tolerance_value
(Foundation.mm ?realmin) (Foundation.mm ?realmax)) (Foundation.mm ?real))
         (or (and (RootCtx.inInterval ?real (RootCtx.interval ex 1 3 in))
                  (RootCtx.inInterval ?realmin (RootCtx.interval in -0.04 -0.02 in))
                  (RootCtx.inInterval ?realmax (RootCtx.interval in 0.02 0.04 in)))
             (and (RootCtx.inInterval ?real (RootCtx.interval ex 3 6 in))
                  (RootCtx.inInterval ?realmin (RootCtx.interval in -0.05 -0.025 in))
                  (RootCtx.inInterval ?realmax (RootCtx.interval in 0.025 0.05 in)))
             (and (RootCtx.inInterval ?real (RootCtx.interval ex 6 10 in))
                  (RootCtx.inInterval ?realmin (RootCtx.interval in -0.06 -0.025 in))
                  (RootCtx.inInterval ?realmax (RootCtx.interval in 0.025 0.06 in)))
             (and (RootCtx.inInterval ?real (RootCtx.interval ex 10 18 in))
                  (RootCtx.inInterval ?realmin (RootCtx.interval in -0.08 -0.03 in))
                  (RootCtx.inInterval ?realmax (RootCtx.interval in 0.03 0.08 in)))
             (and (RootCtx.inInterval ?real (RootCtx.interval ex 18 30 in))
                  (RootCtx.inInterval ?realmin (RootCtx.interval in -0.09 -0.04 in))
                  (RootCtx.inInterval ?realmax (RootCtx.interval in 0.04 0.09 in)))
             (and (RootCtx.inInterval ?real (RootCtx.interval ex 30 50 in))
                  (RootCtx.inInterval ?realmin (RootCtx.interval in -0.11 -0.04 in))
                  (RootCtx.inInterval ?realmax (RootCtx.interval in 0.04 0.11 in))))))

(=> (toleranceBandRelation_01 ?hole)
    (RootCtx.holdsArg toleranceBandRelation_01 1 ?hole))

(<= (toleranceBandRelation_02 ?hole)
    (and (Foundation.Round_Hole ?hole)
         (Foundation.Circular_Closed_Profile ?ccp)
         (Foundation.holds_shape ?hole ?ccp)
         (not (Foundation.through ?ccp))
         (not (Foundation.blind ?ccp))
         (Foundation.measures ?ccp (Foundation.mm ?real))
         (Foundation.holds_size_tolerance ?ccp (Foundation.tolerance_value
(Foundation.mm ?realmin) (Foundation.mm ?realmax)) (Foundation.mm ?real))
         (or (and (RootCtx.inInterval ?real (RootCtx.interval ex 1 3 in))
                  (RootCtx.inInterval ?realmin (RootCtx.interval in -0.10 -0.03 in))
                  (RootCtx.inInterval ?realmax (RootCtx.interval in 0.03 0.10 in)))
             (and (RootCtx.inInterval ?real (RootCtx.interval ex 3 6 in))
                  (RootCtx.inInterval ?realmin (RootCtx.interval in -0.12 -0.04 in))
                  (RootCtx.inInterval ?realmax (RootCtx.interval in 0.04 0.12 in)))
             (and (RootCtx.inInterval ?real (RootCtx.interval ex 6 10 in))
                  (RootCtx.inInterval ?realmin (RootCtx.interval in -0.15 -0.04 in))
                  (RootCtx.inInterval ?realmax (RootCtx.interval in 0.04 0.15 in)))
             (and (RootCtx.inInterval ?real (RootCtx.interval ex 10 18 in))
                  (RootCtx.inInterval ?realmin (RootCtx.interval in -0.18 -0.05 in))
                  (RootCtx.inInterval ?realmax (RootCtx.interval in 0.05 0.18 in)))
```

```
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 18 30 in))
                 (RootCtx.inInterval ?realmin (RootCtx.interval in -0.21 -0.6 in))
                 (RootCtx.inInterval ?realmax (RootCtx.interval in 0.06 0.21 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 30 50 in))
                 (RootCtx.inInterval ?realmin (RootCtx.interval in -0.25 -0.07 in))
                 (RootCtx.inInterval ?realmax (RootCtx.interval in 0.07 0.25 in))))))

(=> (toleranceBandRelation_02 ?hole)
      (RootCtx.holdsArg toleranceBandRelation_02 1 ?hole))

 (<= (toleranceBandRelation_03 ?hole)
     (and (Foundation.Round_Hole ?hole)
          (Foundation.Circular_Closed_Profile ?ccp)
          (Foundation.holds_shape ?hole ?ccp)
          (not (Foundation.through ?ccp))
          (not (Foundation.blind ?ccp))
          (Foundation.measures ?ccp (Foundation.mm ?real))
          (Foundation.holds_size_tolerance ?ccp (Foundation.tolerance_value
(Foundation.mm ?realmin) (Foundation.mm ?realmax)) (Foundation.mm ?real))
          (or (and (RootCtx.inInterval ?real (RootCtx.interval ex 1 3 in))
                   (RootCtx.inInterval ?realmin (RootCtx.interval in -0.25 -0.04 in))
                   (RootCtx.inInterval ?realmax (RootCtx.interval in 0.04 0.25 in)))
              (and (RootCtx.inInterval ?real (RootCtx.interval ex 3 6 in))
                   (RootCtx.inInterval ?realmin (RootCtx.interval in -0.30 -0.05 in))
                   (RootCtx.inInterval ?realmax (RootCtx.interval in 0.05 0.30 in)))
              (and (RootCtx.inInterval ?real (RootCtx.interval ex 6 10 in))
                   (RootCtx.inInterval ?realmin (RootCtx.interval in -0.36 -0.06 in))
                   (RootCtx.inInterval ?realmax (RootCtx.interval in 0.06 0.36 in)))
              (and (RootCtx.inInterval ?real (RootCtx.interval ex 10 18 in))
                   (RootCtx.inInterval ?realmin (RootCtx.interval in -0.43 -0.08 in))
                   (RootCtx.inInterval ?realmax (RootCtx.interval in 0.08 0.43 in)))
              (and (RootCtx.inInterval ?real (RootCtx.interval ex 18 30 in))
                   (RootCtx.inInterval ?realmin (RootCtx.interval in -0.52 -0.09 in))
                   (RootCtx.inInterval ?realmax (RootCtx.interval in 0.09 0.52 in)))
              (and (RootCtx.inInterval ?real (RootCtx.interval ex 30 50 in))
                   (RootCtx.inInterval ?realmin (RootCtx.interval in -0.62 -0.11 in))
                   (RootCtx.inInterval ?realmax (RootCtx.interval in 0.11 0.62 in))))))

(=> (toleranceBandRelation_03 ?hole)
      (RootCtx.holdsArg toleranceBandRelation_03 1 ?hole))

 (<= (toleranceBandRelation_04 ?hole)
     (and (Foundation.Round_Hole ?hole)
          (Foundation.Circular_Closed_Profile ?ccp)
          (Foundation.holds_shape ?hole ?ccp)
          (not (Foundation.through ?ccp))
          (not (Foundation.blind ?ccp))
          (Foundation.measures ?ccp (Foundation.mm ?real))
          (Foundation.holds_size_tolerance ?ccp (Foundation.tolerance_value
(Foundation.mm ?realmin) (Foundation.mm ?realmax)) (Foundation.mm ?real))
          (or (and (RootCtx.inInterval ?real (RootCtx.interval ex 1 3 in))
                   (RootCtx.inInterval ?realmin (RootCtx.interval in -0.25 -0.04 in))
                   (RootCtx.inInterval ?realmax (RootCtx.interval in 0.04 0.25 in)))
              (and (RootCtx.inInterval ?real (RootCtx.interval ex 3 6 in))
                   (RootCtx.inInterval ?realmin (RootCtx.interval in -0.30 -0.05 in))
                   (RootCtx.inInterval ?realmax (RootCtx.interval in 0.05 0.30 in)))
              (and (RootCtx.inInterval ?real (RootCtx.interval ex 6 10 in))
                   (RootCtx.inInterval ?realmin (RootCtx.interval in -0.36 -0.06 in))
                   (RootCtx.inInterval ?realmax (RootCtx.interval in 0.06 0.36 in)))
              (and (RootCtx.inInterval ?real (RootCtx.interval ex 10 18 in))
```

```
                (RootCtx.inInterval ?realmin (RootCtx.interval in -0.43 -0.08 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.08 0.43 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 18 30 in))
                (RootCtx.inInterval ?realmin (RootCtx.interval in -0.52 -0.09 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.09 0.52 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 30 50 in))
                (RootCtx.inInterval ?realmin (RootCtx.interval in -0.62 -0.11 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.11 0.62 in))))))

(=> (toleranceBandRelation_04 ?hole)
        (RootCtx.holdsArg toleranceBandRelation_04 1 ?hole))

 (<= (toleranceBandRelation_05 ?hole)
     (and (Foundation.Round_Hole ?hole)
          (Foundation.Circular_Closed_Profile ?ccp)
          (Foundation.holds_shape ?hole ?ccp)
          (not (Foundation.through ?ccp))
          (not (Foundation.blind ?ccp))
          (Foundation.measures ?ccp (Foundation.mm ?real))
          (Foundation.holds_size_tolerance ?ccp (Foundation.tolerance_value
(Foundation.mm ?realmin) (Foundation.mm ?realmax)) (Foundation.mm ?real))
          (or (and (RootCtx.inInterval ?real (RootCtx.interval ex 1 3 in))
                (RootCtx.inInterval ?realmin (RootCtx.interval in -1.00 -0.06 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.06 1.00 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 3 6 in))
                (RootCtx.inInterval ?realmin (RootCtx.interval in -1.20 -0.08 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.08 1.20 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 6 10 in))
                (RootCtx.inInterval ?realmin (RootCtx.interval in -1.50 -0.09 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.09 1.50 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 10 18 in))
                (RootCtx.inInterval ?realmin (RootCtx.interval in -1.80 -0.11 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.11 1.80 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 18 30 in))
                (RootCtx.inInterval ?realmin (RootCtx.interval in -2.10 -0.13 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.13 2.10 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 30 50 in))
                (RootCtx.inInterval ?realmin (RootCtx.interval in -2.50 -0.16 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.16 2.50 in))))))

(=> (toleranceBandRelation_05 ?hole)
        (RootCtx.holdsArg toleranceBandRelation_05 1 ?hole))



 (<= (toleranceBandRelation_06 ?hole)
     (and (Foundation.Round_Hole ?hole)
          (Foundation.Circular_Closed_Profile ?ccp)
          (Foundation.holds_shape ?hole ?ccp)
          (not (Foundation.through ?ccp))
          (not (Foundation.blind ?ccp))
          (Foundation.measures ?ccp (Foundation.mm ?real))
          (Foundation.holds_size_tolerance ?ccp (Foundation.tolerance_value
(Foundation.mm ?realmin) (Foundation.mm ?realmax)) (Foundation.mm ?real))
          (or (and (RootCtx.inInterval ?real (RootCtx.interval ex 1 3 in))
                (RootCtx.inInterval ?realmin (RootCtx.interval in -1.00 -0.60 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.60 1.00 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 3 6 in))
                (RootCtx.inInterval ?realmin (RootCtx.interval in -1.20 -0.75 in))
                (RootCtx.inInterval ?realmax (RootCtx.interval in 0.75 1.20 in)))
```

```
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 6 10 in))
                 (RootCtx.inInterval ?realmin (RootCtx.interval in -1.50 -0.90 in))
                 (RootCtx.inInterval ?realmax (RootCtx.interval in 0.90 1.50 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 10 18 in))
                 (RootCtx.inInterval ?realmin (RootCtx.interval in -1.80 -1.10 in))
                 (RootCtx.inInterval ?realmax (RootCtx.interval in 1.10 1.80 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 18 30 in))
                 (RootCtx.inInterval ?realmin (RootCtx.interval in -2.10 -1.30 in))
                 (RootCtx.inInterval ?realmax (RootCtx.interval in 1.30 2.10 in)))
            (and (RootCtx.inInterval ?real (RootCtx.interval ex 30 50 in))
                 (RootCtx.inInterval ?realmin (RootCtx.interval in -2.50 -1.60 in))
                 (RootCtx.inInterval ?realmax (RootCtx.interval in 1.60 2.50 in))))))


(=> (toleranceBandRelation_06 ?hole)
        (RootCtx.holdsArg toleranceBandRelation_06 1 ?hole))

 (<= (toleranceBandRelation_07 ?occ)
     (and (RootCtx.inst ?occ Foundation.Activity_Occurrence)
          (RootCtx.inst ?hole Foundation.Round_Hole)
          (Foundation.output ?hole ?occ)
          (toleranceBandRelation_01 ?hole)))

(=> (toleranceBandRelation_07 ?occ)
        (RootCtx.holdsArg toleranceBandRelation_07 1 ?occ))

 (<= (toleranceBandRelation_08 ?occ)
     (and (RootCtx.inst ?occ Foundation.Activity_Occurrence)
          (RootCtx.inst ?hole Foundation.Round_Hole)
          (Foundation.output ?hole ?occ)
          (toleranceBandRelation_02 ?hole)))

(=> (toleranceBandRelation_08 ?occ)
        (RootCtx.holdsArg toleranceBandRelation_08 1 ?occ))

 (<= (toleranceBandRelation_09 ?occ)
     (and (RootCtx.inst ?occ Foundation.Activity_Occurrence)
          (RootCtx.inst ?hole Foundation.Round_Hole)
          (Foundation.output ?hole ?occ)
          (toleranceBandRelation_03 ?hole)))


(=> (toleranceBandRelation_09 ?occ)
        (RootCtx.holdsArg toleranceBandRelation_09 1 ?occ))

 (<= (toleranceBandRelation_10 ?occ)
     (and (RootCtx.inst ?occ Foundation.Activity_Occurrence)
          (RootCtx.inst ?hole Foundation.Round_Hole)
          (Foundation.output ?hole ?occ)
          (toleranceBandRelation_04 ?hole)))

(=> (toleranceBandRelation_10 ?occ)
        (RootCtx.holdsArg toleranceBandRelation_10 1 ?occ))

 (<= (toleranceBandRelation_11 ?occ)
     (and (RootCtx.inst ?occ Foundation.Activity_Occurrence)
          (RootCtx.inst ?hole Foundation.Round_Hole)
          (Foundation.output ?hole ?occ)
          (toleranceBandRelation_05 ?hole)))
```

```
(=> (toleranceBandRelation_11 ?occ)
        (RootCtx.holdsArg toleranceBandRelation_11 1 ?occ))

 (<= (toleranceBandRelation_12 ?occ)
     (and (RootCtx.inst ?occ Foundation.Activity_Occurrence)
          (RootCtx.inst ?hole Foundation.Round_Hole)
          (Foundation.output ?hole ?occ)
          (toleranceBandRelation_06 ?hole)))

(=> (toleranceBandRelation_12 ?occ)
        (RootCtx.holdsArg toleranceBandRelation_12 1 ?occ))
```

E Semantic Reconciliation Layer

E.1 Semantic Mapping Concepts Based on Foundation Semantics

A large number of semantic mapping concepts based on foundation semantics has been explored in this work. As a consequence of this, only the ones that appear and contribute to the definition of semantic mapping concepts evaluated in Test Case 2 have been exposed in this section.

Context Declaration

```
:Ctx foundationMapping
:Inst UserContext
:supCtx TopUserCtx
:name "Foundation Mapping Context"
:rem "This context is used to define relevant semantic mapping concepts for use in the Semantic Reconciliation Layer purely based on foundation semantics."

:Use foundationMapping
```

Reconciliation of Classes

```
:Rel classMappingRelation_018
:Inst BinaryRel
:Sig Property Property
:name "classMappingRelation_018"
:rem "There exists a correspondence between the class ?x in the DomainX context and the class ?y in the DomainY context as a result of both ?x and ?y being subclasses of the foundation class Foundation.Round_Hole. Both ?x and ?y capture the notion of a feature that is of cylindrical or conical negative (removal) volume. It is necessary for instances of ?x and ?y be defined in terms of a first instance of Foundation.Circular_Closed_Profile swept along an instance of Foundation.Linear_Path resulting in a second instance of Foundation.Circular_Closed_Profile of identical or different dimensions. Every instance of ?x and ?y may hold a Foundation.Linear_Profile axis."
:limitationRem "Without reference to the terms assigned to the concepts ?x and ?y, there could potentially be class mismatches present. This is because ?x and ?y could have been defined with a view on specific domain preferences, which vary across domains. Varying levels of abstraction of the foundation class Foundation.Round_Hole in both domains could also result in class mismatches."

(<= (classMappingRelation_018 ?x ?y)
    (and (RootCtx.sup ?x Foundation.Round_Hole)
         (RootCtx.withinContext ?x DomainX)
         (RootCtx.sup ?y Foundation.Round_Hole)
         (RootCtx.withinContext ?y DomainY)))

(=> (classMappingRelation_018 ?x ?y)
    (and (RootCtx.holdsArg classMappingRelation_018 1 ?x)
```

(RootCtx.holdsArg **classMappingRelation_018** 2 ?y)))

:Rel **classMappingRelation_022**
:Inst BinaryRel
:Sig Property Property
:name "classMappingRelation_022"
:rem "There exists a correspondence between the class ?x in the DomainX context and the class ?y in the DomainY context as a result of both ?x and ?y being subclasses of the foundation class Foundation.Activity. Both ?x and ?y capture the notion of types of reusable process behaviours. Instances of ?x and ?y may have multiple occurrences present as instances of Foundation.Activity_Occurrence or present as instances of the subclasses of the latter defined in DomainX and DomainY respectively."
:limitationRem "Without reference to the terms assigned to the concepts ?x and ?y, there could potentially be class mismatches present. This is because ?x and ?y could have been defined with a view on specific domain preferences, which vary across domains. Varying levels of abstraction of the foundation class Foundation.Activity in both domains could also result in class mismatches."

(<= (**classMappingRelation_022** ?x ?y)
    (and (RootCtx.sup ?x Foundation.Activity)
        (RootCtx.withinContext ?x DomainX)
        (RootCtx.sup ?y Foundation.Activity)
        (RootCtx.withinContext ?y DomainY)))

(=> (**classMappingRelation_022** ?x ?y)
    (and (RootCtx.holdsArg **classMappingRelation_022** 1 ?x)
        (RootCtx.holdsArg **classMappingRelation_022** 2 ?y)))

Reconciliation of Functions

:Rel **pointerRelation_003**
:Inst UnaryRel
:Sig UnaryFun
:name "pointerRelation_003"

(<= (**pointerRelation_003** ?funx)
    (and (RootCtx.inst ?funx RootCtx.UnaryFun)
        (RootCtx.withinContext ?funx DomainX)
        (RootCtx.argProp ?funx 1 RootCtx.RealNumber)
        (RootCtx.returnProp ?funx Foundation.Length_Measure)))

:Rel **pointerRelation_004**
:Inst UnaryRel
:Sig UnaryFun
:name "pointerRelation_004"

(<= (**pointerRelation_004** ?funy)
    (and (RootCtx.inst ?funy RootCtx.UnaryFun)
        (RootCtx.withinContext ?funy DomainY)
        (RootCtx.argProp ?funy 1 RootCtx.RealNumber)
        (RootCtx.returnProp ?funy Foundation.Length_Measure)))

:Rel **functionMappingRelation_003**
:Inst BinaryRel
:Sig UnaryFun UnaryFun
:name "functionMappingRelation_003"
:rem "There exists a correspondence between the ontological functions ?funx in the DomainX context and ?funy in the DomainY context as a result of both ?funx and ?funy being used to

denote instances of the foundation class Foundation.Length_Measure. Both ?funx and ?funy capture the intuition about units of measurement for qualifying lengths. It is a necessary condition that all instances of Foundation.Length_Measure in DomainX and DomainY be characterised by units of measurement with RootCtx.RealNumber values."
:limitationRem "Without reference to the terms assigned to the unit of measurement functions ?funx and ?funy, there could be a Concept and Term CT mismatch present. This occurs if different terms are used to refer to two fundamentally different unit functions."
:exampleRem "(m 10) v/s (inch 0.5) In this case, the ontological functions are m and inch which not only use different terms but are also conceptually different. However, the way in which they denote instances of Foundation.Length_Measure is the same."

(<= (**functionMappingRelation_003** ?funx ?funy)
    (and (**pointerRelation_003** ?funx)
         (**pointerRelation_004** ?funy)))

(=> (**functionMappingRelation_003** ?funx ?funy)
    (and (RootCtx.holdsArg **functionMappingRelation_003** 1 ?funx)
         (RootCtx.holdsArg **functionMappingRelation_003** 2 ?funy)))

Reconciliation of Instances

:Rel **pointerRelation_005**
:Inst BinaryRel
:Sig Property Property
:name "pointerRelation_005"

(<= (**pointerRelation_005** ?ccpx ?lengthx)
    (and (RootCtx.inst ?ccpx Foundation.Circular_Closed_Profile)
         (RootCtx.withinContext ?ccpx DomainX)
         (RootCtx.inst ?lengthx Foundation.Length_Measure)
         (Foundation.measures ?ccpx ?lengthx)))

:Rel **pointerRelation_006**
:Inst BinaryRel
:Sig Property Property
:name "pointerRelation_006"

(<= (**pointerRelation_006** ?ccpy ?lengthy)
    (and (RootCtx.inst ?ccpy Foundation.Circular_Closed_Profile)
         (RootCtx.withinContext ?ccpy DomainY)
         (RootCtx.inst ?lengthy Foundation.Length_Measure)
         (Foundation.measures ?ccpy ?lengthy)))

:Rel **pointerRelation_009**
:Inst BinaryRel
:Sig Property Property
:name "pointerRelation_009"

(<= (**pointerRelation_009** ?linx ?lengthx)
    (and (RootCtx.inst ?linx Foundation.Linear_Path)
         (RootCtx.withinContext ?linx DomainX)
         (RootCtx.inst ?lengthx Foundation.Length_Measure)
         (Foundation.measures ?linx ?lengthx)))

:Rel **pointerRelation_010**
:Inst BinaryRel
:Sig Property Property
:name "pointerRelation_010"

```
(<= (pointerRelation_010 ?liny ?lengthy)
    (and (RootCtx.inst ?liny Foundation.Linear_Path)
         (RootCtx.withinContext ?liny DomainY)
         (RootCtx.inst ?lengthy Foundation.Length_Measure)
         (Foundation.measures ?liny ?lengthy)))

:Rel pointerRelation_019
:Inst TernaryRel
:Sig Property Property Property
:name "pointerRelation_019"

(<= (pointerRelation_019 ?fx ?ptx ?vx)
    (and (RootCtx.inst ?fx Foundation.Feature)
         (RootCtx.withinContext ?fx DomainX)
         (RootCtx.inst ?px Foundation.Placement)
         (Foundation.holds_orientation ?fx ?px)
         (RootCtx.inst ?ptx Foundation.Point)
         (RootCtx.inst ?vx Foundation.Vector_Direction)
         (Foundation.is_oriented_at ?px ?ptx ?vx)))

:Rel pointerRelation_020
:Inst TernaryRel
:Sig Property Property Property
:name "pointerRelation_020"

(<= (pointerRelation_020 ?fy ?pty ?vy)
    (and (RootCtx.inst ?fy Foundation.Feature)
         (RootCtx.withinContext ?fy DomainY)
         (RootCtx.inst ?py Foundation.Placement)
         (Foundation.holds_orientation ?fy ?py)
         (RootCtx.inst ?pty Foundation.Point)
         (RootCtx.inst ?vy Foundation.Vector_Direction)
         (Foundation.is_oriented_at ?py ?pty ?vy)))

:Rel pointerRelation_027
:Inst BinaryRel
:Sig Property Property
:name "pointerRelation_027"

(<= (pointerRelation_027 ?holex ?edgex)
    (and (RootCtx.inst ?edgex Foundation.Constant_Radius_Edge_Round)
         (RootCtx.inst ?linx Foundation.Linear_Path)
         (RootCtx.inst ?holex Foundation.Round_Hole)
         (RootCtx.withinContext ?holex DomainX)
         (Foundation.blends ?edgex ?linx)
         (Foundation.holds_shape ?holex ?linx)))

:Rel pointerRelation_028
:Inst BinaryRel
:Sig Property Property
:name "pointerRelation_028"

(<= (pointerRelation_028 ?holey ?edgey)
    (and (RootCtx.inst ?edgey Foundation.Constant_Radius_Edge_Round)
         (RootCtx.inst ?liny Foundation.Linear_Path)
         (RootCtx.inst ?holey Foundation.Round_Hole)
         (RootCtx.withinContext ?holey DomainY)
         (Foundation.blends ?edgey ?liny)
         (Foundation.holds_shape ?holey ?liny)))
```

```
:Rel pointerRelation_029
:Inst BinaryRel
:Sig Property Property
:name "pointerRelation_029"

(<= (pointerRelation_029 ?holex ?chfx)
    (and (RootCtx.inst ?chfx Foundation.Chamfer)
         (RootCtx.inst ?linx Foundation.Linear_Path)
         (RootCtx.inst ?holex Foundation.Round_Hole)
         (RootCtx.withinContext ?holex DomainX)
         (Foundation.blends ?chfx ?linx)
         (Foundation.holds_shape ?holex ?linx)))

:Rel pointerRelation_030
:Inst BinaryRel
:Sig Property Property
:name "pointerRelation_030"

(<= (pointerRelation_030 ?holey ?chfy)
    (and (RootCtx.inst ?chfy Foundation.Chamfer)
         (RootCtx.inst ?liny Foundation.Linear_Path)
         (RootCtx.inst ?holey Foundation.Round_Hole)
         (RootCtx.withinContext ?holey DomainY)
         (Foundation.blends ?chfy ?liny)
         (Foundation.holds_shape ?holey ?liny)))

:Rel pointerRelation_035
:Inst UnaryRel
:Sig Property
:name "pointerRelation_035"

(<= (pointerRelation_035 ?occx)
    (and (RootCtx.inst ?occx Foundation.Activity_Occurrence)
         (RootCtx.withinContext ?occx DomainX)
         (RootCtx.inst ?holex Foundation.Round_Hole)
         (RootCtx.withinContext ?holex DomainX)
         (Foundation.output ?holex ?occx)))

:Rel pointerRelation_036
:Inst UnaryRel
:Sig Property
:name "pointerRelation_036"

(<= (pointerRelation_036 ?occy)
    (and (RootCtx.inst ?occy Foundation.Activity_Occurrence)
         (RootCtx.withinContext ?occy DomainY)
         (RootCtx.inst ?holey Foundation.Round_Hole)
         (RootCtx.withinContext ?holey DomainY)
         (Foundation.output ?holey ?occy)))

:Rel instanceMappingRelation_003
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_003"
:rem "There exists a correspondence between the instances ?ccpx and ?ccpy as a result of both being asserted instances of the foundation class Foundation.Circular_Close_Profile declared in DomainX and DomainY respectively. ?ccpx has a nominal diameter which is numerically smaller than that of ?ccpy."
```

(<= (**instanceMappingRelation_003** ?ccpx ?ccpy)
    (and (**pointerRelation_005** ?ccpx ?lengthx)
        (**pointerRelation_006** ?ccpy ?lengthy)
        (RootCtx.inst ?lengthx Foundation.Length_Measure)
        (RootCtx.inst ?lengthy Foundation.Length_Measure)
        (= ?lengthx (Foundation.mm ?realx))
        (= ?lengthy (Foundation.mm ?realy))
        (ltNum ?realx ?realy)))

(=> (**instanceMappingRelation_003** ?ccpx ?ccpy)
    (and (RootCtx.holdsArg **instanceMappingRelation_003** 1 ?ccpx)
        (RootCtx.holdsArg **instanceMappingRelation_003** 2 ?ccpy)))

:Rel **instanceMappingRelation_008**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_008"
:rem "There exists a correspondence between the instances ?linx and ?liny as a result of both being asserted instances of the foundation class Foundation.Linear_Path declared in DomainX and DomainY respectively. ?linx has a nominal sweeping distance which is numerically greater than that of ?liny."

(<= (**instanceMappingRelation_008** ?linx ?liny)
    (and (**pointerRelation_009** ?linx ?lengthx)
        (**pointerRelation_010** ?liny ?lengthy)
        (RootCtx.inst ?lengthx Foundation.Length_Measure)
        (RootCtx.inst ?lengthy Foundation.Length_Measure)
        (= ?lengthx (Foundation.mm ?realx))
        (= ?lengthy (Foundation.mm ?realy))
        (gtNum ?realx ?realy)))

(=> (**instanceMappingRelation_008** ?linx ?liny)
    (and (RootCtx.holdsArg **instanceMappingRelation_008** 1 ?linx)
        (RootCtx.holdsArg **instanceMappingRelation_008** 2 ?liny)))

:Rel **instanceMappingRelation_022**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_022"
:rem "There exists a correspondence between the instances ?fx and ?fy as a result of both being asserted instances of the foundation class Foundation.Feature declared in DomainX and DomainY respectively. ?fx has a placement orientation which is spatially identical to that of ?fy."

(<= (**instanceMappingRelation_022** ?fx ?fy)
    (and (**pointerRelation_019** ?fx ?ptx ?vx)
        (**pointerRelation_020** ?fy ?pty ?vy)
        (RootCtx.inst ?ptx Foundation.Point)
        (RootCtx.inst ?pty Foundation.Point)
        (= ?ptx (Foundation.coordinates (Foundation.mm ?realptx1) (Foundation.mm ?realptx2) (Foundation.mm ?realptx3)))
        (= ?pty (Foundation.coordinates (Foundation.mm ?realpty1) (Foundation.mm ?realpty2) (Foundation.mm ?realpty3)))
        (RootCtx.inst ?vx Foundation.Vector_Direction)
        (RootCtx.inst ?vy Foundation.Vector_Direction)
        (= ?vx (Foundation.direction ?realvx1 ?realvx2 ?realvx3))
        (= ?vy (Foundation.direction ?realvy1 ?realvy2 ?realvy3))
        (eqNum ?realptx1 ?realpty1)
        (eqNum ?realptx2 ?realpty2)
        (eqNum ?realptx3 ?realpty3)

```
        (eqNum ?realvx1 ?realvy1)
        (eqNum ?realvx2 ?realvy2)
        (eqNum ?realvx3 ?realvy3)))

(=> (instanceMappingRelation_022 ?fx ?fy)
    (and (RootCtx.holdsArg instanceMappingRelation_022 1 ?fx)
        (RootCtx.holdsArg instanceMappingRelation_022 2 ?fy)))

:Rel instanceMappingRelation_023
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_023"
:rem "There exists a correspondence between the instances ?fx and ?fy as a result of both being asserted instances of the foundation class Foundation.Feature declared in DomainX and DomainY respectively. ?fx has a placement orientation which is spatially different from that of ?fy."

(<= (instanceMappingRelation_023 ?fx ?fy)
    (and (RootCtx.inst ?fx Foundation.Feature)
        (RootCtx.withinContext ?fx DomainX)
        (RootCtx.inst ?fy Foundation.Feature)
        (RootCtx.withinContext ?fy DomainY)
        (not (instanceMappingRelation_022 ?fx ?fy))))

(=> (instanceMappingRelation_023 ?fx ?fy)
    (and (RootCtx.holdsArg instanceMappingRelation_023 1 ?fx)
        (RootCtx.holdsArg instanceMappingRelation_023 2 ?fy)))

:Rel instanceMappingRelation_041
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_041"
:rem "There exists a correspondence between the instances ?holex and ?holey as a result of both being asserted instances of the foundation class Foundation.Round_Hole declared in DomainX and DomainY respectively. ?holex and ?holey both share in common the property of having Foundation.blind hole bottom conditions."

(<= (instanceMappingRelation_041 ?holex ?holey)
    (and (RootCtx.inst ?holex Foundation.Round_Hole)
        (RootCtx.withinContext ?holex DomainX)
        (RootCtx.inst ?ccpx Foundation.Circular_Closed_Profile)
        (Foundation.holds_shape ?holex ?ccpx)
        (Foundation.blind ?ccpx)
        (RootCtx.inst ?holey Foundation.Round_Hole)
        (RootCtx.withinContext ?holey DomainY)
        (RootCtx.inst ?ccpy Foundation.Circular_Closed_Profile)
        (Foundation.holds_shape ?holey ?ccpy)
        (Foundation.blind ?ccpy)))

(=> (instanceMappingRelation_041 ?holex ?holey)
    (and (RootCtx.holdsArg instanceMappingRelation_041 1 ?holex)
        (RootCtx.holdsArg instanceMappingRelation_041 2 ?holey)))

:Rel instanceMappingRelation_042
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_042"
:rem "There exists a correspondence between the instances ?holex and ?holey as a result of both being asserted instances of the foundation class Foundation.Round_Hole declared in
```

DomainX and DomainY respectively. ?holex and ?holey both share in common the property of having Foundation.through hole bottom conditions."

```
(<= (instanceMappingRelation_042 ?holex ?holey)
    (and (RootCtx.inst ?holex Foundation.Round_Hole)
         (RootCtx.withinContext ?holex DomainX)
         (RootCtx.inst ?ccpx Foundation.Circular_Closed_Profile)
         (Foundation.holds_shape ?holex ?ccpx)
         (Foundation.through ?ccpx)
         (RootCtx.inst ?holey Foundation.Round_Hole)
         (RootCtx.withinContext ?holey DomainY)
         (RootCtx.inst ?ccpy Foundation.Circular_Closed_Profile)
         (Foundation.holds_shape ?holey ?ccpy)
         (Foundation.through ?ccpy)))

(=> (instanceMappingRelation_042 ?holex ?holey)
    (and (RootCtx.holdsArg instanceMappingRelation_042 1 ?holex)
         (RootCtx.holdsArg instanceMappingRelation_042 2 ?holey)))
```

:Rel **instanceMappingRelation_043**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_043"
:rem "There exists a correspondence between the instances ?holex and ?holey as a result of both being asserted instances of the foundation class Foundation.Round_Hole declared in DomainX and DomainY respectively. ?holex and ?holey do not share the same hole bottom conditions."

```
(<= (instanceMappingRelation_043 ?holex ?holey)
    (and (RootCtx.inst ?holex Foundation.Round_Hole)
         (RootCtx.withinContext ?holex DomainX)
         (RootCtx.inst ?ccpx Foundation.Circular_Closed_Profile)
         (Foundation.holds_shape ?holex ?ccpx)
         (Foundation.blind ?ccpx)
         (RootCtx.inst ?holey Foundation.Round_Hole)
         (RootCtx.withinContext ?holey DomainY)
         (RootCtx.inst ?ccpy Foundation.Circular_Closed_Profile)
         (Foundation.holds_shape ?holey ?ccpy)
         (Foundation.through ?ccpy)))

(=> (instanceMappingRelation_043 ?holex ?holey)
    (and (RootCtx.holdsArg instanceMappingRelation_043 1 ?holex)
         (RootCtx.holdsArg instanceMappingRelation_043 2 ?holey)))
```

:Rel **instanceMappingRelation_048**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_048"
:rem "There exists a correspondence between the instances ?holex and ?holey as a result of both being asserted instances of the foundation class Foundation.Round_Hole declared in DomainX and DomainY respectively. ?holex has a nominal entry diameter which is numerically smaller than that of ?holey."

```
(<= (instanceMappingRelation_048 ?holex ?holey)
    (and (RootCtx.inst ?ccpx Foundation.Circular_Closed_Profile)
         (not (Foundation.blind ?ccpx))
         (not (Foundation.through ?ccpx))
         (RootCtx.inst ?ccpy Foundation.Circular_Closed_Profile)
         (not (Foundation.blind ?ccpy))
         (not (Foundation.through ?ccpy))
```

(**instanceMappingRelation_003** ?ccpx ?ccpy)
(RootCtx.inst ?holex Foundation.Round_Hole)
(RootCtx.withinContext ?holex DomainX)
(Foundation.holds_shape ?holex ?ccpx)
(RootCtx.inst ?holey Foundation.Round_Hole)
(RootCtx.withinContext ?holey DomainY)
(Foundation.holds_shape ?holey ?ccpy)))

(=> (**instanceMappingRelation_048** ?holex ?holey)
(and (RootCtx.holdsArg **instanceMappingRelation_048** 1 ?holex)
(RootCtx.holdsArg **instanceMappingRelation_048** 2 ?holey)))

:Rel **instanceMappingRelation_054**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_054"
:rem "There exists a correspondence between the instances ?holex and ?holey as a result of both being asserted instances of the foundation class Foundation.Round_Hole declared in DomainX and DomainY respectively. ?holex has a nominal hole bottom diameter which is numerically smaller than that of ?holey."

(<= (**instanceMappingRelation_054** ?holex ?holey)
(and (RootCtx.inst ?ccpx Foundation.Circular_Closed_Profile)
(RootCtx.inst ?ccpy Foundation.Circular_Closed_Profile)
(**instanceMappingRelation_003** ?ccpx ?ccpy)
(or (**instanceMappingRelation_041** ?holex ?holey)
(**instanceMappingRelation_042** ?holex ?holey)
(**instanceMappingRelation_043** ?holex ?holey))
(RootCtx.inst ?holex Foundation.Round_Hole)
(RootCtx.withinContext ?holex DomainX)
(Foundation.holds_shape ?holex ?ccpx)
(RootCtx.inst ?holey Foundation.Round_Hole)
(RootCtx.withinContext ?holey DomainY)
(Foundation.holds_shape ?holey ?ccpy)))

(=> (**instanceMappingRelation_054** ?holex ?holey)
(and (RootCtx.holdsArg **instanceMappingRelation_054** 1 ?holex)
(RootCtx.holdsArg **instanceMappingRelation_054** 2 ?holey)))

:Rel **instanceMappingRelation_059**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_059"
:rem "There exists a correspondence between the instances ?holex and ?holey as a result of both being asserted instances of the foundation class Foundation.Round_Hole declared in DomainX and DomainY respectively. ?holex has a nominal hole depth which is numerically greater than that of ?holey."

(<= (**instanceMappingRelation_059** ?holex ?holey)
(and (RootCtx.inst ?linx Foundation.Linear_Path)
(RootCtx.inst ?liny Foundation.Linear_Path)
(**instanceMappingRelation_008** ?linx ?liny)
(RootCtx.inst ?holex Foundation.Round_Hole)
(RootCtx.withinContext ?holex DomainX)
(Foundation.holds_shape ?holex ?linx)
(RootCtx.inst ?holey Foundation.Round_Hole)
(RootCtx.withinContext ?holey DomainY)
(Foundation.holds_shape ?holey ?liny)))

(=> (**instanceMappingRelation_059** ?holex ?holey)
    (and (RootCtx.holdsArg **instanceMappingRelation_059** 1 ?holex)
        (RootCtx.holdsArg **instanceMappingRelation_059** 2 ?holey)))

:Rel **instanceMappingRelation_068**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_068"
:rem "There exists a correspondence between the instances ?holex and ?holey as a result of both being asserted instances of the foundation class Foundation.Round_Hole declared in DomainX and DomainY respectively. ?holex and ?holey both have Foundation.Transition_Feature instances that blend their entry and/or hole bottom surfaces. These Foundation.Transition_Feature instances are, however, different for both ?holex and ?holey."

(<= (**instanceMappingRelation_068** ?holex ?holey)
    (and (**pointerRelation_027** ?holex ?edgex)
        (**pointerRelation_030** ?holey ?chfy)
        (RootCtx.inst ?chfy Foundation.Chamfer)
        (RootCtx.inst ?edgex Foundation.Constant_Radius_Edge_Round)))

(<= (**instanceMappingRelation_068** ?holex ?holey)
    (and (**pointerRelation_029** ?holex ?chfx)
        (**pointerRelation_028** ?holey ?edgey)
        (RootCtx.inst ?chfx Foundation.Chamfer)
        (RootCtx.inst ?edgey Foundation.Constant_Radius_Edge_Round)))

(=> (**instanceMappingRelation_068** ?holex ?holey)
    (and (RootCtx.holdsArg **instanceMappingRelation_068** 1 ?holex)
        (RootCtx.holdsArg **instanceMappingRelation_068** 2 ?holey)))

:Rel **instanceMappingRelation_070**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_070"
:rem "There exists a correspondence between the instances ?fx and ?fy as a result of both being asserted instances of the foundation class Foundation.Feature declared in DomainX and DomainY respectively. ?fx and ?fy are both compound features which are composed of at least two Foundation.Round_Hole instances defined in DomainX and DomainY respectively."
:limitationRem "It is not immediately possible to infer whether ?fx and ?fy have similar geometric complexity based on the features that they aggregate."

(<= (**instanceMappingRelation_070** ?fx ?fy)
    (and (RootCtx.inst ?fx Foundation.Feature)
        (RootCtx.inst ?fx1 Foundation.Round_Hole)
        (RootCtx.inst ?fx2 Foundation.Round_Hole)
        (Foundation.compound ?fx)
        (RootCtx.withinContext ?fx DomainX)
        (Foundation.element_of ?fx1 ?fx)
        (Foundation.element_of ?fx2 ?fx)
        (RootCtx.inst ?fy Foundation.Feature)
        (RootCtx.inst ?fy1 Foundation.Round_Hole)
        (RootCtx.inst ?fy2 Foundation.Round_Hole)
        (Foundation.compound ?fy)
        (RootCtx.withinContext ?fy DomainY)
        (Foundation.element_of ?fy1 ?fy)
        (Foundation.element_of ?fy2 ?fy)))

(=> (**instanceMappingRelation_070** ?fx ?fy)
    (and (RootCtx.holdsArg **instanceMappingRelation_070** 1 ?fx)
        (RootCtx.holdsArg **instanceMappingRelation_070** 2 ?fy)))

:Rel **instanceMappingRelation_071**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_071"
:rem "There exists a correspondence between the instances ?occx and ?occy as a result of both being asserted instances of the foundation class Foundation.Activity_Occurrence declared in DomainX and DomainY respectively. ?occx and ?occy are both hole making activity occurrences based on the reasoning that instances of Foundation.Round_Hole defined in DomainX and DomainY respectively are Foundation.output from ?occx and ?occy."
:limitationRem "It is not immediately possible to infer whether ?occx and ?occy are the same activity occurrences purely based on the fact that hole features are Foundation.output from them. Explication mismatches could be present between the two instances as a result of possible Concept C, Definiens D and Term T disagreements."

(<= (**instanceMappingRelation_071** ?occx ?occy)
    (and (**pointerRelation_035** ?occx)
        (**pointerRelation_036** ?occy)))

(=> (**instanceMappingRelation_071** ?occx ?occy)
    (and (RootCtx.holdsArg **instanceMappingRelation_071** 1 ?occx)
        (RootCtx.holdsArg **instanceMappingRelation_071** 2 ?occy)))

:Rel **instanceMappingRelation_072**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_072"
:rem "There exists a correspondence between the instances ?occx and ?occy as a result of both being asserted instances of the foundation class Foundation.Activity_Occurrence declared in DomainX and DomainY respectively. ?occx and ?occy are both complex Foundation.Activity_Occurrence instances defined in DomainX and DomainY respectively. Both ?occx and ?occy hold a number of subactivity occurrences."

(<= (**instanceMappingRelation_072** ?occx ?occy)
    (and (RootCtx.inst ?occx Foundation.Activity_Occurrence)
        (RootCtx.withinContext ?occx DomainX)
        (RootCtx.inst ?occx1 Foundation.Activity_Occurrence)
        (Foundation.subactivity_occurrence ?occx1 ?occx)
        (RootCtx.inst ?occy Foundation.Activity_Occurrence)
        (RootCtx.withinContext ?occy DomainY)
        (RootCtx.inst ?occy1 Foundation.Activity_Occurrence)
        (Foundation.subactivity_occurrence ?occy1 ?occy)))

(=> (**instanceMappingRelation_072** ?occx ?occy)
    (and (RootCtx.holdsArg **instanceMappingRelation_072** 1 ?occx)
        (RootCtx.holdsArg **instanceMappingRelation_072** 2 ?occy)))

:Rel **instanceMappingRelation_074**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_074"
:rem "There exists a correspondence between the instances ?occx and ?occy as a result of both being asserted instances of the foundation class Foundation.Activity_Occurrence declared in DomainX and DomainY respectively. ?occx and ?occy are both Foundation.initial Foundation.Activity_Occurrence instances defined in DomainX and DomainY respectively. This implies that both ?occx and ?occy are the very first occurrences in their respective occurrence trees."

(<= (**instanceMappingRelation_074** ?occx ?occy)
    (and (RootCtx.inst ?occx Foundation.Activity_Occurrence)
        (RootCtx.withinContext ?occx DomainX)
        (Foundation.initial ?occx)
        (RootCtx.inst ?occy Foundation.Activity_Occurrence)
        (RootCtx.withinContext ?occy DomainY)
        (Foundation.initial ?occy)))

(=> (**instanceMappingRelation_074** ?occx ?occy)
    (and (RootCtx.holdsArg **instanceMappingRelation_074** 1 ?occx)
        (RootCtx.holdsArg **instanceMappingRelation_074** 2 ?occy)))

:Rel **instanceMappingRelation_075**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_075"
:rem "There exists a correspondence between the instances ?occx and ?occy as a result of both being asserted instances of the foundation class Foundation.Activity_Occurrence declared in DomainX and DomainY respectively. ?occx and ?occy are both Foundation.arboreal and Foundation.legal Foundation.Activity_Occurrence instances defined in DomainX and DomainY respectively. This implies that both ?occx and ?occy are allowable occurrences in their respective occurrence trees."

(<= (**instanceMappingRelation_075** ?occx ?occy)
    (and (RootCtx.inst ?occx Foundation.Activity_Occurrence)
        (RootCtx.withinContext ?occx DomainX)
        (Foundation.legal ?occx)
        (Foundation.arboreal ?occx)
        (RootCtx.inst ?occy Foundation.Activity_Occurrence)
        (RootCtx.withinContext ?occy DomainY)
        (Foundation.legal ?occy)
        (Foundation.arboreal ?occy)))

(=> (**instanceMappingRelation_075** ?occx ?occy)
    (and (RootCtx.holdsArg **instanceMappingRelation_075** 1 ?occx)
        (RootCtx.holdsArg **instanceMappingRelation_075** 2 ?occy)))

:Rel **instanceMappingRelation_077**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_077"
:rem "There exists a correspondence between the instances ?fx and ?fy as a result of both being asserted instances of the foundation class Foundation.Feature declared in DomainX and DomainY respectively. ?fx and ?fy are both compound feature instances defined in DomainX and DomainY respectively."
:limitationRem "It is not immediately possible to infer whether ?fx and ?fy involve counterbore and/or countersunk compound hole features."

(<= (**instanceMappingRelation_077** ?fx ?fy)
    (and (RootCtx.inst ?fx Foundation.Feature)
        (RootCtx.withinContext ?fx DomainX)
        (Foundation.compound ?fx)
        (RootCtx.inst ?fy Foundation.Feature)
        (RootCtx.withinContext ?fy DomainY)
        (Foundation.compound ?fy)))

(=> (**instanceMappingRelation_077** ?fx ?fy)
    (and (RootCtx.holdsArg **instanceMappingRelation_077** 1 ?fx)
        (RootCtx.holdsArg **instanceMappingRelation_077** 2 ?fy)))

E.2 Semantic Mapping Concepts Based on Known Cross-Domain Correspondences (Design and Machining Hole Feature Ontology A)

Context Declaration

```
:Ctx domainMapping
:Inst UserContext
:supCtx TopUserCtx
:name "Domain Mapping Context"
:rem "This context is used to define relevant semantic mapping concepts based on known cross-domain correspondences between the Design Hole Feature Ontology A and Machining Hole Feature Ontology A."

:Use domainMapping
```

Reconciliation of Classes

```
:Rel classMappingRelation_001
:Inst BinaryRel
:Sig Property Property
:name "classMappingRelation_001"
:rem "The class ?x in the DomainX context is a conceptually similar class to the class ?y in the DomainY context. The class ?x has been declared from a design function viewpoint whereas the class ?y has been declared from a machining viewpoint."
:limitationRem "It is possible that there is a term and definiens mismatch between the classes ?x and ?y. This would arise in the event that different terms and semantic structures have been chosen to refer to the classes ?x and ?y as a result of domain preferences."

(=> (classMappingRelation_001 ?x ?y)
    (and (RootCtx.holdsArg classMappingRelation_001 1 ?x)
         (RootCtx.holdsArg classMappingRelation_001 2 ?y)))

(<= (classMappingRelation_001 DomainX.Housing_Part_Family
                              DomainY.Housing_Part_Family)
    (and (RootCtx.Property DomainX.Housing_Part_Family)
         (RootCtx.Property DomainY.Housing_Part_Family)))


(<= (classMappingRelation_001 DomainX.Bolt_Hole DomainY.Counterbore_Hole)
    (and (RootCtx.Property DomainX.Bolt_Hole)
         (RootCtx.Property DomainY.Counterbore_Hole)))

(<= (classMappingRelation_001 DomainX.Boss DomainY.Turned_Boss)
    (and (RootCtx.Property DomainX.Boss)
         (RootCtx.Property DomainY.Turned_Boss)))

(<= (classMappingRelation_001 DomainX.External_Flange DomainY.Turned_Flange)
    (and (RootCtx.Property DomainX.External_Flange)
         (RootCtx.Property DomainY.Turned_Flange)))

(<= (classMappingRelation_001 DomainX.Locating_Pin_Hole DomainY.Reamed_Hole)
    (and (RootCtx.Property DomainX.Locating_Pin_Hole)
         (RootCtx.Property DomainY.Reamed_Hole)))
```

(<= (**classMappingRelation_001** DomainX.Plain_**Diameter**_**Hole** DomainY.**Drilled**_**Hole**)
    (and (RootCtx.Property DomainX.Plain_Diameter_Hole)
        (RootCtx.Property DomainY.Drilled_Hole)))

(<= (**classMappingRelation_001** DomainX.**Secondary**_**Hole** DomainY.**Counterbore**)
    (and (RootCtx.Property DomainX.Secondary_Hole)
        (RootCtx.Property DomainY.Counterbore_Hole)))

(<= (**classMappingRelation_001** DomainX.**Primary**_**Depth** DomainY.**Drilled**_**Hole**_**Depth**)
    (and (RootCtx.Property DomainX.Primary_Depth)
        (RootCtx.Property DomainY.Drilled_Hole_Depth)))

(<= (**classMappingRelation_001** DomainX.**Primary**_**Diameter**
                                DomainY.**Drilled**_**Hole**_**Diameter**)
    (and (RootCtx.Property DomainX.Primary_Depth)
        (RootCtx.Property DomainY.Drilled_Hole_Depth)))

(<= (**classMappingRelation_001** DomainX.**Secondary**_**Depth**
                                DomainY.**Counterbore**_**Depth**)
    (and (RootCtx.Property DomainX.Secondary_Depth)
        (RootCtx.Property DomainY.Counterbore_Depth)))

(<= (**classMappingRelation_001** DomainX.**Secondary**_**Diameter**
                                DomainY.**Counterbore**_**Diameter**)
    (and (RootCtx.Property DomainX.Secondary_Diameter)
        (RootCtx.Property DomainY.Counterbore_Diameter)))

(<= (**classMappingRelation_001** DomainX.**Primary**_**Depth**
                                DomainY.**Reamed**_**Hole**_**Depth**)
    (and (RootCtx.Property DomainX.Primary_Depth)
        (RootCtx.Property DomainY.Reamed_Hole_Depth)))

(<= (**classMappingRelation_001** DomainX.**Primary**_**Diameter**
                                DomainY.**Reamed**_**Hole**_**Diameter**)
    (and (RootCtx.Property DomainX.Primary_Depth)
        (RootCtx.Property DomainY.Reamed_Hole_Depth)))

:Rel **classMappingRelation_002**
:Inst BinaryRel
:Sig Property Property
:name "classMappingRelation_002"
:rem "From a feature geometry standpoint, the dimensional parameters that define DomainX.Bolt_Hole instances differ from those of DomainY.Counterbore_Hole instances."

(<= (**classMappingRelation_002** DomainX.**Bolt**_**Hole** DomainY.**Counterbore**_**Hole**)
    (and (RootCtx.Property DomainX.Bolt_Hole)
        (RootCtx.Property DomainY.Counterbore_Hole)))

(=> (**classMappingRelation_002** ?x ?y)
    (and (RootCtx.holdsArg **classMappingRelation_002** 1 ?x)
        (RootCtx.holdsArg **classMappingRelation_002** 2 ?y)))

:Rel **classMappingRelation_003**
:Inst BinaryRel
:Sig Property Property
:name "classMappingRelation_003"
:rem "When applied to a DomainX.Bolt_Hole, a DomainX.Primary_Depth is the subtraction of a DomainY.Counterbore_Depth from a DomainY.Drilled_Hole_Depth. In other words, a

Drilled_Hole_Depth is the addition of a DomainX.Secondary_Depth to a Primary_Depth for a DomainY.Counterbore_Hole."

```
(<= (classMappingRelation_003 DomainX.Primary_Depth DomainY.Drilled_Hole_Depth)
    (and (RootCtx.Property DomainX.Primary_Depth)
         (RootCtx.Property DomainY.Drilled_Hole_Depth)))

(=> (classMappingRelation_003 ?x ?y)
    (and (RootCtx.holdsArg classMappingRelation_003 1 ?x)
         (RootCtx.holdsArg classMappingRelation_003 2 ?y)))

:Rel classMappingRelation_004
:Inst BinaryRel
:Sig Property Property
:name "classMappingRelation_004"
:rem "A DomainX.Boss in DomainX is a feature of compound property which consists of an aggregation of a Foundation.Cylinder and a Foundation.Round_Hole, whereas a DomainY.Turned_Boss in DomainY is not."

(<= (classMappingRelation_004 DomainX.Boss DomainY.Turned_Boss)
    (and (RootCtx.Property DomainX.Boss)
         (RootCtx.Property DomainY.Turned_Boss)))

(=> (classMappingRelation_004 ?x ?y)
    (and (RootCtx.holdsArg classMappingRelation_004 1 ?x)
         (RootCtx.holdsArg classMappingRelation_004 2 ?y)))
```

Reconciliation of Functions

```
:Rel pointerRelation_001
:Inst UnaryRel
:Sig UnaryFun
:name "pointerRelation_001"

(<= (pointerRelation_001 ?funx)
    (and (RootCtx.inst ?funx RootCtx.UnaryFun)
         (RootCtx.withinContext ?funx DomainX)
         (RootCtx.argProp ?funx 1 RootCtx.RealNumber)
         (RootCtx.returnProp ?funx Foundation.Length_Measure)))

:Rel pointerRelation_002
:Inst UnaryRel
:Sig UnaryFun
:name "pointerRelation_002"

(<= (pointerRelation_002 ?funy)
    (and (RootCtx.inst ?funy RootCtx.UnaryFun)
         (RootCtx.withinContext ?funy DomainY)
         (RootCtx.argProp ?funy 1 RootCtx.RealNumber)
         (RootCtx.returnProp ?funy Foundation.Length_Measure)))

:Rel functionMappingRelation_001
:Inst BinaryRel
:Sig Property Property
:name "functionMappingRelation_001"
:rem "The function ?funx in the DomainX context is equivalent to the function ?funy in the DomainY context. There are no semantic mismatches between them."
```

(<= (**functionMappingRelation_001** ?funx ?funy)
    (and (**pointerRelation_001** ?funx)
        (**pointerRelation_002** ?funy)))

(=> (**functionMappingRelation_001** ?funx ?funy)
    (and (RootCtx.holdsArg **functionMappingRelation_001** 1 ?funx)
        (RootCtx.holdsArg **functionMappingRelation_001** 2 ?funy)))

Reconciliation of Instances

:Rel **pointerRelation_003**
:Inst TernaryRel
:Sig Property Property Property
:name "pointerRelation_003"

(<= (**pointerRelation_003** ?x ?sax ?realx)
    (and (RootCtx.inst ?x Foundation.Feature)
        (RootCtx.withinContext ?x DomainX)
        (RootCtx.inst ?sax Foundation.Shape_Aspect)
        (Foundation.holds_shape ?x ?sax)
        (RootCtx.inst ?lengthx Foundation.Length_Measure)
        (= ?lengthx (Foundation.mm ?realx))
        (Foundation.measures ?sax (Foundation.mm ?realx))))

:Rel **pointerRelation_004**
:Inst TernaryRel
:Sig Property Property Property
:name "pointerRelation_002"

(<= (**pointerRelation_004** ?y ?say ?realy)
    (and (RootCtx.inst ?y Foundation.Feature)
        (RootCtx.withinContext ?y DomainY)
        (RootCtx.inst ?say Foundation.Shape_Aspect)
        (Foundation.holds_shape ?y ?say)
        (RootCtx.inst ?lengthy Foundation.Length_Measure)
        (= ?lengthy (Foundation.mm ?realy))
        (Foundation.measures ?say (Foundation.mm ?realy))))

:Rel **instanceMappingRelation_001**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_001"
:rem "The Foundation.Round_Hole instance ?x in the DomainX context is an equivalent individual to the Foundation.Round_Hole instance ?y in the DomainY context, in terms of the nominal dimensional parameters that each instance carries. The instance ?x has been declared from a design function viewpoint whereas the instance ?y has been declared from a machining viewpoint."
:limitationRem "Dimensional tolerances and orientations carried by the instances ?x and ?y have not been considered in the reasoning. Furthermore, is possible that there is a term and definiens mismatch between the instances ?x and ?y. This would arise in the event that different terms and semantic structures have been chosen to refer to the instances ?x and ?y as a result of domain preferences."

(<= (**instanceMappingRelation_001** ?x ?y)
    (and (**pointerRelation_003** ?x ?sax1 ?realx1)
        (**pointerRelation_004** ?y ?say1 ?realy1)
        (**pointerRelation_003** ?x ?sax2 ?realx2)
        (**pointerRelation_004** ?y ?say2 ?realy2)

(RootCtx.inst ?x Foundation.Round_Hole)
(RootCtx.inst ?y Foundation.Round_Hole)
(RootCtx.inst ?sax1 Foundation.Circular_Closed_Profile)
(RootCtx.inst ?say1 Foundation.Circular_Closed_Profile)
(RootCtx.inst ?sax2 Foundation.Linear_Path)
(RootCtx.inst ?say2 Foundation.Linear_Path)
(eqNum ?realx1 ?realy1)
(eqNum ?realx2 ?realy2)))

(=> (**instanceMappingRelation_001** ?x ?y)
(and (RootCtx.holdsArg **instanceMappingRelation_001** 1 ?x)
(RootCtx.holdsArg **instanceMappingRelation_001** 2 ?y)))

:Rel **instanceMappingRelation_002**
:Inst BinaryRel
:Sig Property Property
:name "instanceMappingRelation_002"
:rem "The Foundation.Cylinder instance ?x in the DomainX context is an equivalent individual to the Foundation.Cylinder instance ?y in the DomainY context, in terms of the nominal dimensional parameters that each instance carries.The instance ?x has been declared from a design function viewpoint whereas the instance ?y has been declared from a machining viewpoint."
:limitationRem "Dimensional tolerances and orientations carried by the instances ?x and ?y have not been considered in the reasoning. Furthermore, is possible that there is a term and definiens mismatch between the instances ?x and ?y. This would arise in the event that different terms and semantic structures have been chosen to refer to the instances ?x and ?y as a result of domain preferences."

(<= (**instanceMappingRelation_002** ?x ?y)
(and (**pointerRelation_003** ?x ?sax1 ?realx1)
(**pointerRelation_004** ?y ?say1 ?realy1)
(**pointerRelation_003** ?x ?sax2 ?realx2)
(**pointerRelation_004** ?y ?say2 ?realy2)
(RootCtx.inst ?x Foundation.Cylinder)
(RootCtx.inst ?y Foundation.Cylinder)
(RootCtx.inst ?sax1 Foundation.Circular_Closed_Profile)
(RootCtx.inst ?say1 Foundation.Circular_Closed_Profile)
(RootCtx.inst ?sax2 Foundation.Linear_Path)
(RootCtx.inst ?say2 Foundation.Linear_Path)
(eqNum ?realx1 ?realy1)
(eqNum ?realx2 ?realy2)))

(=> (**instanceMappingRelation_002** ?x ?y)
(and (RootCtx.holdsArg **instanceMappingRelation_002** 1 ?x)
(RootCtx.holdsArg **instanceMappingRelation_002** 2 ?y)))

F Interoperability Evaluation Layer

F.1 Sitemap for the Interoperability Evaluation Assistant

Figure F-1 illustrates the sitemap for the Interoperability Evaluation Assistant. The Interoperability Evaluation Assistant is a Web-based UI which enables the user to retrieve the appropriate queries during the interoperable knowledge discovery process at the fourth level of the SMIF.

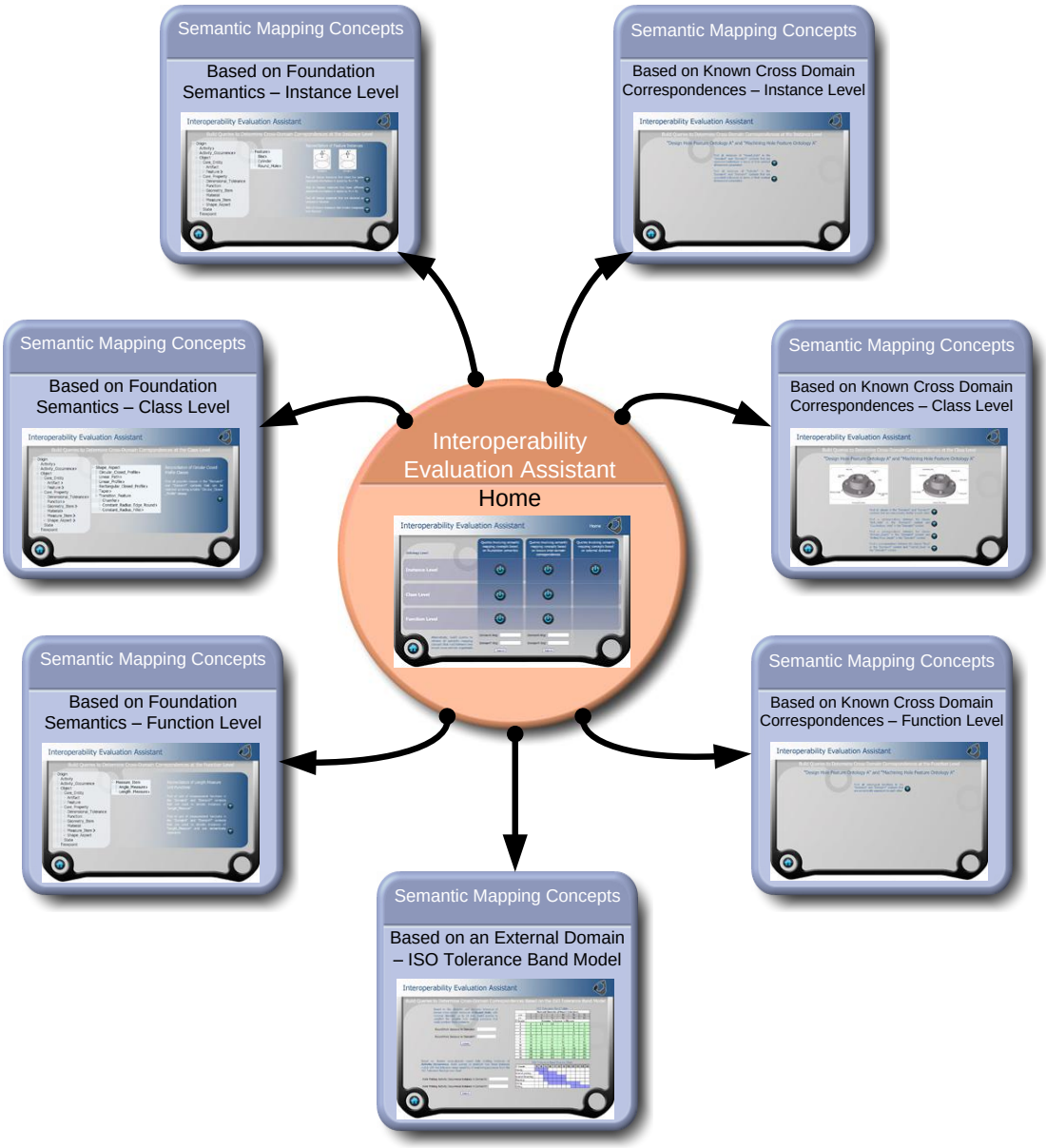
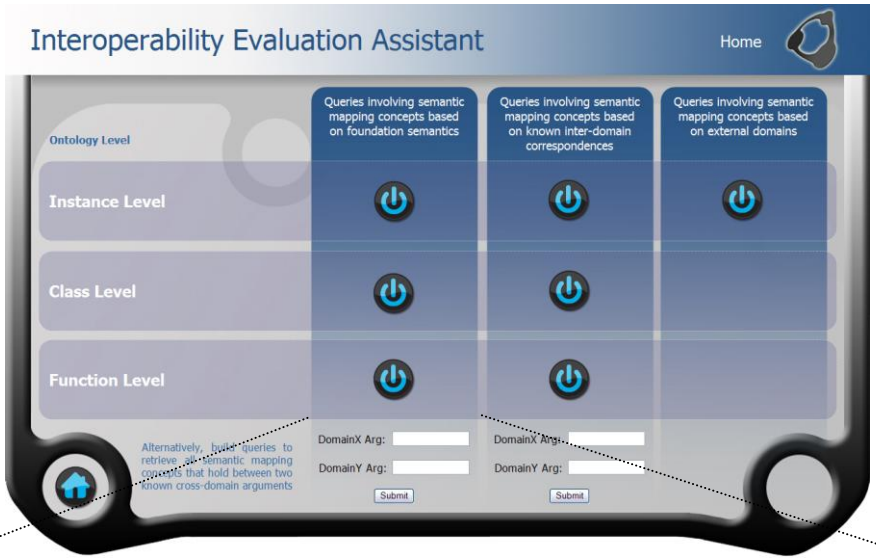


Figure F-49 Sitemap for the Interoperability Evaluation Assistant

F.2 Java-Based Modules

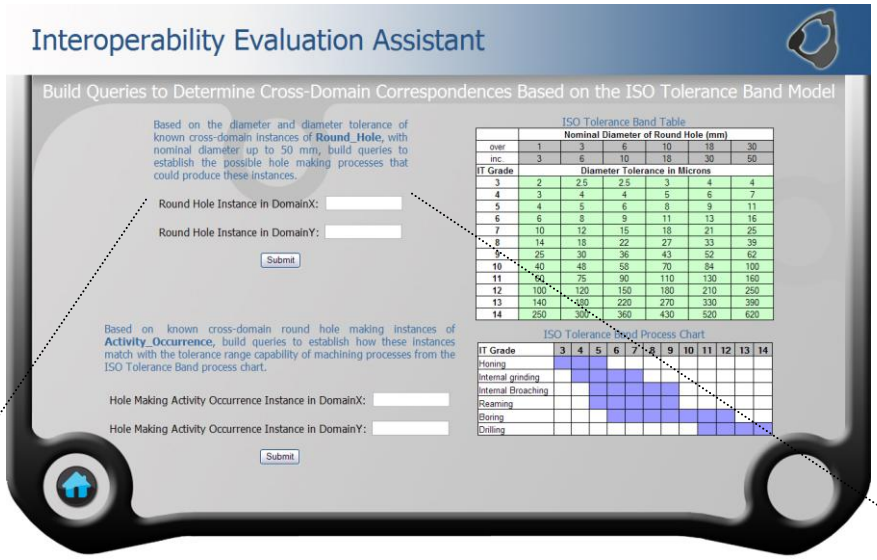
The Interoperability Evaluation Assistant employs Java-based modules to input the names of domain arguments and retrieve queries to evaluate all cross-domain correspondences in a single transaction. These modules are written in JavaScript and embedded in the HTML code of the relevant page on the Web-based interface. Figure F-2 depicts the main panel of the Interoperability Evaluation Assistant and a sample JavaScript code for one of the Java-based modules appearing on the Homepage.




```
<div style="position: absolute; width: 250px; height: 114px; z-index: 11; left: 498px; top: 655px" id="layer62">
<html>
<head>
<title>Retrieve Semantic Mapping Concepts 1</title>
<script LANGUAGE="JavaScript" type="text/javascript">
function display1() {
  DispWin = window.open('','NewWin', 'toolbar=no,status=no,width=300,height=150')
  message = "(and (RootCtx.BinaryRel ?rel) (RootCtx.withinContext ?rel foundationMapping) (RootCtx.holdsArg
?rel 1 DomainX." + document.form1.domainx1.value;
  message += ") (RootCtx.holdsArg ?rel 2 DomainY." + document.form1.domainy1.value;
  message += "))";
  DispWin.document.write(message);
}
</script>
</head>
<body>
<form name="form1">
<p align="center"><font face="Tahoma">DomainX Arg:  </font>
<input TYPE="TEXT" SIZE="15" NAME="domainx1"><p align="center">
</p>
<p align="center"><font face="Tahoma">DomainY Arg:</font> 
<input TYPE="TEXT" SIZE="15" NAME="domainy1">
</p>
<p align="center">
<input TYPE="BUTTON" VALUE="Submit" onClick="display1();" style="float: centre"></p>
</form>
</body>
</html>
            <p> </div>
```

Figure F-2 Sample JavaScript Code for a Java-Based Module on the Homepage

Figure F-3 illustrates the page for building queries in order to evaluate cross-domain correspondences based on the ISO Tolerance Band Model as an external domain. A sample JavaScript code applicable to one of the Java-based modules present on this page is also listed.




```
<div style="position: absolute; width: 382px; height: 114px; z-index: 11; left: 260px; top: 302px" id="layer62">
<html>
<head>
<title>Retrieve Semantic Mapping Concepts 1</title>
<script LANGUAGE="JavaScript" type="text/javascript">
function display1() {
  DispWin = window.open('','NewWin', 'toolbar=no,status=no,width=320,height=200')
  message = "(and (RootCtx.UnaryRel ?rel1) (RootCtx.UnaryRel ?rel2) (RootCtx.withinContext ?rel1
isoToleranceBand) (RootCtx.withinContext ?rel2 isoToleranceBand) (RootCtx.holdsArg ?rel1 1 DomainX." +
document.form1.domainx1.value;
  message += ") (RootCtx.holdsArg ?rel2 1 DomainY." + document.form1.domainy1.value;
  message += "))";
  DispWin.document.write(message);
}
</script>
</head>
<body>
<form name="form1">
<p align="center"><font face="Tahoma">Round Hole Instance in DomainX:  </font>
<input TYPE="TEXT" SIZE="15" NAME="domainx1"><p align="center">
</p>
<p align="center"><font face="Tahoma">Round Hole Instance in DomainY:</font> 
<input TYPE="TEXT" SIZE="15" NAME="domainy1">
</p>
<p align="center">
<input TYPE="BUTTON" VALUE="Submit" onClick="display1();" style="float: centre"></p>
</form>
</body>
</html>
            <p> </div>
```

Figure F-3 Sample JavaScript Code for a Java-Based Module on the ISO Tolerance Band Model Page