Loughborough University

Towards an Assembly Reference Ontology for Assembly Knowledge Sharing

By

Muhammad Imran

A Doctoral Thesis

Submitted in partial fulfilment of the requirements
for the award of
Doctor of Philosophy of Loughborough University

September 2013

Loughborough University

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)

Muhammad Imran

……………………………………………. (Date)

Abstract

Information and Communication Technologies (ICT) have been increasingly used to support the decision making in manufacturing organizations however they lack the ability to fully support the capture and sharing of specific domain knowledge across multiple domains. The ability of ICT based systems to share knowledge is impeded by the semantic conflicts arising from loosely defined meanings and intents of the participating concepts. This research work exploits the concept of formal ontologies to rigorously define the semantics of domain concepts to support knowledge sharing within the assembly domain.

In this thesis, a novel research framework has been proposed in the form of a assembly reference ontology which can provide a common semantic base to support knowledge sharing across the assembly design and assembly process planning domains. The framework consists of a set of key reference concepts identified to represent the assembly domain related knowledge. These concepts have been specialized from the most generic level to the most specialized level and have been formally defined to support the capture and sharing of assembly knowledge. The proposed framework also supports the creation of application specific ontologies by providing them with a common semantic base.

The research concept has been experimentally investigated by using a selected set of assembly reference concepts which have been used to formally represent and relate assembly design and assembly process planning knowledge. The results of the experiments verify that the implemented ontology facilitates the system to understand the semantics of concepts and supports knowledge sharing across the assembly design and assembly process planning domains. The experimental results also show that the proposed framework can also support the development of a range of application specific ontologies.

Keywords: knowledge sharing, semantics, formal ontologies, assembly knowledge, assembly design and assembly process planning

Dedication

I dedicate this work to my beloved mother Husaina Bibi who passed away on 25th November 2012. To me, she was a source of inspiration and guidance. This moment is a mixture of sadness and happiness for me. I would have been the happiest person if she could have lived to date to see me back home however I am happy that her prayers are still with me. May Almighty Allah bless her soul.

Acknowledgements

First of all I thank to Almighty Allah who has given me the power to gain understanding of this research work.

After that I would like to thank my supervisor Dr. Bob Young who has provided me insightful guidance throughout my PhD tenure. I was really impressed by his supervision and technical skills during my MSc project and therefore I made my mind to undertake PhD research under his supervision. He always encouraged me whenever I got stuck in the research and provided me useful feedback and positive criticism which have helped me to produce this research work. Without his help and guidance it would not have been possible for me to accomplish this research work.

I would like to extend my thanks to Dr. Nitishal Chungoora who has provided me useful help in learning the KFL and IODE. I am especially thankful to Lynne Plettenberg from Highfleet who has always quickly responded my queries related to KFL and IODE. I am also thankful to Dr. Zahid Usman who has always encouraged me and provided useful feedback to my queries. I would also like to seize this opportunity to say thanks to Dr. George Gunendran and Dr. Najam Akber Anjum for their support.

I would also like to say thanks to Atta and Farukh (my housemates) for their support and encouragement. I am also thankful to Dr. Amjad Hussain who has always encouraged me whenever I felt depressed. I wish thanks to Dr. Sri Krishna Kumar and Dr. Claire Palmer for their support. I also like to say thanks to all others who have directly or indirectly supported me during my PhD.

Finally, I would like to say thanks to my siblings specially my elder brother Muhammad Saleem Akhtar who has provided me moral support during my PhD. I also wish to forward special thanks to my father whose prayers have never let me down.

Contents

List of Figures

List of Tables

Acronyms

AD	As Designed
ADACOR	ADaptive holonic COntrol aRchitecture for distributed manufacturing system
AF	Assembly Feature
AP	Application Protocol
API	Application Programming Interface
APP	Assembly Process Planning
AR	As Required
ARF	Assembly Resource Feature
ARO	Assembly Reference Ontology
ASP	Assembly Sequence Planning
AyD	Assembly Design
BFO	Basic Formal Ontology
BOM	Bill of Materials
BOP	Bill of Process
BOR	Bill of Resource
CL	Common Logic
CWA	Closed World Assumption
DFA	Design For Assembly

DFM	Design For Manufacture
DoD	Department of Defence
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
EA	Enterprise Architect
EBOM	Engineering Bill of Materials
EOL	End-Of-Life
Flogic	Frame Logic
GFO	General Formal Ontology
GT	Glossary of Terms
IC	Integrity Constraint
ICT	Information and Communication Technology
IODE	Integrated Ontology Development Environment
KFL	Knowledge Frame Language
KIF	Knowledge Interchange Format
MASON	Manufacturing’s Semantic Ontology
MBOM	Manufacturing Bill of Materials
MDA	Model Driven Architecture
MDI	Model Driven Interoperability
MLO	Middle Level Ontology
MRP	Material Requirement Planning

MSE	Manufacturing System Engineering
OCHRE	Object-Centred High-level Reference Ontology
OEM	Original Equipment Manufacturer
OKBC	Open Knowledge Base Connectivity
OMG	Object Management Group
OWA	Open World Assumption
OWL	Web Ontology Language
PF	Product Family
PLM	Product Lifecycle Management
RDF	Resource Description Framework
STEP	STandard for the Exchange of Product model data
SUMO	Suggested Upper Merged Ontology
SWRL	Semantic Web Rule Language
ULO	Upper Level Ontology
UML	Unified Modelling Language

CHAPTER 1

INTRODUCTION

1.1 Research Background

With the growing competition in the manufacturing sector, manufacturers are pushed to shorten the product development cycle to produce customized products within the minimum possible time. One of the factors which help manufacturing organizations to remain competitive is the collaboration environment in which all stakeholders can share information across the product manufacturing domains. Due to the boom in Information and Communication Technologies (ICTs) over the last couple of decades, industries are using them as a support tool for the capture and exchange of information within and across multiple domains.

However these enterprise applications resist collaboration efforts (Chen and Doumeingts, 2003) due to their limited capability in representing and sharing information (Young et al., 2010). The solution to this problem lies in addressing the interoperability issues (Ouksel and Sheth, 1999). Interoperability can be achieved by establishing common or equivalent semantics (Chen and Vernadat, 2004) across multiple software systems. It has been found that ontologies can help to establish these common semantics which can consequently support knowledge sharing across multiple software systems.

Ontologies are broadly categorized into lightweight and heavyweight ontologies (Gomez-Perez et al. 2004). The lightweight ontologies are based on textual definitions of concepts and terms (Young et al., 2007) and are susceptible to multiple interpretations (Chungoora et al., 2013) which can lead to ambiguities when defining the semantics of concepts. The heavyweight version of ontologies restricts the meanings of terms by applying constraints and deduces new knowledge by using the inference rules. In spite of potential benefits of heavyweight or formal ontologies, it is uncommon to find wide spread applications of formal ontologies, in particular, in the manufacturing domain.

This research explores specifically the manufacturing assembly domain concepts and focuses on the capture and sharing of assembly knowledge in this domain. Assembly design and assembly process planning are two important sub-domains in the assembly domain which require frequent collaboration for efficient product development. However these domains represent different perspectives of the information associated with assembly related concepts which could lead to interoperability issues. In an environment where multiple software systems are involved, interoperability issues could cost millions if they are not identified and settled at the initial stages.

Product assembly is a complex task and especially complex in industries like automotive and aerospace where a large number of components are assembled together across a range of product variants which may require a variety of different assembly resources. Different departments across the supply network of the Original Equipment Manufacturer (OEM) can be involved in the design, and assembly of these components and the resources needed for their assembly. Due to multi-faceted understanding that each department brings to the problem, the information related with the assembly components becomes problematic when it is necessary to share this information across these departments. The participating departments which use computer based applications to store and retrieve information are likely to use different underlying formats for their applications which will cause interoperability issues (Lopez-Ortega and Ramirez, 2005). This is because these software systems use different standards to represent information (Ray and Jones, 2006). While these software systems may operate well in isolation however when subjected to the need to share knowledge, they fail to serve their purpose (Cochrane et al., 2005). This research work proposes a common semantic base in the form of a formal assembly reference ontology which provides intermediate concepts that can be linked to multiple domains. A formal ontology requires the application of axioms to establish the formal semantics. A formal ontology is also known as a heavyweight ontology (Borgo and Lesmo, 2008). Thus the formal assembly reference ontology is a heavyweight ontology which uses axioms to capture the semantics of assembly concepts.

1.2 Aims and Objectives

The aim of this research work is to provide and improve the understanding of how heavyweight ontological approaches can provide and improve support of knowledge sharing across assembly design and assembly planning process.

This aim would be addressed by the proposed hypothesis described in the next section. The accomplishment of the aim of this research should enhance the understanding of the use of ontology based decision support systems for Product Lifecycle Management (PLM) in general and product assembly in particular. Furthermore, knowledge base developed by heavyweight ontological approach in this research, retains useful assembly information that can be easily and effectively retrieved, shared, and reused across and within the product assembly domains.

In order to achieve the aim of this research, four main objectives have been recognized as follows.

1. To review the relevant literature to identify the research gap.
2. To propose a method for improved assembly knowledge sharing.
3. To design the knowledge sharing environment based on the new method.
4. Build an experimental system to evaluate the ideas developed in the research.

These objectives have been further explained in the following table.

Table 1-1: Explanation of objectives in context of the research work for this thesis

Objectives	Objectives Explained
Objective 1 To review the relevant literature to identify significant research gap.	A comprehensive literature review involving knowledge sharing, interoperability, ontologies, and assembly is required to understand the related existing research and to identify suitable research gap.
Objective 2 To propose a method for improved assembly knowledge sharing.	A mechanism is required to support interoperability across multiple assembly systems. This can be done by creating an assembly reference ontology comprising of reference concepts at various levels of specializations. This will help to capture the depth of meanings at these specialization levels. The proposed method has been explained in section 3.4.1 of chapter 3 where the specialization levels have been shown in figure 3.2. The formally defined assembly reference ontology can be used as a semantic base for multiple assembly systems to support assembly knowledge sharing. This method improves the support of knowledge sharing because the assembly reference ontology can be used as a common ontology for a range of application specific assembly ontologies.
Objective 3 To design the knowledge sharing environment based on the new method.	To identify a set of reference concepts for the assembly domain. To define (informally and formally) a set of assembly reference concepts for the assembly domain. Identifying the ways to relate assembly design and assembly process planning domains using the reference

	ontology.
Objective 4 Build an experimental system to evaluate the ideas developed in this research.	To implement the formal ontology. To evaluate the formal ontology by populating facts and making queries.

1.3 Hypothesis

The hypothesis of this research is that

1. A formal assembly reference ontology can support interoperability across both the assembly design and assembly process planning domains.

2. The assembly reference ontology concepts can be specialized into specific domain concepts to represent assembly design and assembly process planning knowledge consequently enabling knowledge sharing across the assembly design and assembly process planning domain systems.

Figure 1.1 shows the visual illustration of the research hypothesis proposed. The formal or heavyweight reference ontology refers to the ontology formed by formally defining and relating a set of assembly reference concepts (more details about heavyweight ontologies are provided in section 2.3.2.2).

Concepts associated with the assembly domain may have different implications in assembly design and assembly process planning domains; as such the concepts used by one domain are less likely to be understood by the other due to different understandings and perspectives of terms used within these domains. Therefore it requires intermediate concepts which can capture the related knowledge and are understandable across multiple domains. In this research these intermediate concepts are termed as reference concepts and they are relatively more generic as compared to the localized assembly design and assembly process planning domains system concepts.

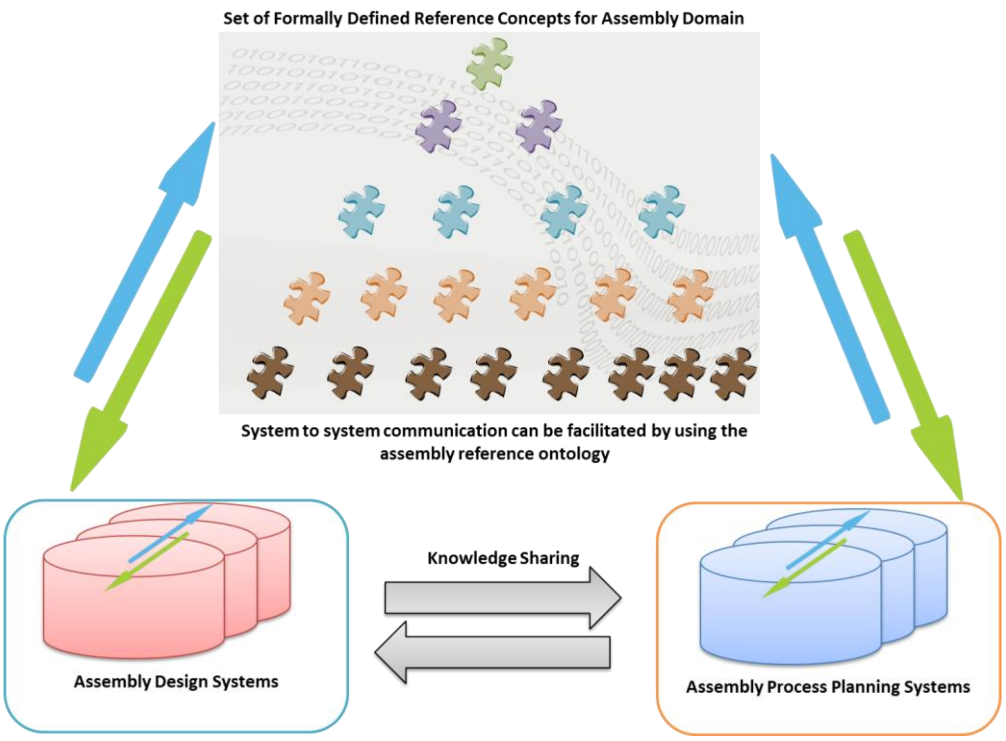

Set of Formally Defined Reference Concepts for Assembly Domain
System to system communication can be facilitated by using the assembly reference ontology
Knowledge Sharing
Assembly Design Systems
Assembly Process Planning Systems

Figure 1.1: Illustration of proposed research hypothesis

The assembly reference concepts can be specialized into multiple assembly design and assembly process planning domain specific concepts. These domain specific concepts can be related with each other using the assembly reference concepts. The formal representation of assembly domain concepts allow the capture of constrained meanings and the inference of new knowledge which enable knowledge sharing across the assembly design and assembly process planning domain systems.

1.4 Scope of the Research

The framework proposed in this research can be applied to range of assembly domain scenarios. More specifically the scope of this research is mainly focussed on addressing the interoperability issues across assembly design and assembly process planning domains. Due to substantial depth of interoperability issues in assembly design and assembly process planning domains, particular examples have been chosen to explore the representation and sharing of assembly knowledge related to Bill of Materials (BOM), tolerance and fits and their implications on the assembly process planning domain. In addition, this research work also explores a case study which investigates the effect of product design change onto the assembly process planning domain.

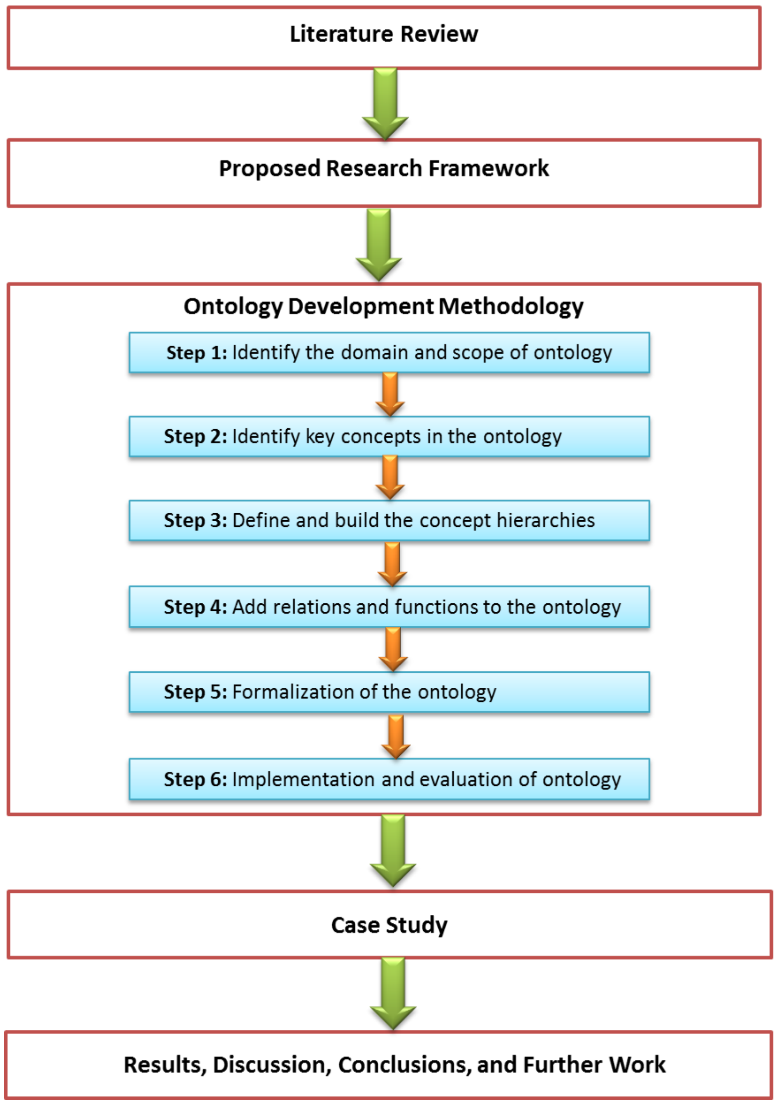
This research uses various ontology development languages and tools to realise the proposed research framework. In this research, Knowledge Frame Language (KFL) has been used to formally represent the assembly domain knowledge. Notepad++ a free source code editor (Ho, 2011) has been used to write the KFL code whereas Integrated Ontology Development Environment (IODE) has been exploited to test and evaluate the KFL code. In addition Unified Modelling Language (UML) has been used as system design tool for the lightweight representation of reference concepts and their relationships. Enterprise Architect (EA) has been exploited as a modelling tool to create the UML diagrams.

1.5 Research Methodology

The purpose of this research is to develop a method to support knowledge sharing across multiple assembly domains. The research methodology for this work takes the view from the objectives listed in section 1.2. The main components of the research methodology are shown in figure 1.2. The methodology starts from the review of existing literature which helps to understand the related research and to identify the potential research issues and research gaps. This forms the basis to propose a novel research framework (for further details please see chapter 3).

The proposed research framework is based on the idea of reference ontologies therefore a methodology for ontology development is laid out as shown in figure

1.2. This methodology has been adapted from the ontology development methodologies of Uschold and King (1995) (explained on page 25) and Noy and McGuinness (2001) (explained on page 30). The first step in the methodology is to outline the purpose, scope, intended uses, and potential questions which the ontology requires to answer. The second step in the ontology development methodology is to list down the concepts which fall within the scope of intended ontology. For this purpose, concepts from existing ontologies can be reused if they are related to the ontology to be modelled.



Literature Review
Proposed Research Framework
Ontology Development Methodology
Step 1: Identify the domain and scope of ontology
Step 2: Identify key concepts in the ontology
Step 3: Define and build the concept hierarchies
Step 4: Add relations and functions to the ontology
Step 5: Formalization of the ontology
Step 6: Implementation and evaluation of ontology
Case Study
Results, Discussion, Conclusions, and Further Work

Figure 1.2: Research methodology

In the next step concept hierarchies are built. This involves the specialization and/or generalization of the selected set of concepts. The concepts are then linked with each other via relationships and ontological functions (which are special case of relations (Chungoora, 2010)) in the next step. It is worth noticing that step 2, 3 and 4 can be done as parallel activities.

UML can be used to visually represent the concepts and the relationships between these concepts. The UML representation provides support to ontology modeller by acting as an intuitive design tool for ontology development and most importantly it has been used as design tool for Common Logic (CL) based formal ontologies in the manufacturing sector (Palmer, et al., 2012).

In the next step, ontology is formalized using CL based KFL. The KFL allows capturing domain semantics by applying constraints and rules which form the basis of heavyweight ontology. The formalized ontology is then implemented in IODE (a CL based ontology development tool). The implemented ontology is then evaluated by asserting the instances and making queries. More detail about the IODE can be found in appendix A.

Finally a case study is explored to verify the proposed research framework. The results, discussion, conclusions and future work are then drawn to sum up the research work in this thesis.

1.6 Thesis Structure

This chapter follows the review of existing literature described in chapter 2. Chapter 2 provides an initial understanding of the relevant existing research and identify the potential research gap. Chapter 3 describes the novel research concept in the form of Assembly Reference Ontology (ARO) and briefly discusses its main points. Chapter 4 further explores the concepts identified in chapter 3 and presents a detailed overview of the ARO.

Chapter 5 specifically explores some of the assembly reference concepts and the latter are then formally captured as described in the research methodology. In chapter 6, a number of experiments have been analysed and a case study has been explored to verify the proof of the proposed approach. Finally chapter 7 discusses the research findings, conclusions and possible further extensions of this research work.

CHAPTER 2

BACKGROUND LITERATURE REVIEW

2.1 Introduction

This chapter presents a comprehensive background literature review on potential research areas and the related issues. The review of existing research provides an exposure to the research work undertaken in different directions to date and explores the potential research gaps and opportunities in line with the targets of this research. The chapter is organised as follows.

Section 2.2 discusses the concepts of knowledge sharing and interoperability and highlights potential research issues related to these areas. Section 2.3 then explains the ontology concept and its classifications. There are various ontology development methodologies, languages and tools which are currently being used as support to develop ontologies, are described in section 2.4.

Section 2.5 is dedicated to providing an overview of the existing work on ontologies in the manufacturing domain. Section 2.6 discusses the concepts related to manufacturing assembly. This section describes the concept of assembly, its importance, potential research avenues, role of features in assembly and also provides an overview of the existing ontologies in the assembly domain. Finally, a summary of the whole chapter is presented in section 2.7.

2.2 Knowledge Sharing and Interoperability

2.2.1 Knowledge Sharing

Lee (2001) defines knowledge sharing as "*activities of transferring or disseminating from one person, group or organization to another*". Other researchers have defined knowledge sharing on similar lines. For example, Hooff and Ridder (2004) define knowledge sharing as "*the process where individuals mutually exchange their knowledge and jointly create new knowledge*". Tsui et al. (2006) gave a more generic definition of knowledge sharing as they describe it as the process of exchanging knowledge among different individuals. Hendriks (1999) believes that Knowledge sharing assumes relationship between at least two parties, the one which possesses knowledge and the other which acquires knowledge. These parties may refer to people, organizations or software tools.

Knowledge sharing has been considered as an important factor to improve the business performance of the companies (Huang et al., 2010) (Riege, 2005). Particularly the role of knowledge sharing is critical for the manufacturing companies to remain competitive in the market (Fathi et al., 2011). However the potential benefits that knowledge sharing contributes to support the product development have not been yet completely understood (Hong et al., 2004).

Within the scope of manufacturing sector, knowledge sharing across the manufacturing functions has been considered a key research issue (Oztemel and Tekez, 2009). Furthermore knowledge sharing in cross disciplinary teams across the organization is not a straightforward task (Young et al., 2007) and requires effective mechanisms to support knowledge sharing.

For instance knowledge sharing difficulties between individuals across different departments in an organization are caused by difference in their languages, their work contexts and understanding of the product from their own perspectives (Bechky, 2003) (Riege, 2005). Moreover the issues related to knowledge sharing between the software tools (in comparison to humans) are more complex however these tools carry potential benefits if they are able to support knowledge sharing (Young et al., 2005).

Knowledge sharing between the product design and manufacturing is very important because of the fact that product design has massive influence on subsequent stages of product lifecycle. For instance, True and Izzi (2002) report that product design can impact up to 70% of product lifecycle costs. Although Barton et al. (2001) have challenged this figure however they support the fact that product design decides majority of the product lifecycle costs.

Designers are provided with product related information however they also require information other than they are provided with, and spend sufficient amount of time on searching and managing recently updated information (Kuffner and Ullman, 1991). This additional information may be related to design domain or other product lifecycle domains especially the manufacturing domain. Furthermore manufacturing knowledge supports the product design related decisions (Wang and Tong, 2008), therefore designers should have an easy access to product manufacturing information.

This triggers the demand for the development of mechanisms to support the capture and sharing of information within the product design domain as well as across the product lifecycle domains including the manufacturing. The capture and sharing of domain knowledge requires means of knowledge representation that should enable the definition and meaning of the content of information (Kryssanov et al., 2006). In Information and Communications Technology (ICT) based context, information is exchanged between software systems which in turns leads to the concept of interoperability and is discussed in the next section.

2.2.2 Interoperability

The word interoperability is derived from the word "interoperable" and Oxford dictionary states that computer systems are interoperable with each other if they are "*able to exchange information and make use of information*". This suggests that interoperability is the ability of computer systems to exchange as well as understand the information.

There are various other definitions of interoperability found in literature. For example, Woodley (2005) defines interoperability as "*The ability of different types of computers, networks, operating systems, and applications to work together effectively without prior communication, in order to exchange information in a useful and meaningful manner*". A more relevant definition to this work is provided by Chen and Vernadat (2004) who define interoperability as "*the ability of two or more systems or components to exchange and use shared information*". A similar definition is also given in IEEE standard computer dictionary (1991) where interoperability has been described as "*the ability of two or more systems or components to exchange information and to use the information that has been exchanged*".

Chungoora (2010) extended the definition of interoperability for product design and manufacture domains and describe it as the ability of knowledge base systems to seamlessly exchange design and manufacturing related information across these domains. However interoperability across multiple, heterogeneous, and autonomous systems is a challenging task for the modern organizations (Panetto, 2007) (Castano and Antonellis, 1997) and is a common issue for many applications (Hardwick et al., 1996) including the design and manufacture.

These interoperability issues in turn costs huge amount of money. For example, a study conducted by NIST (1999) reveals that imperfect interoperability causes around one billion dollars annually to US automotive industry and majority of these costs are incurred on repairing or resending the information which was not exchanged properly.

Interoperability issues are caused by different reasons however the most common reason is the syntactic and semantic incompatibilities of the information to be shared (Das et al., 2007). Syntactic incompatibilities are instigated due to software systems using different information representation structures whereas semantic incompatibilities are caused due to the lack of clearly defined semantics of the information to be shared (Chen, 2006).

In the literature, researchers have mainly emphasized on resolving the semantic issues. For instance, Chungoora (2010) has found that there exists a potential gap to resolve semantic interoperability issues and has emphasized the need to investigate these issues. Chen and Vernadat (2004) say that interoperability issues can be addressed by establishing "*common or equivalent*" semantics. Furthermore, Chungoora et al. (2012) argued that interoperability across the product lifecycle domains can be achieved by rigorously defining the meaning of PLM system concepts.

Potential methods which are currently being explored to achieve interoperability are: model driven interoperability, standard based approaches to interoperability and ontology based interoperability. The Model Driven Interoperability (MDI) is based on the Model Driven Architecture (MDA). MDA is a framework introduced by the Object Management Group (OMG) (http://www.omg.org/mda/) and is based on various OMG standards. MDA supports creation of highly abstract, machine readable models (Kleppe et al., 2003) which can then be transformed into domain specific models (MDA-Guide-Document, 2003) to support interoperability.

However MDA lacks the ability to unambiguously specify the domain concepts which is a requirement for semantic interoperability (Komatsoulis et al., 2008). MDA also does not support reasoning and querying about the system structure and its components (Usman, 2012).

In addition to the above mentioned approach, efforts have also been made to use standards to promote interoperability. One such standard is ISO 10303 which is also known as STandard for the Exchange of Product model data (STEP) (Pratt, 2001). STEP provides standard neutral representation of product data in computer understandable form throughout the product lifecycle (SCRA, 2006) thus supporting the interoperability across product lifecycle systems (Saaksvuori and Immonen, 2008). STEP consists of domain specific Application Protocols (AP) which makes it more manageable from implementation point of view (SCRA, 2006). Currently AP 203 is the most widely used AP which mainly deals with the product assembly related information (Pratt, 2001).

Despite these standardization efforts to support interoperability, researchers have found potential issues in standard based interoperability approach. For instance, Newman et al. (2008) argue that although this approach supports interoperability in manufacturing systems however a potential barrier to the development of these standard based interoperable systems is the resistance from software/hardware vendors who exploits opportunity of lack of standards. This argument is further supported by Young et al. (2009) who claim that implementation of such standards requires consensus from users to commit one standard way of information representation which they say, has not been successful over the time due to the lack of flexibility.

Researchers have found that even if the users of the information systems agree on a specific standard, interoperability issues will remain. For instance, Ray and Jones (2006) say that communities can agree on the standardization of domain terms however interoperability issues will exist because of the different understanding of the meanings of these terms. Their argument is further endorsed by Young et al. (2007) who maintain that despite the implementation of a specific standard, semantic conflicts could still exist because of the lack of rigorous definition of the domain concepts.

The potential reasons of above mentioned standard based interoperability issues is that currently the concepts defined in ISO standards are based on textual definitions (Michel, 2005) which could have multiple interpretations. Moreover, the lack of formal definitions of these concepts results in poor interoperability across the participating computer systems because of the ambiguities involved in their meanings (Chungoora et al., 2013).

In context of assembly multiple viewpoints may exist which can consequently cause interoperability issues. For example the bearing and shaft assembly shown in figure 2.1 has assembly design and assembly process planning viewpoints. The assembly design viewpoint considers the functional, material, tolerance and fits related information. However the assembly process planning viewpoint reflects the assembly process and resource related information. In a situation where these viewpoints are captured in two different knowledge base systems, the interoperability across such systems will be difficult to achieve.

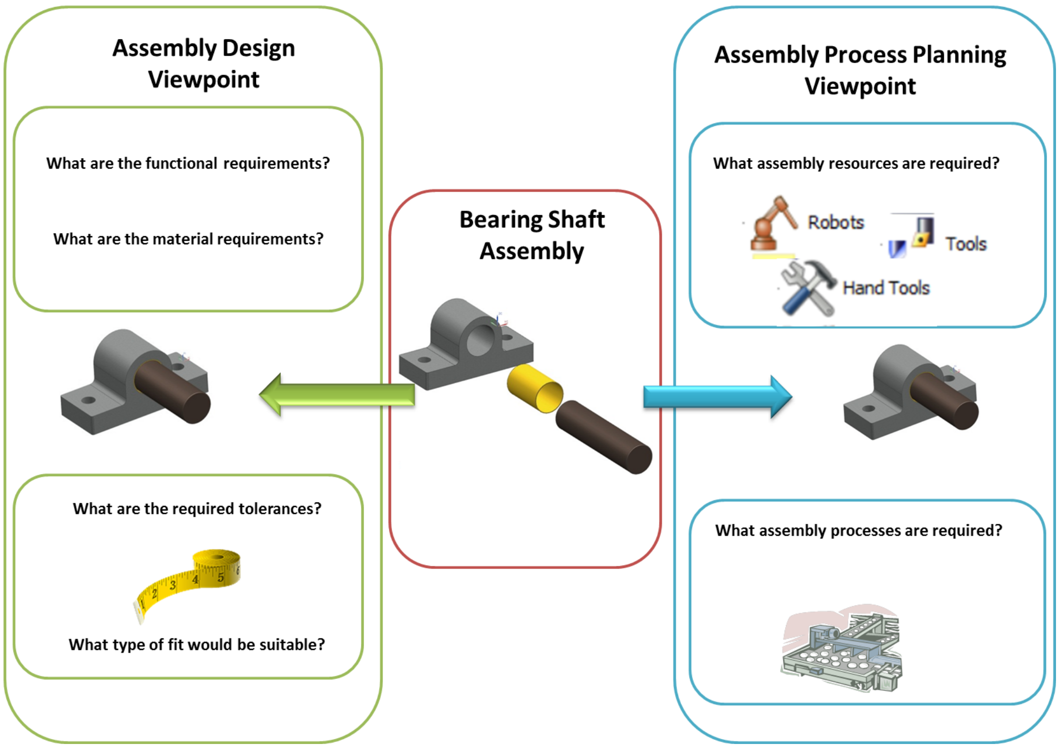

Assembly Design Viewpoint
What are the functional requirements?
What are the material requirements?
What are the required tolerances?
What type of fit would be suitable?
Bearing Shaft Assembly
Assembly Process Planning Viewpoint
What assembly resources are required?
Robots
Tools
Hand Tools
What assembly processes are required?

Figure 2.1: Implications of bearing shaft assembly in assembly design and assembly process planning.

A possible solution to this problem is to use ontologies to rigorously define the semantics of assembly related concepts. An assembly reference ontology can be used to support interoperability across multiple domains. More specifically, formally defined concepts in the assembly reference ontology can support knowledge sharing across assembly design and assembly process planning domains. For instance, the concepts of assembly feature, dimension and tolerance (when formally defined) can be used to capture limits and fits related assembly design information. However these concepts are comparatively generic and may need to be linked with more specialized domain specific assembly design concepts. The examples of such domain specific concepts may be different types of BS 4500 tolerance grades such as H7, H8, f7, k6 as shown in figure 2.2. The assembly design related knowledge (e.g. type of fits, allowance etc.) can be captured using these concepts and by applying the axioms.

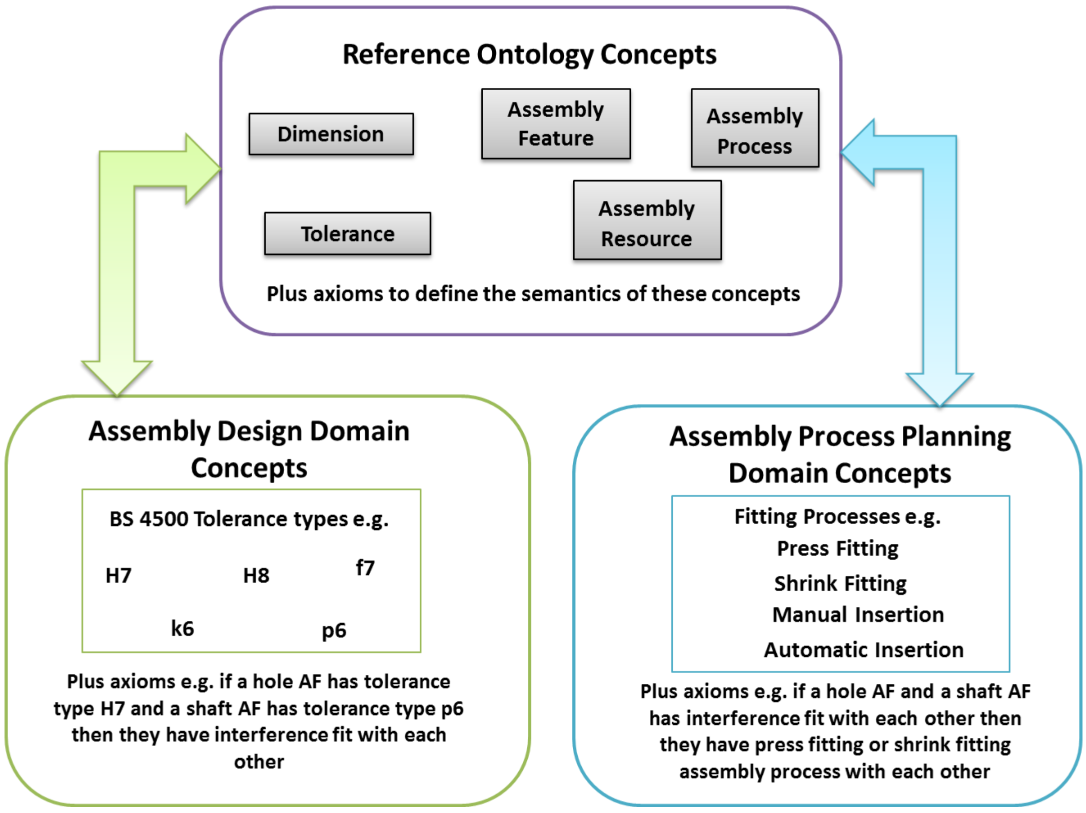

Reference Ontology Concepts
Dimension
Assembly Feature
Assembly Process
Tolerance
Assembly Resource
Plus axioms to define the semantics of these concepts
Assembly Design Domain Concepts
BS 4500 Tolerance types e.g.
H7
H8
f7
k6
p6
Plus axioms e.g. if a hole AF has tolerance type H7 and a shaft AF has tolerance type p6 then they have interference fit with each other
Assembly Process Planning Domain Concepts
Fitting Processes e.g.
Press Fitting
Shrink Fitting
Manual Insertion
Automatic Insertion
Plus axioms e.g. if a hole AF and a shaft AF has interference fit with each other then they have press fitting or shrink fitting assembly process with each other

Figure 2.2: Example of knowledge sharing across assembly design and assembly process planning

For example if the hole assembly feature (AF) has tolerance type H7 and shaft assembly feature (AF) has tolerance type p6 then the resulting type of fit would be interference fit. This is because the maximum allowable dimension of the hole AF having H7 tolerance is always less than the minimum allowable dimension of the shaft AF having p6 tolerance (more detail can be found in section 5.3).

Similarly the reference concepts assembly feature, assembly process and assembly resource (shown in figure 2.2) can be used to determine the assembly process planning related knowledge. For instance, the types of assembly processes can be determined for a specific type of fit. Because the concept assembly process is comparatively generic as compared to domain specific assembly process planning concepts such as press fitting, shrinking fitting, therefore these domain specific concepts can be linked with the assembly process concept. Axioms can be applied to infer this kind of information. For example, when a hole AF has interference fit with a shaft AF, the resulting process would be either press fitting or shrink fitting as displayed in figure 2.2 (more detail can be found in section 5.3). In this way the assembly process planning knowledge can be shared with the assembly design knowledge and vice versa.

The next section discusses the concept of ontologies in detail.

2.3 Ontology Based Interoperability

Semantic interoperability across heterogeneous systems can be achieved if the meanings of the information to be shared are well understood across these systems (Wache et al., 2001). Ontologies have the potential to provide semantic base (Young et al., 2005) for such systems thus supporting interoperability across these systems. The following sections summarise some of the ontology definitions and classifications in relation to this thesis work.

2.3.1 Ontology Definitions

The term ontology has been borrowed from the field of philosophy where it is described as the systematic account of existence (Ciocoiu et al., 2001). In ICT based context, an ontology is the representation of domain information (Chandrasekaran et al., 1999) and more precisely it defines "*the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary*" (Neches et al., 1991). However perhaps the most referred definition of ontology in the literature is provided by Gruber (1993) who defines ontology as "*an explicit specification of a conceptualisation*" where conceptualisation is an abstract and simplified view of the universe of discourse.

Borst (1997) argues that there should be a consensus on conceptualisation to support ontology reuse and his definition of ontology is a variation of Gruber (1993)'s definition. He defines ontology as "*a formal specification of a shared conceptualisation*". Studer et al. (1998) modified the definition of ontology by combining the definitions of Gruber (1993) and Borst (1997). They define ontology as "*a formal, explicit specification of a shared conceptualisation*". They further explain the terms used in the definition of ontology which are: formal, explicit, shared and conceptualisation. According to them, the term "formal" refers to machine readable, "explicit" suggests that concepts and their constraints are explicitly defined, "shared" means agreed or consensual knowledge and "conceptualisation" reflects an abstract model of the world.

Many other researchers have defined ontology from their own perspectives e.g. more ontology definitions can be found in (Guarino and Giaretta, 1995) (Schreiber et al., 1995) (Heijst et al., 1996) (Guarino, 1997) and (Gruninger et al., 2001).

However a more relevant definition of ontology to this work is given in (ISO-18629, 2005) where it is described as "*a lexicon of specialised terminology along with some specification of the meaning of the terms in the lexicon*". This definition leads to the emphasis that an ontology describes a set of concepts with their meaning defined by axioms that provide a basis for shared meaning (Young et al., 2007). The axioms help to restrict the interpretation of domain concepts and support inference of new knowledge and are basic building blocks of heavyweight ontologies (discussed in 2.3.2.2.).

2.3.2 Classification of Ontologies

Several types of classifications of ontologies are found in literature which have been summarised in (Zhou and Dieng-Kuntz, 2004). However in relation to this research work, two main ontology classifications are explained in the following sections.

2.3.2.1 Foundation, Domain and Reference Ontologies

Foundation ontologies sometimes known as upper ontologies (FinES-Cluster, 2011) consist of generic, abstract and high level concepts which can be applied to a wide range of domains (Sanchez-Alonso and Garcia-Barriocanal, 2006). Foundation ontologies provide a knowledge base to more specialized ontologies and are comprised of formally defined concepts (Sanchez-Alonso and Garcia-Barriocanal, 2006).

Examples of key foundation ontologies are: Basic Formal Ontology (BFO), Cyc's upper ontology, Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), General Formal Ontology (GFO), Highfleet's Upper Level Ontology (ULO), Open Knowledge Base Connectivity (OKBC) ontology, Suggested Upper Merged Ontology (SUMO), and the Object-Centred High-level Reference Ontology (OCHRE) (Borgo and Leitao, 2007) (FinES-Cluster, 2011). The concepts related to Highfleet's ULO are further explained in section 4.2 of chapter 4.

Domain or application ontologies comprise of formally defined concepts and relationships intended to represent an application area (Musen, 1998) (Jean et al., 2006), and are hardly used outside a particular research environment they are designed for (Navigli and Velardi, 2004). An example of foundation and domain ontologies based on Uschold and Gruninger (2004)'s work is shown in figure 2.3. The concepts like "things", "individuals" are part of the foundation ontology and they can be applicable to any domain. However the domain concepts like "pump", "engine" etc. belong to a specific domain and they cannot represent domains other than their own.

Sabou et al. (2005) introduced the term quality domain ontology and argue that it can cover a wide range of a domain's terminology. This argument leads to the fact that there is a type of ontology which sits in between the foundation and domain ontologies as described by Navigli and Velardi (2004). This type of ontology is called reference ontology. The term reference ontology was first introduced by Nicola Guarino in 26th German conference on artificial intelligence held in Hamburg in 2003 (Grenon, 2003). Guarino found that the reference ontology aims to "*clarify the meanings of terms used in a specific domain*" (Grenon, 2003).

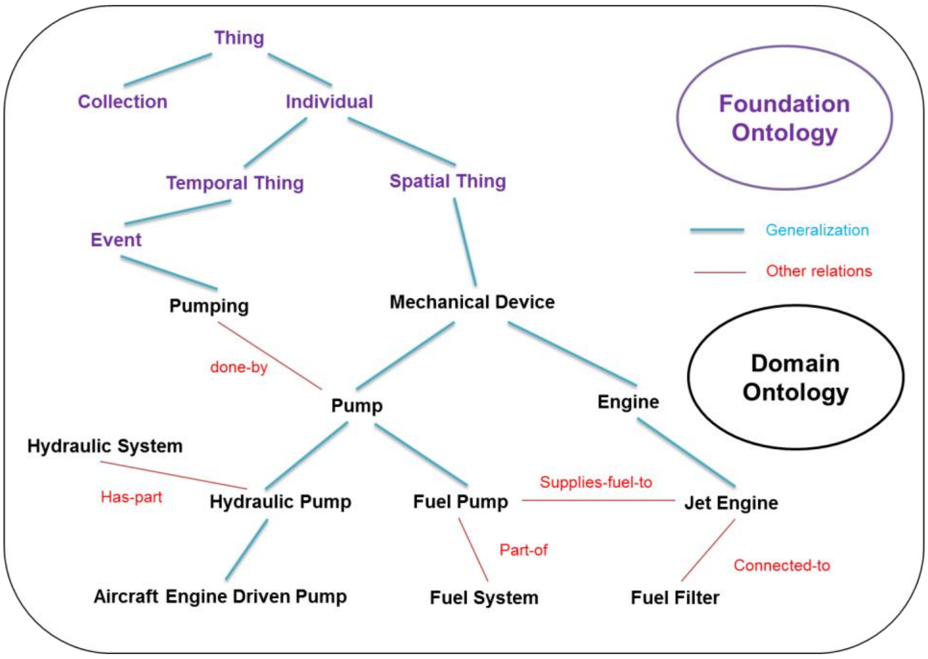


Figure 2.3: Examples of foundation and domain ontologies (Uschold and Gruninger, 2004)

Burgun (2006) describes reference ontologies as a way to represent domain knowledge without focussing on specific objectives. Leila (2009) summarises the definitions of reference ontology and described it as an ontology which represents a domain adequately and is validated by majority of the domain experts. He further argues that reference ontologies tend to be broad, satisfy needs of large community of domain, support shared meanings, use axioms, and can be derived from the foundation ontology.

Reference ontologies are comparatively new development and are emerging as potential candidates for the representation of domain knowledge in a way that they can be re-used in different ways (Brinkley et al., 2006). Recently a few reference ontologies have been developed in the field of medicine (Burgun, 2006), however despite having potential to support knowledge sharing, they have not yet got wide spread applications in other domains like manufacturing and assembly. Usman et al. (2013) has proposed a reference ontology in the field of manufacturing however it deals with single piece part manufacturing only and therefore cannot be used for the assembly domain. This thesis aims towards the creation of assembly reference ontology which is further explained in chapter 3.

In this work, the assembly reference ontology is based on Highfleet's foundation ontology and provides a common semantic base for domain specific assembly design and assembly process planning ontologies. An example of foundation ontology concept is "quantity" which represents measurement related information. This could be weight, dimension or anything which is specified by measurements. The related assembly reference ontology concept is "tolerance" (shown in figure 2.2 in section 2.2.2) which represents the tolerance information and is specified in length units e.g. mm, cm etc. Similarly the domain specific concepts which can be specialized from the tolerance concepts are the BS 4500 standard tolerance grades or types e.g. H7, H8, f7, k6 (shown in figure 2.2 in section 2.2.2) which are applicable to hole and shaft type assembly components only.

2.3.2.2 Lightweight and Heavyweight Ontologies

Gomez-Perez et al. (2004) report that the ontology community identifies two types of ontologies from the perspective of their ability to represent domain knowledge. They are called: lightweight and heavyweight ontologies. Lightweight ontologies consist of concepts, taxonomies and simple relationships while heavyweight ontologies are one step further to lightweight ontologies as they contain axioms in addition to the lightweight ontologies (Gomez-Perez et al., 2004). A structure of heavyweight ontology is displayed in figure 2.4 where the top layer shows axioms in addition to the lightweight ontologies.

Lightweight ontologies are easy to create as compared to heavyweight ontologies (Zhu and Madnick, 2006) however they are unable to convey meanings and interpretations of domain concepts (Oberle et al., 2009). In contrast to lightweight ontologies, heavyweight ontologies are difficult to deploy (Zhu and Madnick, 2006) however they have the potential to rigorously define the domain concepts (Uschold and Gruninger, 2004) and thus promote shared meaning from across the heterogeneous systems (Young et al., 2007). This thesis uses the heavyweight ontological approach and investigates its ability to capture and share knowledge in the assembly domain.

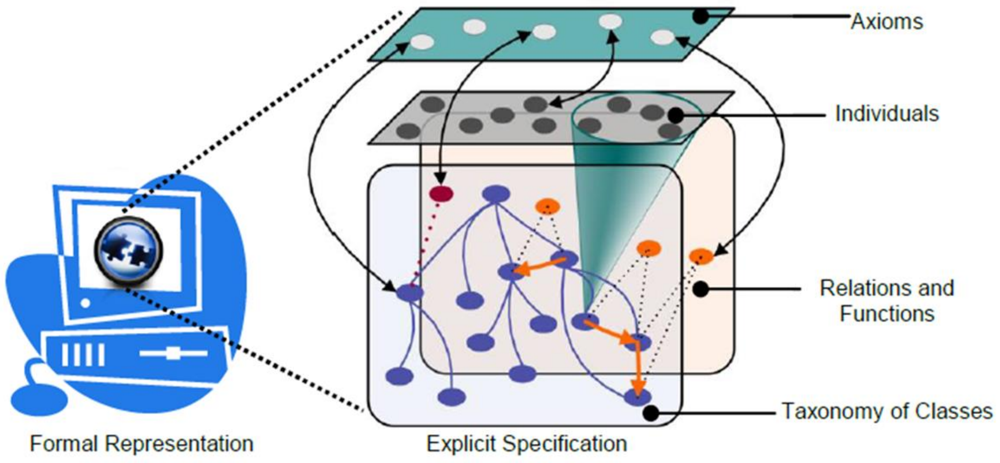


Figure 2.4: Structure of heavyweight ontology (Chungoora, 2010)

2.4 Development of Ontologies

There are three main pillars for the development of ontologies which are: ontology development methodologies, ontology development languages, and ontology development tools. A methodology enumerates the necessary steps required to develop an ontology. Two important steps in the methodology are the ontology representation in a formal language and the implementation of ontology in an ontology development environment. Hence the ontology development languages and tools are also required to develop the ontology. Key ontology development methodologies, languages and tools are explained in the following sections.

2.4.1 Methodologies

Ontology development methodologies describe the steps required to develop an ontology (Usman, 2012). Various ontology development methodologies have been proposed by the ontologists over the years however some of those widely reported methodologies are discussed in this thesis.

2.4.1.1 Uschold and King Methodology

This methodology was proposed by Uschold and King (1995) and has also been used in this research as described in section 1.5. The main steps involved in this methodology are (1) Identification of purpose, (2) Building Ontology, (3) Evaluation and (4) Documentation. The first step emphasizes to identify the potential purpose of the ontology to be developed and range of its intended users.

The second step: building ontology consists of further three sub-steps which include (i) Ontology Capture, (ii) Coding and (iii) Integrating Existing Ontologies. In this methodology “ontology capture” refers to the identification of main concepts and relationships, textual definitions of these concepts and relationships, and identifying the terms for these concepts and relationships.

The coding refers to the explicit representation of the concepts and relationships (captured in the ontology capture) in a formal language. Coding requires selection of an appropriate representation language to create the code for the ontology. Finally "Integrating Existing Ontologies" means whether the existing ontologies can be used during ontology capture and coding processes.

The third step evaluation requires the assessment of ontologies against a frame of reference e.g. the requirement specifications and then adapting the ontologies accordingly. Finally the fourth step aims at developing an adequate documentation to support knowledge sharing.

2.4.1.2 Gruninger and Fox Methodology

This ontology development methodology was proposed by Gruninger and Fox (1995) and consists of following six steps.

- Motivating Scenario
- Informal Competency Questions
- First Order Logic: Terminology
- Formal Competency Questions
- First Order Logic: Axioms
- Completeness Theorem

2.4.1.2.1 Motivating Scenario

The need to develop ontologies arises from the motivating scenarios particularly drawn by the industrial problems. These industrial problems normally exist in the form of story problems or examples which were not properly dealt with by the existing ontologies. Hence motivating scenario form the first step to create new ontology or extending the existing ontology.

2.4.1.2.2 Informal Competency Questions

The informal competency questions are a set of queries (triggered by the motivating scenario) which need to be answered by the new ontology. The term informal suggests that these queries/questions have not been yet represented in a formal ontology language. The relationship between the motivating scenario and the informal competency questions help in evaluating the new/extended ontology. Based on this evaluation the need for the new ontology or extension ontology can be determined.

2.4.1.2.3 First Order Logic: Terminology

Once the informal competency questions have been proposed the terminology of ontology should be expressed in first-order logic. This terminology of ontology results from the previous step when competency questions/queries were proposed for a new or extended ontology. The very first step in formally specifying the ontology terminology is to identify the objects in the domain of interest. These objects are described by variables and constants in the ontology language and subsequently relations between these objects can be defined.

2.4.1.2.4 Formal Competency Questions

After the competency questions are informally defined and terminology of ontology is defined, the competency questions are formally defined. It is important to realize that the terminology of the ontology should have all the terms used in the formal competency questions. The formal competency questions help to evaluate the proposed new or extension ontologies.

2.4.1.2.5 First Order Logic: Axioms

The first order logic axioms define ontological terms and helps to apply constraints on these terms. Axioms constitute an essential part of the ontology and describe semantics of the terms used in first order logic. Defining axioms is the most difficult part of defining ontologies however the formal competency questions help to specify these axioms. The expressiveness of axioms is determined by their ability to represent competency questions e.g. if a set of axioms completely represent competency question as compared to the other one then we can say that first set of axioms is more expressive than the latter one.

2.4.1.2.6 Completeness Theorem

Once the competency questions are formally defined, the conditions which fulfil the solutions of competency questions are specified.

2.4.1.3 METHONTOLOGY

METHONTOLOGY a methodology for ontology development was proposed by Ferndndez et al. (1997) and was developed in Artificial Intelligence Lab (Ontology Engineering Group) at Technical University of Madrid. This methodology was created in the domain of chemicals however it can be used as reference for other domains as well. Main steps in this methodology are as follows:

- Specification
- Knowledge Acquisition
- Conceptualisation
- Integration
- Implementation
- Evaluation
- Documentation

2.4.1.3.1 Specification

The aim of this step is to create a specification document expressed in natural language using the competency questions or by using a set of intermediate concept representations. This methodology recommends that the specification should include the purpose of ontology, the level of formality required, and the scope of the ontology. The ontology specification document should have conciseness, partial completeness and consistency in it.

2.4.1.3.2 Knowledge Acquisition

The knowledge acquisition phase runs parallel to the specification phase where all the relevant information is acquired using the expert guidance, books, figures, and by consulting similar ontologies. Relevant information from these resources is elucidated by using methods like brainstorming, interviews and knowledge acquisition tools. In METHONTOLOGY, the ontology developers have used a range of techniques during knowledge acquisition phase which include interviews with experts, informal and formal text analysis.

2.4.1.3.3 Conceptualisation

In conceptualisation phase, domain knowledge is structured using the domain vocabulary identified in the specification phase. A Glossary of Terms (GT) that includes concepts, and verbs, is developed to capture all the applicable domain knowledge with its meanings. Part of the GT is identified from the specification document while others are identified as the ontology development process progresses. On the basis of concepts and verbs rules are built which collects the domain knowledge.

2.4.1.3.4 Integration

Integration refers to reusing and interlinking the terms used in current ontology with the existing ontologies to expedite ontology development process. The existing meta-ontologies must be explored to align the definitions used in current ontology. For this purpose, the ontology developers should explore relevant libraries of ontologies which provide coherent semantics.

2.4.1.3.5 Implementation

The ontology implementation phase requires ontology development environment which can support the formal ontology. The ontology development environment should be able, to display lexical and syntactic errors, to provide an editor to modify the definitions, a browser to look for library of the ontology and other similar functions to facilitate the implementation process.

2.4.1.3.6 Evaluation

The purpose of evaluation phase in METHONTOLOGY is to make "*technical judgement of ontologies, their software environments and documentations*" with reference to the specification document. The output of this phase entails various evaluation documents listing the techniques of evaluation and the errors found during each step of this methodology.

2.4.1.3.7 Documentation

This phase requires the documentation of the developed ontology. METHONTOLOGY necessitates the documentation phase in the ontology development cycle because of the fact that no consensual guidelines are available to facilitate the developers in documenting the ontology development process.

2.4.1.4 Noy and McGuinness Methodology

This methodology was proposed by Noy and McGuinness (2001) and has also been used in this research as mentioned in section 1.5. It consists of following steps:

- Determine the Domain and Scope of the Ontology
- Consider Reusing Existing Ontologies
- Enumerate Important Terms in the Ontology
- Define the Classes and the Class Hierarchy
- Define the Relations and Functions
- Create Instances

2.4.1.4.1 Determine the Domain and Scope of the Ontology

The first step in this methodology is to define the domain and scope of the ontology. A list of competency questions can be prepared for this purpose. These questions can relate to the domain of interest, the purpose of ontology, and the type of queries the intended ontology should answer.

2.4.1.4.2 Consider Reusing Existing Ontologies

Step 2 in this methodology suggests reuse of existing ontologies. Many of the existing ontologies are in electronic form and can be imported in the ontology development environment.

2.4.1.4.3 Enumerate Important Terms in the Ontology

The third step in this methodology requires listing down of all the related terms for the domain of interest. Sometimes these terms overlap however the overlapping terms can be sorted out in the later stages as well.

2.4.1.4.4 Define the Classes and the Class Hierarchy

Once the important terms are listed, the next step is to define the classes and their hierarchies. Top-down, bottom-up and combination (of top-down and bottom up) approaches can be used to develop the class hierarchies.

2.4.1.4.5 Define the Relations and Functions

Once the class and class hierarchy is defined, the next step is to define relations and functions. The relations and functions help to add more information to classes. The cardinality of relations and functions is also defined to specify the order of relations and functions i.e. unary, binary etc.

2.4.1.4.6 Create Instances

The last step in this ontology development methodology is to create instances of the classes defined in the ontology.

2.4.2 Languages

Ontology development languages provide a representation of the internal structure of an ontology. A set of key ontology development languages are reviewed in the following sections.

2.4.2.1 RDF and RDF (Schema)

Resource Description Framework (RDF) is a language developed by W3C. RDF provides a basis for "processing meta-data" and can support interoperability across a range of applications to share machine readable information on the web (Lassila and Swick, 1999). RDF is an object-attribute-value triple (Decker et al., 2000) that can be represented as A (O, V) which means that an object O has an attribute A which has a value V (Decker et al., 2000). However RDF is limited because it represents objects with named attributes and values only (Lam et al., 2008).

RDF (Schema) "*is a semantic extension of RDF*" (Brickley and Guha, 2004) which provides built-in classes and sub-classes to represent the domain semantics (Mizoguchi, 2004). RDF (Schema) supports RDF by providing a set of pre-defined classes and properties however it can only help to define simple ontologies (Pan, 2009).

2.4.2.2 Web Ontology Language (OWL)

The OWL has been specially designed to process the content of information as opposed to the situations where it is required to just present the information to humans (W3C, 2004). OWL provides better interoperability to web content as compared to RDF and RDF Schema by providing supplementary vocabulary with formal semantics (W3C, 2004). OWL has been developed to accommodate the limitations of RDF and RDF Schema. For instance, RDF and RDF Schema cannot support the representation of disjointness of classes, cardinality constraints, special characteristics of relations e.g. transitive relation, and constraining the relations for limited set of classes (Antoniou and Harmelen, 2009).

OWL is considered highly expressive language which can also support many reasoning services (Sengupta and Hitzler, 2013). OWL semantics are based on the Open World Assumption (OWA) that follows the assumption that things which are not known to be true may not be necessarily false (Palmer et al., 2012) (Sengupta and Hitzler, 2013). However the domains like manufacturing and assembly are facts driven and need certainty that can be supported by the closed world assumption (Palmer et al., 2012).

OWL is also limited in representing the relations and functions. For instance, OWL cannot directly support relations having arity more than 2 and functions having arity more than 1 (Palmer et al., 2012). Another potential limitation of OWL is that it does not have conjunction, disjunction, and negation operators (Sengupta and Hitzler, 2013) which can limit its reasoning capabilities.

OWL has been classified into three sublanguages called OWL Lite, OWL DL, and OWL Full (W3C, 2004) (Antoniou and Harmelen, 2009). OWL Lite is aimed for applications requiring classification hierarchies and simple constraints. For example, it only supports cardinality constraints between 0 or 1 and, it excludes enumerated classes and disjointness statements. OWL DL supports applications requiring maximum expressiveness while retaining all conclusions computable in limited time. OWL DL has all OWL language constructs however they are used under certain constraints. OWL Full uses all of OWL language constructs and it aims for applications where maximum expressiveness is required. However it lacks computational completeness and decidability, as well as it lacks the support of a tool that would be able to support complete reasoning for all the OWL Full features.

2.4.2.3 Knowledge Interchange Format (KIF)

Knowledge Interchange Format (KIF) is a formal language developed to exchange knowledge across different computer systems (Genesereth et al., 1992). KIF has declarative semantics, is logically comprehensive, and supports the representation of knowledge about the knowledge (Genesereth et al., 1992). Ginsberg (1991) found that, efforts to standardise the knowledge representation has led to the development of an Interlingua or knowledge interchange format (KIF). KIF behaves like a mediator in translating from other languages to KIF and vice versa (Gasevic et al., 2006).

Corcho et al. (2002) describes KIF as the most expressive language for representing ontologies as it supports the representation of "*concepts, taxonomies of concepts, binary relations, functions, axioms, instances and procedures*". However they question the ability of KIF to develop reasoning mechanisms and suggest that KIF does not provide reasoning support due to its high expressiveness.

2.4.2.4 Frame Logic (FLogic)

Frame Logic or FLogic (Kifer et al., 1995) was developed in 1995 at the Department of Computer Science in State University of New York. FLogic is an extension of the first order backed up with the object oriented modelling (Bruijn, 2007). FLogic supports generalization/specialization of the concepts, capturing knowledge using rules, and retrieving knowledge by making queries (Angele et al., 2009). FLogic also allows deduction of new information and constraint checking (Corcho, et al., 2002) making it a suitable language for heavyweight ontological modelling.

The use of FLogic has been witnessed by a lot of commercial as well as open source academic systems and it is now broadly recognised for developing intelligent information systems (Angele et al., 2009). However there are still some limitations of FLogic. For example, relations having arity 2 or more are not directly supported by FLogic rather they are modelled by taking one argument at a time and the constraints on the relation arguments are specified by using axioms (Gomez-Perez et al., 2004).

2.4.2.5 Common Logic

Common Logic (CL) is a logic framework aimed for sharing and transmission of information (ISO/IEC-24707, 2007). CL is based on first order logic which is a foundation for knowledge representation (Nemuraite et al., 2009). CL supports integration and reuse of knowledge (Polovina et al., 2009) and offers potential benefits in comparison to other ontology representation languages. For example, CL is more expressive as compared to RDF, OWL-Lite, and OWL-DL and is computationally more powerful as compared to OWL-Full (Delugach, 2008) (Sánchez-Ruíz et al., 2009).

More specifically, in comparison to OWL, CL based formalism enjoys key potential benefits. For instance, in contrast to OWL, CL is based on Closed World Assumption (CWA) (Chungoora et al., 2013) whereas CWA assumes that everything stated or implied is true and everything else is false (Date, 2007). As described in section 2.4.2.2, the assembly domain is fact driven and requires certainty; therefore a CL based approach with CWA best suits for this domain. Other potential advantages of CL based approach (in contrast to OWL) is that it supports ternary relations (and relations having arity more than 3), binary functions (and functions having arity more than 2), conjunction, disjunction, and the negation operators (Palmer et al., 2012) which are potentially required for modelling complex domains like assembly. Research conducted by Chungoora et al. (2013) stipulates that CL has proved itself more competent than OWL in rigorously defining the semantics which is a key requirement for heavyweight modelling to support knowledge sharing.

Although CL can support effective knowledge representation and reasoning however it has not got widespread acceptability in information systems community (Delugach, 2009). This is evident from a study carried out by Cardoso (2007) which was accomplished by 627 surveys from a range of respondents from academia and industry. This study reveals that the OWL and RDF(S) are the most used languages for data exchange and knowledge sharing as shown in figure 2.5. The figure clearly indicates that CL is not being fully exploited by the research community besides its potential benefits.

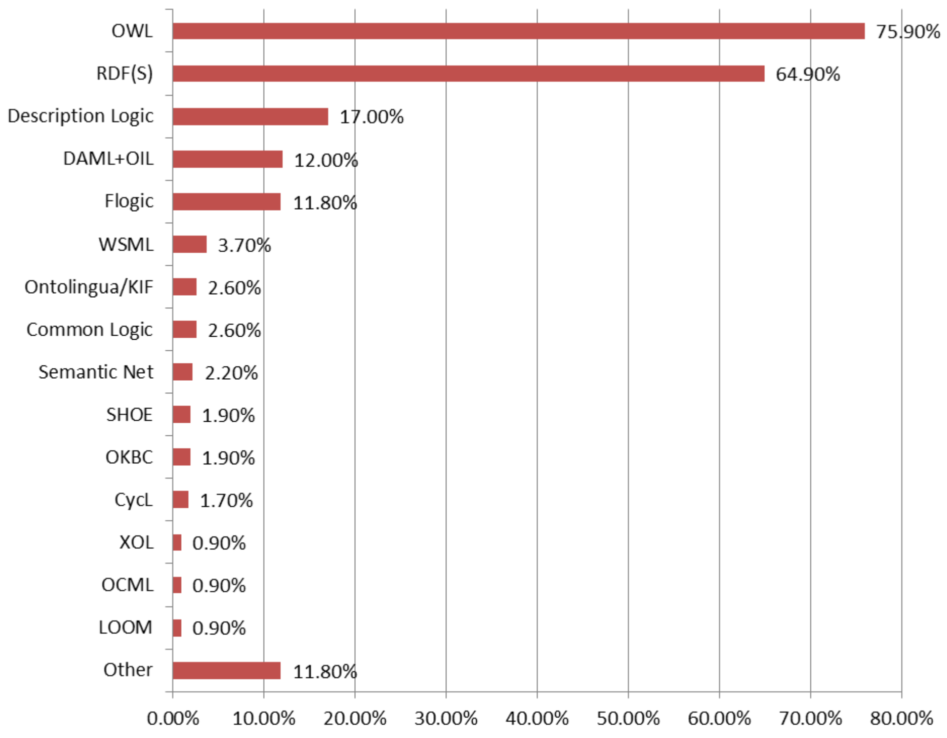


Figure 2.5: Ontology languages by user percentage (Cardoso, 2007)

The research in this thesis uses CL based Knowledge Frame Language (KFL) to represent and share assembly knowledge. The code written in KFL takes the form of directives which are specified with a colon at the start of the line followed by the keywords and the arguments. More detail about the KFL can be found in appendix A.3.

2.4.3 Tools

Ontology development tools provide an environment to load, instantiate, and query the ontologies. Following sections discuss some of the ontology development tools.

2.4.3.1 Ontolingua Server

Ontolingua Server (Farquhar et al., 1997) was the first ontology tool, built during the mid-1990s (Gomez-Perez et al., 2004) by the Knowledge Systems Laboratory (KSL) at Stanford University (Corcho et al., 2002). Ontolingua server "*provides a distributed collaborative environment to browse, create, edit, modify, and use ontologies*" (http://www.ksl.stanford.edu/software/ontolingua/). Farquhar et al. (1997) identify three modes of interaction with Ontolingua Server involving (1) remote collaborators, (2) remote applications and (3) stand-alone applications. In the first case, Ontolingua server helps users (remotely distributed) to browse, create and maintain ontologies using web browsers and allow users to collaborate in a shared session. The second mode: remote applications support users to query and modify ontologies as well as offer access to data and meta-data. Finally Ontolingua server helps users to translate ontologies into a particular format as per their requirements.

2.4.3.2 Protégé

Protégé is a free software tool which helps the users to build domain models and knowledge based applications using ontologies (http://protege.stanford.edu/overview/). Protégé is a standalone application (Corcho et al., 2002) and is widely used for ontology development due to availability of online help (Khondoker and Mueller, 2010). Protégé provides customizable user interface and customizable output file format where the latter can be used to adapt with any formal language (Mizoguchi and Kozaki, 2009).

Protégé facilitates integration with other applications, tools, knowledge bases and storage formats, and supports ontology representation languages like OWL and RDFS (Gasevic, et al., 2006).

2.4.3.3 OntoEdit

OntoEdit (Sure et al., 2002) was first developed at the Institute of Applied Informatics and Formal Description Methods (AIFB) in Karlsruhe University (Corcho et al., 2002). OntoEdit is an ontology engineering tool which provides ontology development environment and can support the users to collaborate from geographically distributed locations (Sure et al., 2002). OntoEdit's ontology editor helps editing and browsing ontologies and has capability of importing ontologies in various formats including XML, RDFS, and FLogic (Gomez-Perez, 2004).

2.4.3.4 WebODE

WebODE (Arpírez et al., 2001) was developed by Ontology and Knowledge Reuse Group at Technical University of Madrid Spain (Su and Ilebrekke, 2002). WebODE is an integrated ontological engineering workbench which allows editing of ontologies as well as provides a development environment for other ontology development tools and applications (Arpírez et al., 2001). WebODE supports collaborative edition of ontologies as the OntoEdit do and its client-server architecture supports high usability and extensibility as compared to Protégé 2000 and OntoEdit (Mizoguchi, 2004). WebODE can support import, and export to and from XML, and can provide translation services for other ontology development languages e.g. RDF(S), OWL, and F-Logic (Mizoguchi and Kozaki, 2009).

2.4.3.5 IODE

Integrated Ontology Development Environment (IODE) is an ontology development tool developed by the HighFleet which provides a platform to build databases, assert the instances, delete the assertions, browse the ontology and allows queries to be made using the query tool (IODE, 2013). Specifically the assertion delete tool provides flexibility in the sense that it allows selected assertions be deleted whenever they are not required. IODE is the only commercially available software tool which provides a development environment for the Common Logic based ontologies (Usman, 2012). Unlike other ontological tools e.g. Protégé, it allows to write ontology codes outside the ontological environment (Chungoora, 2010). This research work uses Notpad++ to write the KFL code for ontologies and these KFL code files are then loaded in IODE to create databases.

Once the database is created in the IODE, it can be instantiated by populating the facts. Afterwards queries can be made to retrieve the required information and/or to evaluate the ontology. The IODE has predefined Upper Level Ontology (ULO) and Middle Level Ontology (MLO) concepts which basically provide a base for the ontologies to be developed.

2.5 Ontologies in Manufacturing Research

A significant amount of work has been done on the use of ontologies in the field of manufacturing engineering. Researchers have exploited the concept of ontologies to deal with different aspects of the manufacturing domain. The main focus of research has been on design and manufacturing planning areas.

On product design side, Chang et al. (2010) proposed an ontology to support design decision and explained ontology development phases for Design For Manufacture (DFM). Wei et al. (2009) proposed an ontology for reuse, integration and sharing of design knowledge to support designer in making decisions during product development. Lin and Harding (2007) proposed Manufacturing System Engineering (MSE) ontology model to support information exchange across the inter-enterprise multi-disciplinary design teams. The work of Wang and Tong (2008) was focussed on analysing the manufacturing knowledge needed to support design decisions and they developed an ontology to support knowledge sharing between the design and the manufacturing domains.

On manufacturing planning side, Borgo and Leito (2004) proposed ADaptive holonic COntrol aRchitecture for distributed manufacturing system (ADACOR) ontology which have been derived from the foundational ontology DOLCE. ADACOR represents the concepts from the manufacturing control area and provides a good understanding related to this domain. Zhou and Dieng-Kuntz (2004) investigated ontology based solution aimed at sharing manufacturing knowledge for realisation of excellent manufacturing. Their work also provides a good understanding of ontologies in general as well as in the field of manufacturing engineering.

Lemaignan et al. (2006) proposed Manufacturing’s Semantic Ontology (MASON) to support capture and sharing of manufacturing knowledge. Their ontology is based on three main concepts: entities, operations and resources. Entities represent geometric features, raw material, cost entities etc. Operations consist of manufacturing operations e.g. machining, logistic operations, human operations etc. and finally resources capture machine tools, human resources and other such resources. Semere et al. (2007) developed a machining ontology to represent the domain knowledge. Their ontology considered various machining related concepts e.g. machining processes, form features, and machining resources.

From the ontological formalism point of view, most of the above mentioned ontologies are based on OWL and/or combination of OWL and SWRL. However as mentioned in section 2.4.2, these formal languages (in comparison to CL) lack the rigour and expressiveness required in the manufacturing domain. CL based ontological approaches have also been exploited in the manufacturing domain. For instance Young et al. (2007), Gruninger and Delaval (2009), Chungoora (2010), Chungoora and Young (2011a), Palmer et al. (2012) and Usman et al. (2013) have used CL based ontological approach to deal with different issues in the manufacturing domain.

However all of the above mentioned researchers have worked in single piece part manufacturing. This research work targets the manufacturing assembly domain for investigating the role of CL based approach for knowledge sharing. Therefore research opportunity exists in this domain.

2.6 Manufacturing Assembly

2.6.1 What is Assembly and why is It Important?

The Oxford dictionary defines assembly as "*a unit consisting of components that have been fitted together*". Various other definitions of assembly have been reported in the literature. For example, Linn (1997) describes assembly as "*a series of tasks putting together a set of components to produce an end product*". In Baudin (2002)'s point of view "*assembly consists of putting or fitting together different parts into a product*". Holland (1997) describes assembly as "*a group of components merged together is called assembly*". Whitney (2004) argues that "*assembly is more than putting parts together*".

It can be inferred from the above definitions that assembly is the combination of different parts held together using different assembly processes and resources. The term assembly can be taken in two contexts: assembly as a process and assembly as a product (Linn, 1997). However in this work, the term assembly process has been used for the first context and the term product has been used for the second context (more details can be found in chapter 4).

Assembly is the most complex process in the industry (Delchambre, 1992) which has not been given attention in the past as compared to other manufacturing processes and therefore it is least understood (Whitney, 2004). Assembly is also a major time and cost contributor towards the development of product as Cho (2005) noted that almost 53% of manufacturing time is consumed in carrying out assembly tasks; 10% to 30% of manufacturing cost is associated with the assembly of the majority of the products and 20-60% of total labour is involved in assembly in making a product in US.

The importance of assembly is further highlighted by Walker (2001), who points out that Boeing and Ford consume 5% to 7% and 10% to 12% of their cost in assembly respectively. He also argues that the assembly cost increases as the size of the company decreases. It implies that even large companies like Boeing and Ford are consuming a handful percentage of cost in product assembly and therefore assembly cost will increase for small and medium industries.

Martin-Vega et al. (1995) investigated that whether the investment in research and development activities, help to significantly reduce the cost and increase the effectiveness of manufacturing assembly. The research was carried out by the Department of Defence (DoD) in USA on 24 product lines in various companies whose annual sales volume ranges from $10 million to $2 billion. They noted that the assembly cost accounts nearly 20% of the manufacturing cost in comparison to previous research which suggests a figure of 4.8%. However their research findings suggest that the investment in assembly research is only significantly useful when it focuses on assembly integration and/or assembly support activities rather than considering assembly processes only.

Whitney (2004) reported that currently assembly is facing technical, economical and managerial challenges. He noted that the technical challenges exist due to increasing complexity and customization of products, economic challenges are caused due to increasing customer demands for high quality and low price products, and managerial challenges are due to increasing dependence on suppliers, and other stakeholders.

From the above discussion it is obvious that there is a need to investigate key aspects of assembly particularly the support activities required for collaboration across the assembly domain. It is also noted that a significant amount of resources (in terms of money, time, labour etc.) are consumed in the assembly of products therefore there is a potential need to undertake research in this area.

2.6.2 Assembly Structure

LV et al. (2011) classified product assembly structure into four layers named as assembly layer, part layer, feature layer and presentation layer as shown in the figure 2.6. The assembly layer represents information related to various aspects of subassemblies e.g. subassembly identification, subassembly relations etc. The parts layer information describes information related to parts identification code, respective subassembly code and parts relationship. The feature layer represents feature related information e.g. feature type, feature name, feature identifier, etc. Finally the presentation layer represents face level assembly information.

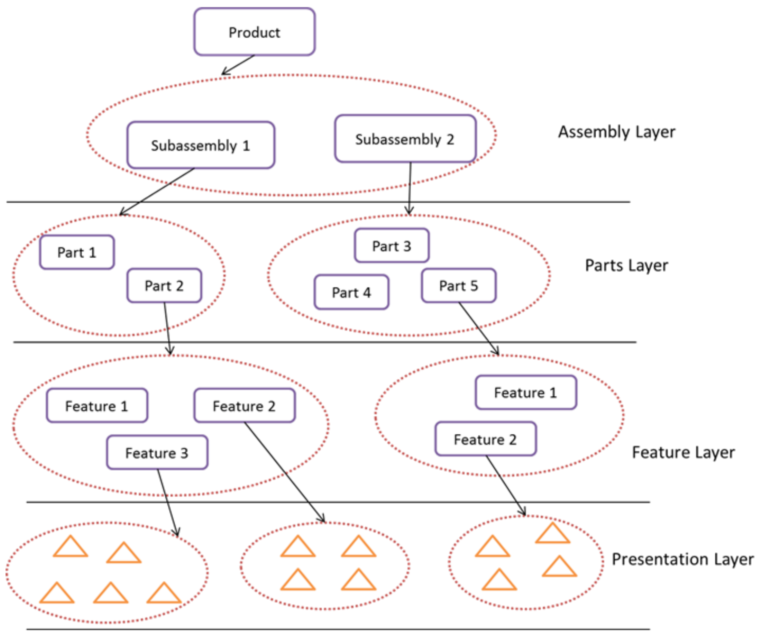


Figure 2.6: Different structure levels of product assembly (LV et al., 2011)

A similar classification of assembly structure has been proposed by Hui et al. (2006). The different levels described in his research are assembly, subassembly, part and feature levels. This thesis also takes similar view of the assembly structure and defines the related concepts e.g. product, component, subassembly, part, and feature which will be described in detail in chapter 4.

2.6.3 Integration of Assembly Design and Assembly Process Planning

Integration of various assembly related tasks including the assembly design, and assembly process planning results in an efficient assembly design (Lit and Delchambre, 2003). The integration of production engineering and design departments should be promoted at the early stages of product development (Sackett and Holbrook, 1988). It suggests that assembly planning knowledge should be introduced during early stages of product design to support the product development process. The existing research has shown an evidence of work done on the integration of assembly design and assembly process planning related tasks.

For instance, Zha and Du (2002) found that integration of assembly design and assembly process planning can support the product development. They proposed a STEP based model for the integration of assembly design and assembly process planning and have found that data in computer interpretable form reduces the product development time by diminishing the intervention of humans for knowledge sharing, and encourages information sharing on product data level rather than on document data level.

Dorador and Young (1999) suggested that the product and manufacturing information models can be linked together to support knowledge sharing between Design For Assembly (DFA) and Assembly Process Planning (APP) domains. They believe that DFA techniques only focus on assemblability and design issues of the product and they do not address assembly planning issues consequently causing lack of assembly planning knowledge during the design phase. Figure 2.7 displays an interaction between DFA and APP which will allow the designer to access the APP related information while doing design for assembly analysis.

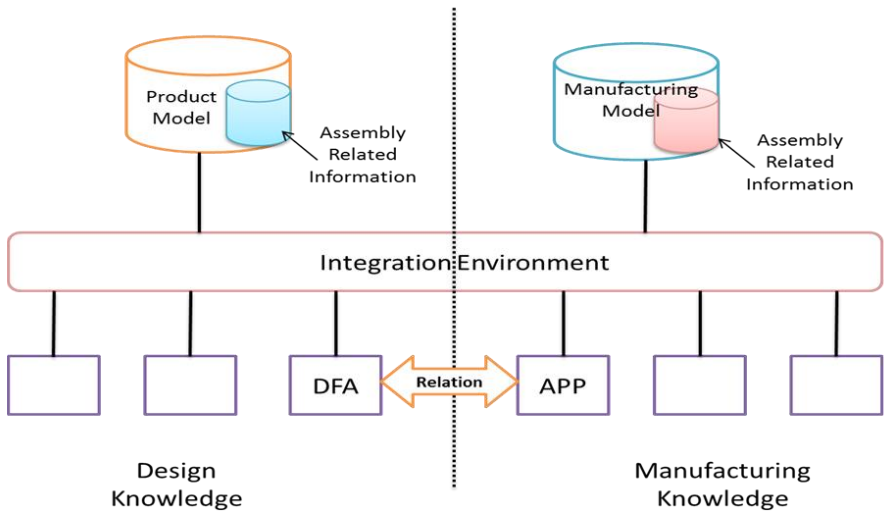


Figure 2.7: DFA and APP integration environment (Redrawn from (Doradar and Young, 1999))

The research carried out by Demoly et al. (2011) focus on the integration of assembly design and assembly process planning phases. They promoted the idea of introducing assembly process planning knowledge during the early product design phase and have found that it is a potential research area in an assembly oriented design. They further argue that semantic and knowledge based assembly models offer a better integration and understanding of assembly planner's intents in early stages of product development.

However to capture semantics to form a knowledge base requires the use heavyweight ontologies as identified in sections 2.2 and 2.3. Therefore it is evident that semantic integration issues in the assembly domain need to be addressed.

2.6.4 Feature based Assembly Knowledge Representation

Features play a key role in the representation of assembly knowledge as Haasis et al. (2003) describe features as career of descriptive and semantic information of product development processes. In another study by Case and Harun (2000), features have been recognised as a better source of assembly knowledge representation as compared to components or parts themselves. Further evidence of features for assembly knowledge representation is given by Holland

and Bronsvoort (2000) who say that "*assembly features can be profitably used in assembly planning modules*". While Molloy et al. (1991) also stress that both Design For Assembly (DFA) and Assembly Process Planning (APP) rely on feature based information.

The previous research has also shown that features have the potential to support integration of assembly design and assembly process planning domains. For example, Bley and Franke (2004) found that features contain information related to different aspects of assembly domain and can support integration of assembly design and assembly process planning. Mantyla et al. (1996) claim that features provide design and manufacturing reusable data repository and can support integration of these domains.

Xu (2009) found that although features have the potential to support integration of design and process planning domains however they are complicated in the sense that they have multiple contexts and definitions. This becomes further problematic as feature based approaches only capture single context of the information (Young et al., 2007). This requires the support of PLM systems which can facilitate the capture of multiple viewpoints of information and their relationships (Young et al., 2007). It is therefore can be established that there is a potential available to address the multiple viewpoints of information attached to assembly features and finding out the relationship between these viewpoints.

2.6.5 Ontologies in Manufacturing Assembly

Increasing recognition of ontologies as a source of knowledge management has attracted many researchers towards the use of ontologies in the assembly domain. However still, the work reported in this area is not very high. The following paragraphs provide an overview of existing research in the assembly domain.

Chakrbarty et al. (2009) developed an ontology to semantically enrich the Variation Reduction Advisor (VRA) system used in General Motor (GM). Their ontology was aimed to control the vocabulary related to the assembly problems and their solutions. Lohse et al. (2006) proposed an ontology to facilitate decisions related to the selection of assembly equipment and to support the reconfigurable assembly system. Lanz et al. (2008) proposed ontology to capture product and process related assembly knowledge by using the feature concept.

Majority of the researchers who carried out ontology based research in the assembly domain, have used OWL as ontological formalism. For instance, Delamer and Lastra (2006) developed an ontology for the modelling of assembly processes. The ontology enables reasoning and inferences of assembly knowledge representation based on OWL and SWRL rules. Kim et al. (2006) proposed an assembly design ontology which provides formal specification of product design related knowledge using OWL and SWRL ontological formalisms. Mostefai et al. (2005) proposed an OWL based ontological approach to capture the product design knowledge to support the product development process. Demoly et al. (2012) developed an ontology to capture the product design and assembly sequence planning knowledge where they have used OWL and SWRL.

However Fiorentini et al. (2007) argued that the main purpose of using ontologies in modelling the assembly knowledge is to exploit the potential advantages of these semantic approaches to formally define the meanings of assembly concepts to enable interoperability across the assembly systems. As the assembly domain is very complex because of the multiple components and features involved in it therefore an ontological approach with high expressive power and reasoning capabilities is required.

Most of the approaches discussed above are either based on lightweight ontological approaches or they use the OWL based approaches. As discussed in section 2.4.2, OWL based approach lacks the capabilities to model complex domains like assembly therefore it requires the new methods for the exploration of semantic representation and interoperability in the assembly domain.

Also the existing ontologies in the assembly domain focussed on specific application areas and no ontology provides a comprehensive set of assembly reference concepts which can be used as foundation to build semantic base to support interoperability across the assembly domain. The thesis targets this research opportunity to explore the role of reference ontologies in the assembly domain to probe the interoperability issues.

2.7 Summary of the Research Gaps

This chapter provided a comprehensive review of the background literature to identify and address the knowledge sharing issues for the assembly domain. Particular focus has been given to (1) knowledge sharing and interoperability, (2) ontology based interoperability, (3) development of ontologies, (4) review of existing ontologies in the manufacturing domain, (5) issues, and opportunities to support knowledge sharing in the assembly domain. The following paragraphs provide highlights of the background literature review in these areas.

The current ICT based tools lack the ability to share knowledge effectively and interoperability is a common issue which is causing huge costs to the modern organizations. Therefore efforts are required to resolve the interoperability issues.

MDI and standard based interoperability approaches have not proved themselves sufficient to resolve interoperability issues due to the lack of their ability to rigorously define the semantics of domain concepts.

Ontological methods based on the heavyweight approach have the potential to provide rigorous definition of domain concepts. However, to large extent, it depends upon the expressiveness and reasoning capabilities of available heavyweight ontological formalisms.

Foundation and domain ontologies have been sufficiently explored in the existing research. However reference ontologies which bridge the foundation and domain ontologies are comparatively a new development and have not been widely researched especially in the manufacturing and assembly domains.

Various ontology development methodologies have been discussed in this chapter however the methodologies developed by Uschold and King (1995) and Noy and McGuinnes (2001) appear to be potential candidates to provide a method to develop reference ontology for the assembly domain.

Different ontology development languages have been explored however Common Logic (CL) based ontological formalism proved more competent for the definition of domain concepts due to its powerful expressive and reasoning capabilities as compared to other languages like OWL. The CL based approach appeared more appropriate to deal with the complex domains like assembly.

Amongst the various ontology development tools explored, IODE is found to be the only commercially available tool which supports CL based ontologies.

Manufacturing assembly has potential impact on the manufacturing cost and is the least understood topic as compared to the single piece part manufacturing. Especially research efforts are required to address the semantic integration issues in the assembly domain.

Features in assembly are found to be useful as they carry important assembly related information. Although, features have the potential to support integration of assembly design and assembly process planning domains however they carry multiple viewpoints which can resist the knowledge sharing efforts. Hence there is a potential available to address the multiple viewpoints of information attached to assembly features.

It has been observed that the use of ontologies in assembly domain is not very old and not many researchers have used ontologies in the assembly domain. Furthermore, most of the heavyweight assembly ontologies use OWL based ontological formalism which lacks the required expressive and reasoning capabilities to deal with the assembly domain. Therefore a more capable CL based approach needs to be investigated to address the requirement of assembly ontology.

Moreover these existing assembly ontologies are found to be dealing with specific application areas and as per author’s knowledge, no attempt has been made, to date, towards the development of a CL based formal reference ontology for the assembly domain that should address the knowledge sharing issues across the assembly design and assembly process planning domains.

The understanding obtained from the background literature review has helped to identify key research gaps and issues, and those specifically in relation to this thesis are listed below.

- There is a requirement for improved ontology based methods to support knowledge sharing across the assembly domain.
- There is a need to understand how to exploit the reference ontologies to address the knowledge sharing issues in the assembly domain.
- The multiple viewpoints of the assembly concepts especially assembly feature concept need to be understood and the issues pertaining how they can relate assembly design and assembly process planning domains.
- There is a need to explore CL based formal ontological methods to represent and share the assembly domain related concepts and knowledge.

The research gaps identified in this chapter fulfils the first research objective as mentioned in section 1.2. The next chapter proposes a mechanism to address these research gaps.

CHAPTER 3

ASSEMBLY REFERENCE ONTOLOGY: A FRAMEWORK TO SHARE ASSEMBLY KNOWLEDGE

3.1 Introduction

This chapter outlines some of the important assembly knowledge sharing issues and identifies key requirements for assembly knowledge sharing. The chapter also explains the research concept presented in the form of an Assembly Reference Ontology (ARO) and highlights aspects of novelty in the ARO. This chapter is organized in the following manner.

Section 3.2 discusses the research issues in assembly knowledge sharing. Section 3.3 highlights the requirements for assembly knowledge sharing. Section 3.4 explores the main features of research concept and covers various aspects of novelty. The structure of the ARO is explained in section 3.5. Finally a short summary of the whole chapter is presented in section 3.6.

3.2 Research Issues in Assembly Knowledge Sharing

Assembly Design (AyD) and Assembly Process Planning (APP) are two important domains in manufacturing assembly, which require frequent collaboration for efficient product development. With the rapid development of Information and Communication Technologies (ICTs), various knowledge based systems have been developed over the years in order to store and reuse the product and process information. However most of the contemporary knowledge based systems lack the requirements of modern manufacturing industry (Fischer & Stokic, 2002). This is because; most of these kinds of knowledge based systems operate well in an isolated capacity (Cochrane, et al., 2005). However when subjected to knowledge sharing environment they fail to serve the purpose.

A potential hindrance in the way of knowledge sharing across different knowledge based systems (including assembly systems) is the incapability of such systems to acquire consensus on the semantics of knowledge content (Musen, 1992). These kinds of systems can be made semantically interoperable, if the semantics of the knowledge associated with such systems can potentially be exchanged without losing their meaning and intent (Chungoora, 2010). It implies that the systems should capture the semantics and the contexts of the knowledge in order to make it applicable for a range of domain systems. However technological support is required to fully capture the semantics of the knowledge and the choice of formal language is also an issue.

The concepts used to capture assembly knowledge may have different implications across the assembly design and assembly process planning domains. For example, the concepts; assembly feature, assembly component, Bill of Materials (BOM) and Product Family (PF), as shown in figure 3.1 are viewed from functional and design aspects during the assembly design stage and are associated with assembly processes and resources during the assembly process planning stage. This implies that these domains dictate the semantics of these concepts and knowledge sharing across these domains may be problematic without taking into account the context in which they are used.

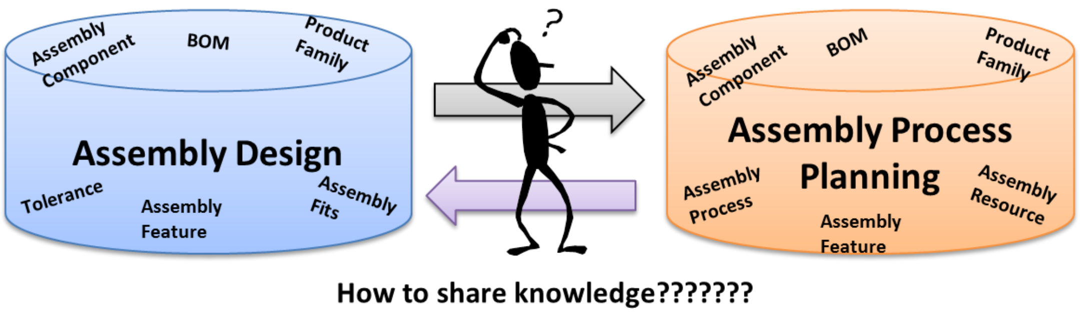


Figure 3.1: Assembly Knowledge sharing problem

So far we have considered the two assembly domains as two different databases where semantic conflicts exist due to the varying nature of these two domains. We term these issues as inter-domain assembly knowledge sharing issues. Another important issue is the intra-domain assembly knowledge sharing issue. Owing to the complexity involved in the manufacturing assembly environment, multiple viewpoints may also exist for the same domain e.g. assembly design or assembly process planning. For example, if a designer using a particular CAD system wants to share information with another designer working on a different CAD system, semantic interoperability issues may arise. Hence semantic conflicts also occur for intra-domain assembly systems as they are caused by multiple overlapping concepts and definitions and multiple representations of similar concepts (Chungoora & Young, 2011b). This problem may be further exacerbated for inter-domain assembly domains as the impact of overlapping concepts and multiple representations (contexts) may increase when we consider two different domains.

To understand the semantic conflicts in terms of manufacturing assembly we can, for example, say that the terms “assembly” and “product” are overlapping concepts in AyD and APP respectively. Similarly the terms: BOM, product family, assembly component and assembly feature are examples of multiple representations of similar concepts in AyD and APP. These multiple representation concepts consequently have different data structures for their respective domains. For example, the concept “BOM” may represent different lists of components for AyD system and APP systems and therefore the associated concepts for both the systems may be different causing the data structure to be different for both domains.

One way to solve the semantic mismatches problem is to use standards to induce interoperability for inter and intra-domain assembly knowledge sharing. However it is important that the system participants should agree to use these standards. Although it may not be possible to have the same standards for all the assembly systems however even if we use standards as a recourse for interoperability, semantic conflicts could still result due to less rigorously defined concepts (Young, et al., 2007). Hence a mechanism is required to reconcile the semantics of multiple assembly systems in order to share assembly knowledge.

3.3 Requirements to Support Assembly Knowledge Sharing

The requirements to support knowledge sharing have been determined based on the analysis of literature especially Michel (2005), Young et al. (2007), Usman (2012), Chungoora et al. (2012) and Palmer et al. (2012), and from the understanding of the assembly knowledge sharing problem. From the analysis, three potential requirements have been identified and these are listed as follows:

1. There is a need to capture the semantics of multiple viewpoints of assembly information and the relationships between them (Young et al., 2007) in order to support assembly knowledge sharing.
2. There is a need to identify a set of reusable assembly reference concepts whose semantics are well defined, for multiple assembly systems to use these concepts in order to share assembly knowledge.
3. There is a need to use an appropriate formal language in order to capture the assembly semantics and to provide shared meanings for multiple assembly systems.

The first requirement is to capture the semantics of multiple viewpoints of assembly concepts which in turn facilitates assembly knowledge sharing. In sections 2.2.2.2 and 3.2, it has been described that the viewpoint or perspective is important for assembly systems to interoperate with each other. It captures the intent of a particular domain or a system which is a requirement for seamless exchange of knowledge. For example, in figure 2.1 assembly design and assembly process planning viewpoints of a bearing shaft assembly were shown and discussed. It was found that the assembly design perspective is more inclined towards finding out the requirements related to the function, material, tolerance and fits of assembly components/features, whereas the intent of the assembly process planning domain is to figure out the requirements related to the assembly process and assembly resources. Hence the context of assembly information is important in the sense that it captures the true intent of the assembly information.

The assembly viewpoints can be captured by identifying a set of reusable assembly reference concepts whose semantics are well defined and this forms the basis of the second requirement of this research. A knowledge base can be created by defining and relating the assembly concepts which can subsequently be used as a common base for multiple assembly domains e.g. assembly design and assembly process planning domains. However if knowledge bases for these assembly domains are developed independently, then there is a potential chance that semantic conflicts could result which may subsequently hinder the knowledge sharing process. Therefore a common reference ontology is required which can capture the meanings of concepts at various levels of specializations. This common reference ontology is comprised of assembly reference concepts that represent assembly information at various levels of specialization.

For example, the concepts of assembly feature, dimension, tolerance, tolerance type and shape attribute can be used to capture the assembly design perspective of bearing shaft assembly shown in figure 2.1. These concepts represent information at various levels of specializations. For instance, the concept shape attribute is a Generic Reference Concept (see figure 3.2 and 3.5) which represents the shape of an object. The Generic Reference Concepts are applicable to multiple domains including the product lifecycle domain and are most generic concepts in the ARO.

The concepts dimension and tolerance represent the design information and are applicable to both single piece part manufacturing and assembly. Therefore they have been included in the Design and Manufacturing Reference Layer of the ARO (see figures 3.2 and 3.5). The concept assembly feature is applicable to assembly therefore an Assembly Specific Layer has been included in the ARO. The Assembly Specific Layer provides the concepts which are applicable to both assembly design and assembly process planning. The concept tolerance type is Assembly Design Reference Concept and because the limits and fits standard BS (4500) was applied on bearing and shaft assembly therefore the concept tolerance type was used in the ARO at this level.

Similarly the concepts assembly process and assembly resource has been identified during the exploration of bearing and shaft assembly (shown in figure 2.1) to capture the assembly process planning viewpoint. As these concepts support the capture of assembly process planning information therefore they have been placed in the Assembly Process Planning Reference layer.

The different levels of specialized concepts within the ARO are required to capture the assembly design and assembly process planning knowledge. These reference concepts can then be specialized and linked with domain specific concepts to support knowledge sharing across assembly design and assembly process planning domains (as was explained in section 2.2.2 with the help of bearing shaft assembly).

The third requirement is related to the use of the most appropriate technological support for the representation and sharing of assembly knowledge. There are various formal languages as discussed in section 2.4.2 which can support the representation of assembly knowledge. However it is imperative to understand that the choice of formal language should be made by taking into account its expressive power and reasoning potential to deal with the complexity involved in assembly. This research work will use KFL as a formal language because it is more expressive and computationally powerful than its competitors such as OWL as explained in section 2.4.2. The KFL supports higher order relations and functions which can be used to capture the semantics of assembly concepts such as tolerance. The KFL also deploys axioms to constrain the semantics of the concepts and infer new knowledge which help to share assembly knowledge.

3.4 The Assembly Reference Ontology (ARO) Concept

3.4.1 Introduction to the ARO Framework

Domain ontologies capture domain specific knowledge and are fairly independent from each other. However in an environment where collaboration is required between multiple domains or systems, domain ontologies fail to interoperate with each other effectively. In contrast to domain ontologies, foundation ontologies are very generic in their content and are highly abstract. Ideally they are designed to cover every domain that exists. For example, the concept “Particular” is a foundational concept taken from the HighFleet’s upper level ontology that refers to those things which are unique or in other words things which are only identical to themselves. In the context of assembly design and assembly process planning domains, there would be a large number of concepts classed as “Particular”. This kind of situation would lead towards finding similarities between enormously different concepts (Usman, 2012) when viewed from the assembly design and assembly process planning domains. Hence there is a need to identify a set of concepts whose semantics are generic, as compared to assembly design and assembly process planning domains, and more specialized in comparison with foundation concepts. In this research, we call these concepts “reference concepts” and the ontology developed through these concepts a “reference ontology”.

The Assembly Reference Ontology (ARO) contains multiple layers of reference concepts and aims at fulfilling the assembly knowledge sharing requirements. Different layers of ARO are delineated in figure 3.2. The ARO starts from the Generic Reference Concepts and ends up at Assembly Design and Assembly Process Planning Reference Concepts layer. Other layers of reference concepts include the Product Lifecycle Reference Concepts, Design and

Manufacturing Reference Concepts and Assembly Specific Reference Concepts. These layers have been identified based on the analysis of existing literature and exploration of examples of product assemblies (butterfly valve assembly, journal bearing assembly, and products and assembly resources considered for the case study).

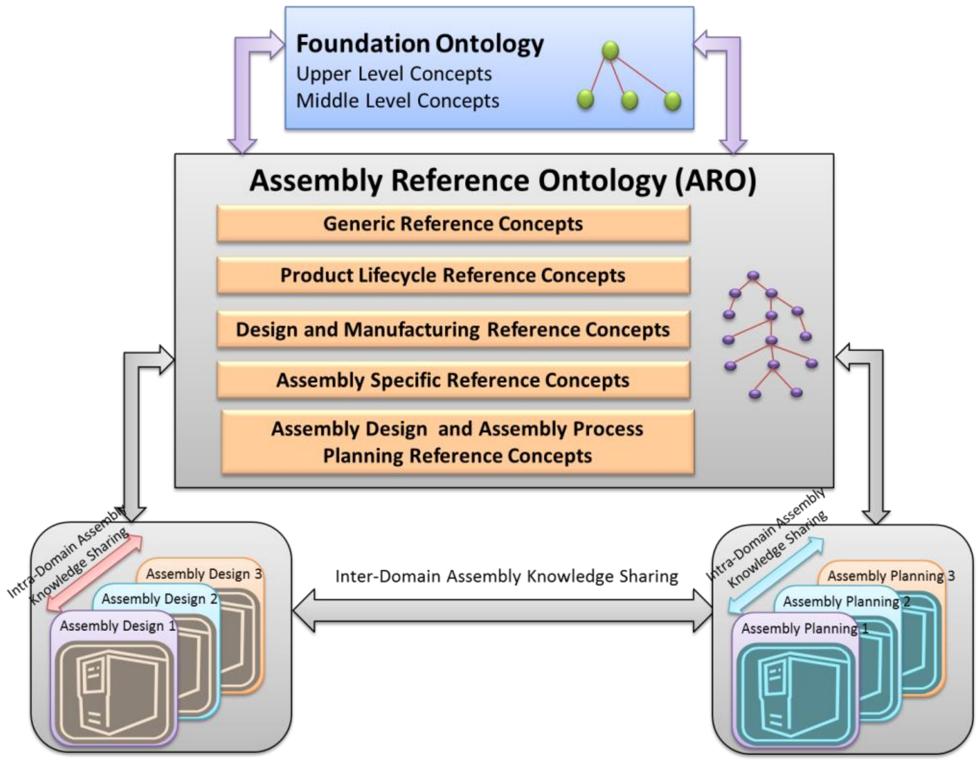

Foundation Ontology
Upper Level Concepts
Middle Level Concepts
Assembly Reference Ontology (ARO)
Generic Reference Concepts
Product Lifecycle Reference Concepts
Design and Manufacturing Reference Concepts
Assembly Specific Reference Concepts
Assembly Design and Assembly Process Planning Reference Concepts
Intra-Domain Assembly Knowledge Sharing
Assembly Design 3
Assembly Design 2
Assembly Design 1
Inter-Domain Assembly Knowledge Sharing
Intra-Domain Assembly Knowledge Sharing
Assembly Planning 3
Assembly Planning 2
Assembly Planning 1

Figure 3.2: Assembly Reference Ontology (ARO): A Framework to Share Assembly Knowledge

The layers that capture the Generic Reference Concepts and the Product Lifecycle Reference Concepts are taken from Usman (2012) in the Interoperable Manufacturing Knowledge Sharing (IMKS) project for single piece part machining and are extended for assembly in this work. The purpose of these layers was to capture the meanings of concepts at a generalized level which could then provide a semantic base for more specialized design and production concepts. Although his layers are appropriate for the ARO, there are a number of assembly concepts that need to be added. These are process, material, operation, spatial location and shape attribute at generic reference level and product version, product feature, BOM, component and auxiliary material at Product Lifecycle Reference level.

Usman (2012) have also used product design and production level reference concepts however these layers were very specific to single piece part manufacturing especially part machining. Therefore a more generic level was needed that should cover both single piece part manufacturing and product assembly. For this reason the Design and Manufacturing Reference Layer has been introduced in this research to represent the design and manufacturing related information. For example, the dimension and tolerance concepts shown in figure 2.2 are captured at this level. These concepts are further explained in section 5.3.

The Assembly Specific and Assembly Design and Assembly Process Planning Reference layers have been introduced in this work to specifically represent the assembly related information. These layers have been identified by analysing the examples of product assemblies e.g. butterfly valve assembly (will be reported in chapter 4, 5 and 6), journal bearing assembly (will be reported in chapter 5 and 6) and engines and assembly resources (will be reported in chapter 6). The Assembly Specific Reference layer is needed to represent the concepts which are applicable within the assembly domain and the concepts in this layer are common across the assembly design and assembly process planning. For example, the concept “assembly feature” shown in figure 2.2 was captured at this level which has been used to link assembly design and assembly process planning knowledge. A further explanation of this concept has been done in section 5.3.

The Assembly Design and Assembly Process Planning Reference layers are required to represent the assembly design and assembly process planning specific concepts. For example the BOM concept was explored using the example of butterfly valve assembly and it was found that engineering BOM (EBOM) and manufacturing BOM (MBOM) represent the assembly design and assembly process planning related information respectively. Therefore these concepts were captured at Assembly Design and Assembly Process Planning Reference layer. Similarly the concepts assembly process and assembly resource shown in figure 2.2 are captured at this level.

It is important to understand that layers of reference concepts do not necessarily follow the layer by layer specialization or generalization. For example the assembly process planning layer is not a specialized layer of assembly design layer. Similarly some reference concepts may not have a parent class from the very next generalized layer. For instance MBOM is an assembly process planning reference concept and its super class BOM is a product lifecycle concept. This implies that MBOM class has by-passed the immediate generalized layers: the assembly specific layer and the design and manufacturing layer.

The application or domain specific assembly design and assembly process planning ontologies are shown at the bottom of figure 3.2. The concept is that these domain ontologies can interoperate with each other through the ARO and by exploiting some of the foundation concepts. Another aspect of the research is the intra-domain assembly knowledge as discussed in the previous section. It is proposed that the ARO can also be potentially used to support assembly design and/or assembly process planning intra-domain knowledge sharing. It implies that the developed framework ARO behaves like a reference ontology for both assembly design and assembly process planning domains as a whole as well as acting like a reference ontology for assembly design and assembly process planning individually. This is the reason it is necessary to include some of the reference concepts in ARO from the assembly design and assembly process planning domains.

3.4.2 Novel aspects in the ARO Framework

The ARO is proposed to deal with the assembly knowledge sharing problems and to fulfil the requirements of assembly knowledge sharing. The novelty of the proposed framework covers the following major aspects.

1. The identification and formal definition of a set of reusable intermediate assembly reference concepts which can support assembly knowledge representation and sharing.
2. The proposed ARO framework contributes towards the understanding of the need for specialization of assembly concepts which can ultimately support knowledge sharing across the assembly domain and the development of application ontologies by providing higher level abstract concepts as a base for their development.

3.4.2.1 Knowledge Representation and Sharing Through the Identification and Formalization of Assembly Reference Concepts

The reference concepts have been identified by reviewing the existing literature on manufacturing and assembly ontologies, general assembly literature, and by using some of the assembly design and assembly process planning software systems. As described by Chungoora (2010) that all the product design and manufacturing related domains, which are associated with similar kind of products, share a set of concepts whose semantics may be applicable to all these domains. Similarly, it is argued in this research that most of the reference concepts identified in the ARO may be applicable to both assembly design and assembly process planning domains, and therefore the knowledge represented by these reference concepts can be shared across these domains.

The examples of a set of reference concepts identified for this research work are: product, BOM, dimension, tolerance, assembly feature, assembly component, assembly process, assembly operation, assembly resource and manufacturing facility. These concepts have been explained in sections 4.3.1, 4.3.6, 4.3.9, 4.3.10, 4.3.11, 4.3.12, 4.3.14, 4.3.15, 4.3.17 and 4.3.18 respectively. The BOM related concepts e.g. EBOM, MBOM and assembly component have been further explored in section 5.2 to investigate the case of intra-domain assembly knowledge sharing. The concepts assembly feature, dimension, tolerance, assembly process, assembly resource and manufacturing facility have been used to investigate a case of inter-domain assembly knowledge sharing in section 5.3. The concepts product, assembly feature, assembly resource and assembly operation have been further investigated in section 6.4 for a case study in the automotive sector.

A key requirement for the formal definition of assembly reference concepts is the use of an appropriate formal language. As discussed in the literature review section, textual definition and description of domain content does not necessarily support the interoperability across the information systems. Hence it requires the use of heavyweight ontologies which can computationally capture the domain semantics and thus provide a support for knowledge sharing. As axioms establish the semantic interpretation of ontological concepts and relations (Fürst, 2005) they are an essential part of heavyweight ontologies. However the potential of reasoning and inference capabilities depends upon the selection of the formal ontological approach.

The ontological formalism used for this research is the Knowledge Frame Language (KFL) (KFL Reference, 2012) which is based on Common Logic (CL) (ISO/IEC 24707, 2007). CL is more expressive than Web Ontology Language (OWL) (W3C Website, 2006) which is currently used in the majority of existing ontology research in manufacturing. The author is of the view that CL is more capable of representing the semantics of complex manufacturing concepts and relationships (Chungoora et al., 2013).

Knowledge can be represented at the meta–level as well as at instance or individual level (Usman, 2012). In the context of this research, the meta-level knowledge can be captured using the reference concepts identified in the ARO and their inter-relationships. These concepts and relationships are declared as properties and relations respectively in the experimental software system “Integrated Ontology Development Environment” (IODE). These concepts or properties are treated as variables which can have more than one instance or individuals. This implies that these concepts can represent any number of instances and can hence build various types of relationships between these instances. Furthermore rules and constraints can be applied by using these concepts and their relationships to support the inference of new knowledge and to prevent the faulty assertions of instances. Once the meta-level knowledge structure is in place, this can be used as a knowledge structure for instances as well.

For example, the statement “Operator assembles Product” is a meta-level knowledge structure where both “Operator” and “Product” represent concepts and “assembles” represent a relationship. This kind of knowledge structure can accommodate any number of instances. For example, one of the possible individual level knowledge conversions of the above mentioned statement may be: “Mike assembles AutoEngine001A” where Mike is an instance of the concept “Operator” and AutoEngine001A is an instance of the concept “Product”. It is important to notice that as instances cannot be further instantiated hence the statements constructed by these instances cannot be further instantiated as well.

The next argument is the fact that a set of formally defined assembly reference concepts support knowledge sharing across assembly design and assembly process planning domains. It has been reported in the previous research that the use of common vocabulary supports sharing of formally represented knowledge (Gruber, 1993). In the context of manufacturing assembly, most of the reference concepts are common to both assembly design and assembly process planning domains and provide a link to share assembly knowledge across these domains. However the other concepts in the ARO support the representation of domain specific knowledge and such concepts may be used when the extension of the ARO is required for a particular domain. These concepts also support the determination of the impact of changes between domains.

For instance assembly feature is a common concept for the assembly domain however the associated design and planning concepts e.g. assembly fits and assembly process are not common across the assembly domain. As these concepts are linked with the common concept, hence both of these concepts can be linked with each other as well. Furthermore as these concepts are formally defined they can be used to deduce new information as well (as shown in figure 3.3).

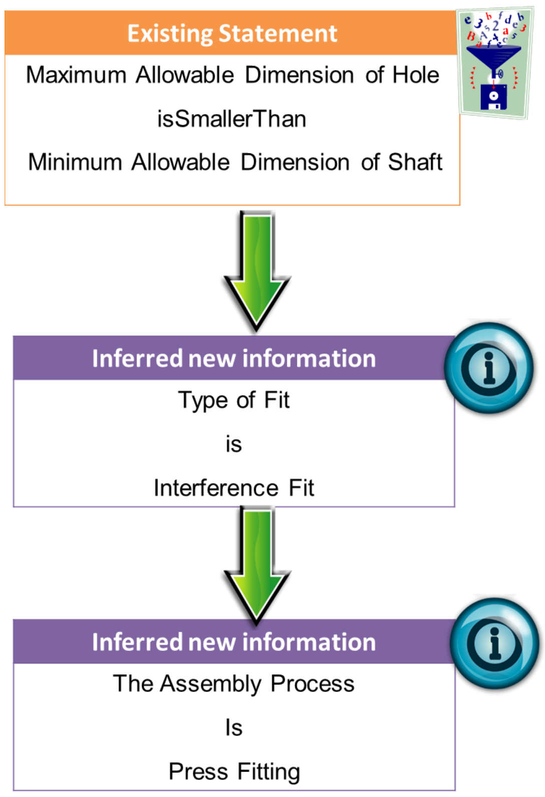


Figure 3.3: An example of knowledge rule for assembly scenario

Figure 3.3 demonstrates an example of formally defined assembly concepts which shows an assembly scenario. The figure suggests that if minimum and maximum allowable dimensions of an assembly feature are known then the type of fit and assembly process can be determined. This implies that the design knowledge associated with assembly feature can be shared in the assembly process planning phase and vice versa. A complete detail of this scenario can be found in chapter 5.

3.4.2.2 Understanding the need for concept specialization

The ARO comprises of reference concepts which are specialized from the most generic form to the most specialized form. The concepts in foundation ontologies are fairly generic and theoretically can be applicable to anything that exists in the universe. For example, the foundation concept “Top” (taken from the HighFleet’s ULO) is described as ‘anything which exists in the universe of discourse is an instance of “Top”’. It implies that it can have a large number of interpretations and can be applicable to wide range of things. Although it is possible to constrain the concept “Top” for a more specialized purpose using the logic constraints however it may clutter the model with a lot of axioms. Another reason is that the foundation concepts may not be constrained for a large number of specialized concepts. Hence the obvious solution to these kinds of problems is to define more specialized concepts which are less generic as compared to foundation concepts.

The various levels of concept specialization provide a support to formally define the respective domain knowledge and to provide a link for knowledge sharing across the domains. For example, consider the concept hierarchy shown in figure 3.4. There are three tiers of concepts starting from BOM class leading to other domain specific concepts. Generally a BOM is considered a list of items which are required to manufacture a product (more details about BOM concept are provided in Chapter 4 and 5). This BOM concept is common for both assembly design and assembly process planning domains, and can be used as link to support knowledge sharing across these domains. The second tier consists of Engineering BOM (EBOM) and the Manufacturing BOM (MBOM) which represent the assembly design and assembly process planning perspective. These concepts may further have some domain specific concepts as shown in figure 3.4 and the EBOM and MBOM concepts would be common concepts for their respective third tier of concepts and hence can support knowledge sharing across these sub-domains.

The above mentioned example also leads to the fact that the ARO concepts can also provide a foundation for developing application ontologies. For instance, formally defined EBOM and MBOM concepts can behave as parent/base concepts for application based ontologies. This is because there could be different interpretations of these concepts for different manufacturing facilities across the globe (more details about different interpretations of MBOM concepts will be discussed in chapter 5).

It is interesting to evaluate the capability of ARO for a selected application and this requires a real manufacturing assembly scenario. A case study in the automotive sector has been used as test case to evaluate and validate the ARO and is discussed in chapter 6.

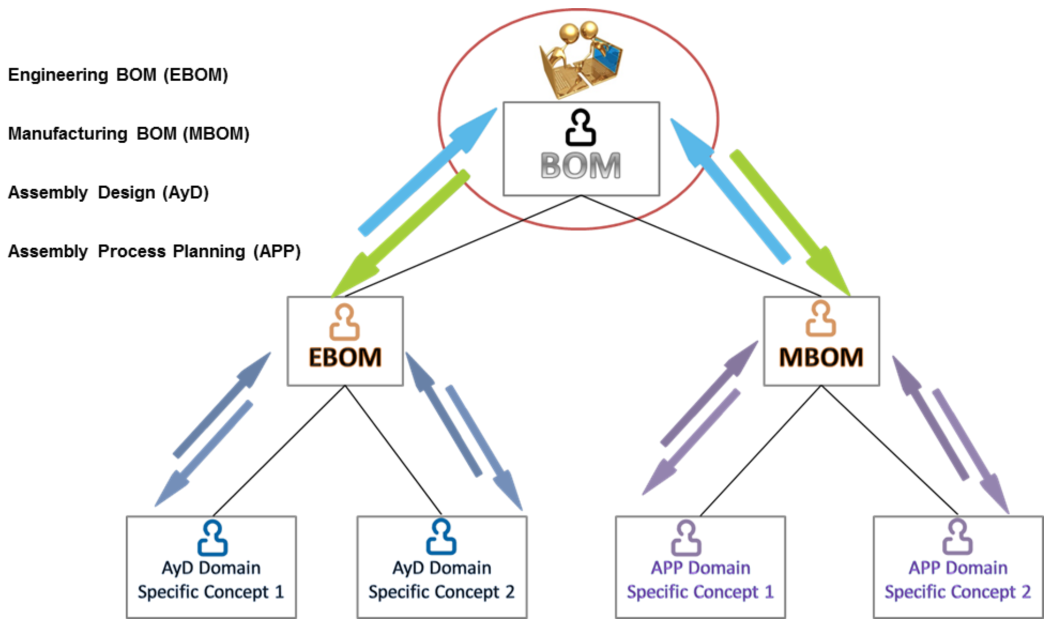

Engineering BOM (EBOM)
Manufacturing BOM (MBOM)
Assembly Design (AyD)
Assembly Process Planning (APP)
BOM
EBOM
MBOM
AyD Domain Specific Concept 1
AyD Domain Specific Concept 2
APP Domain Specific Concept 1
APP Domain Specific Concept 2

Figure 3.4: Knowledge sharing through assembly reference concepts

3.5 The Structure of ARO

The ARO consists of different levels of specializations of reference concepts starting from the most generic concepts to most specialized concepts as shown in figure 3.5. Earlier, levels of specialization have been developed for single piece part manufacturing by Usman (2012) however they are not applicable to the assembly domain and hence they have been modified for the assembly domain. The specialization levels defined in the ARO provide an understanding of the varying levels of depth of the semantics of assembly concepts and consist of the following layers of reference concepts as shown in figure 3.5.

- Generic Reference Concepts
- Product Lifecycle Reference Concepts
- Design and Manufacturing Reference Concepts
- Assembly Specific Reference Concepts
- Assembly Design Reference Concepts
- Assembly Process Planning Reference Concepts

The concepts within these specialization levels (shown in figure 3.5) have been identified by analysing the existing literature and the examples of product assembly scenarios explored in this thesis. In this research at first the key concepts have been identified and then they have been specialized and/or generalized. For example the concept assembly feature was identified from the existing sources (more detail can be found in section 4.3.12) however its generalized concepts e.g. product feature and feature has been adapted from Usman (2012). He used the concept part feature at Product Lifecycle Reference level and feature at Generic Reference level. The concept part feature has been adapted as product feature in this research.

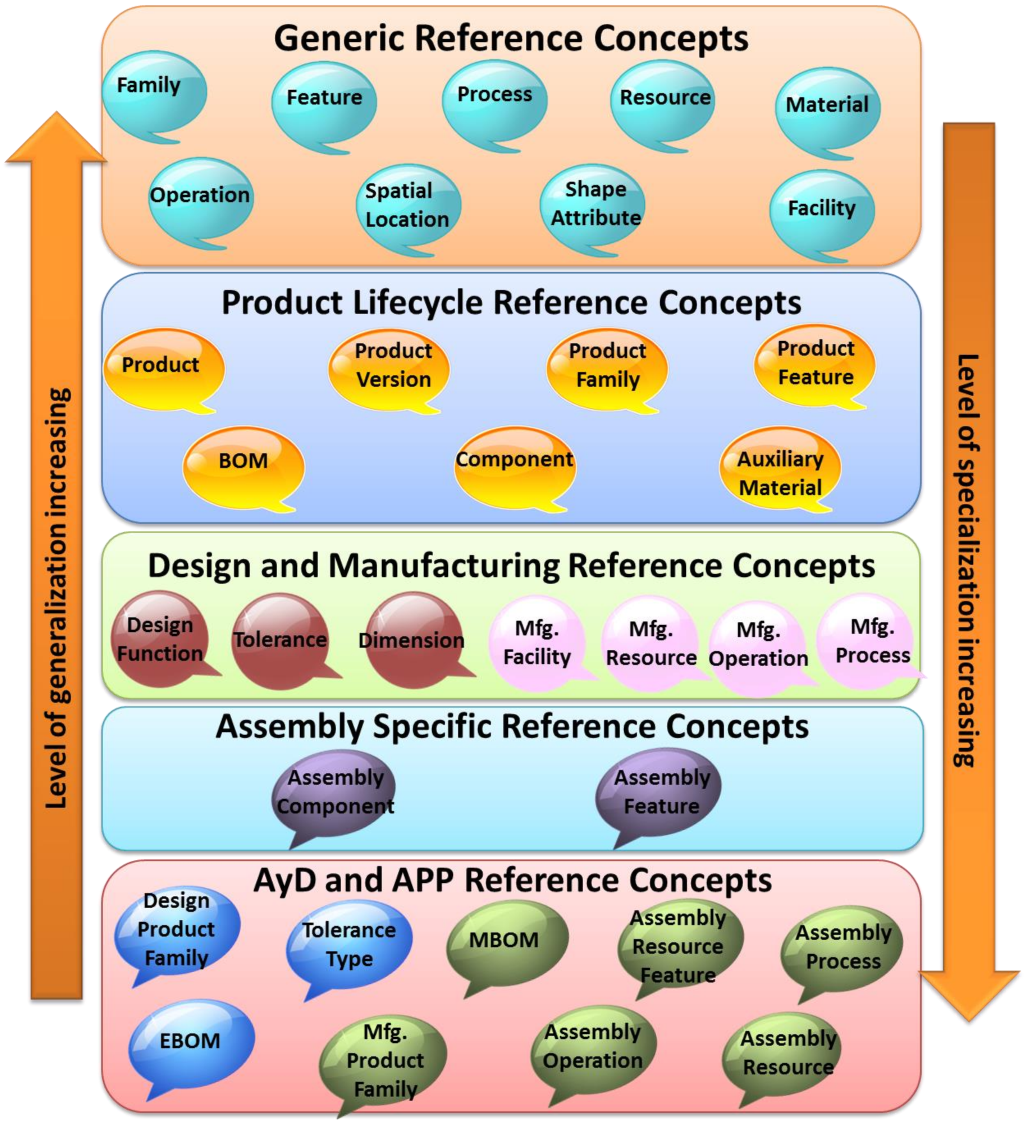


Figure 3.5: Structure of Assembly Reference Ontology

Similarly Usman (2012) used the Product Lifecycle Reference Concept part family which has been adapted as product family because the latter is applicable to assembly. The AyD and APP Reference Concepts "design product family" and "manufacturing product family" have also been adapted from Usman (2012)'s design part family and production part family respectively. Likewise the Product Lifecycle Reference Concept product version has also been adapted from part version which was used for part machining by Usman (2012). The concept design function has also been taken from (Usman, 2012) to represent the function of assembly components and features.

The reference concepts: BOM, component, auxiliary material, tolerance, dimension, assembly component, tolerance type, assembly process, assembly resource and manufacturing facility have been identified by exploring the examples of butterfly valve assembly and journal bearing assembly for assembly knowledge sharing. These reference concepts have been further explained in section 4.3.

The Generic Reference Concept shape attribute has been identified by exploring the examples of journal bearing assembly to capture the fits related knowledge. The Generic Reference Concept spatial location has been identified during the exploration of the case study to represent the location of features on product and assembly resource. The other Generic Reference Concepts such as family, process, resource, material, operation and facility are the generalized classes of product family, assembly process, assembly resource, auxiliary material, assembly operation and manufacturing facility.

From the analysis of case study (described in section 6.4) the concepts: product, assembly operation and assembly resource feature were identified. A detail description of these concepts can be found in section 4.3.1, 4.3.14 and 4.3.19.

The different levels of reference concepts shown in figure 3.5 are explained in the following sections.

3.5.1 Generic Reference Concepts

The generic reference concepts are the first layer of concepts which are more specialized than the foundation concepts and are more generic as compared to the product lifecycle concepts. This implies that the generic reference concepts can be used to support interoperability across the product lifecycle domains e.g. design, manufacture, assembly, operations, services, quality, and disposal, as well as other domains like finance, human resource, marketing etc. However this research work focuses on manufacturing assembly only which comes under the umbrella of product lifecycle domain, hence other domains are not part of the scope right now. The examples of generic reference concepts include concepts like family, feature, process, resource, material, operation, spatial location, shape attribute, and facility.

3.5.2 Product Lifecycle Reference Concepts

The product lifecycle reference concepts capture the semantics of concepts which are applicable to the product lifecycle domain only. The purpose of introducing these reference concepts is to support interoperability within the product lifecycle domain and to provide reference concepts for its sub-domains. Examples of product lifecycle generic concepts include product, product version, product family, product feature, BOM, component, and auxiliary material. These reference concepts interlink the generic reference concepts with more specialized concepts like design and manufacturing reference concepts, assembly specific reference concepts etc. The knowledge captured through these concepts can be used across the product lifecycle domain. For example, knowledge related to the BOM can be used during the design, assembly, disassembly and service of the product.

3.5.3 Design and Manufacturing Reference Concepts

In general, design and manufacturing domain covers single piece part design, assembly design, single piece part manufacturing and assembly process planning domains. Hence the design and manufacturing reference concepts can support interoperability across the above mentioned domains by providing reference concepts which are applicable to all these domains. The examples of design reference concepts include design function, tolerance and dimension, and examples of manufacturing reference concepts include manufacturing facility, manufacturing resource, manufacturing operation, and manufacturing process.

3.5.4 Assembly Specific Reference Concepts

Conceptually manufacturing assembly is significantly different from single piece part manufacturing as the former deals with relationships of parts rather than focussing on a single part. Hence it may require some concepts which are applicable to the assembly only. This would also support interoperability across the assembly design and assembly process planning domains. However not all of the assembly design and assembly process planning knowledge can be routed through assembly specific reference concepts. For example, in some cases, assembly specific reference concepts can be bypassed if appropriate reference concepts are not available to support interoperability across these domains. In those cases, higher level reference concepts e.g. product lifecycle reference concepts may be used to bridge the assembly design and assembly process planning domains. Examples of assembly specific reference concepts are assembly component, and assembly feature.

3.5.5 Assembly Design Reference Concepts

The assembly design reference concept layer is one of the most specialized layers in the ARO. The assembly design reference concepts have been added in the ARO to support the creation of new assembly design domain ontologies and/or to support interoperability across these domain ontologies. Examples of assembly design reference concepts are design product family, EBOM and tolerance type. These concepts capture the assembly design knowledge and can be further specialized to support specific applications.

3.5.6 Assembly Process Planning Reference Concepts

The assembly process planning reference concept layer is also one of the most specialized layers of ARO. Like the assembly design reference concepts, assembly process planning reference concepts also support the creation of new assembly process planning domain ontologies and facilitate interoperability across these domain specific ontologies. This kind of interoperability is termed as intra-domain interoperability as shown in figure 3.2. Examples of assembly process planning reference concepts are MBOM, manufacturing product family, assembly resource feature, assembly operation, assembly process and assembly resource.

3.6 Summary

This chapter has highlighted some of the assembly knowledge sharing issues which provide a base to develop a set of requirements and the research idea to deal with these requirements. It is argued in the chapter that the Assembly Reference Ontology (ARO) can support assembly knowledge sharing across assembly design and assembly process planning domains by providing a set of reference concepts. The chapter also put emphasis on the use of a heavyweight ontological based approach to represent and share assembly domain knowledge. It is also claimed that the ARO can support intra-domain knowledge sharing and the development of new application based ontologies. Various levels of concept specialization have also been explored at the end of the chapter.

CHAPTER 4

THE DEFINITION OF ASSEMBLY REFERENCE ONTOLOGY CONCEPTS AND THEIR RELATIONSHIPS

4.1 Introduction

This chapter explains the detailed investigation of Assembly Reference Ontology (ARO) concepts and the relationships between them. Specifically the assembly specific reference concepts and assembly design and assembly process planning reference concepts (shown in figure 3.5) have been identified through the analysis of literature review and the evaluation of example scenarios considered in this research. These reference concepts and the higher level concepts have been adapted to support the development of the ARO by concept specialization and by associating them with each other as well as with the upper level concepts. They are then formalized using the heavyweight ontological approach. The chapter addresses the requirements of identifying a set of assembly reference concepts and their heavyweight formalization as mentioned in section 3.3 of chapter 3.

The chapter is organised in four main sections. Section 4.2 describes key foundation level concepts. Section 4.3 explains the detailed investigation of assembly reference ontology concepts. The exploration of the ARO concepts involves their informal definitions, generalization, and associations with each other and their formal definitions. Section 4.4 explains the combined representation of assembly reference concepts specializations. Section 4.5 describes the inter-class relations between the ARO concepts. Section 4.6 provides an overview of the formalization of the ARO in KFL based formal language. Finally section 4.7 describes the summary of chapter 4.

4.2 Foundation Level Concepts

It is important to understand some of the most relevant foundation level concepts which will provide a base for the ARO concepts. These foundation level concepts have been taken from the Highfleet’s Upper Level Ontology (ULO) and Middle Level Ontology (MLO) concepts and therefore are named as upper level and middle level concepts as shown in figure 3.2 in chapter 3.

The ULO’s upper most concept “Top” (shown in figure 4.1) is the foundation level concept which represents all the things that exist in a particular domain. The concept “Top” is similar to the most upper level concepts for other foundation ontologies. For instance, Cyc uses the concept “Thing” (Guha and Lenat, 1990), whereas SUMO and BFO use the concept “Entity” (Oberle et al., 2007) (Lambert et al., 2009), at the top of their hierarchies of concepts.

The next level foundation concept shown in figure 4.1 is the concept “Particular” which describes unique things in the universe of discourse. The concept “Particular” has further two sub-classes named as “AbstractEntity” and “ConcreteEntity” as shown in figure 4.1. These “AbstractEntity” and “ConcreteEntity” concepts are in line with the SUMO concepts “Abstract” and “Physical” respectively. The “AbstractEntity” concept represents those particulars which cannot be located somewhere or particulars which do not have any location. Concepts like “DesignFunction”, and “ShapeAttribute” can be subsumed under the concept “AbstractEntity”. The “AbstractEntity” has a sub-class named as “Quantity” which is a MLO concept and describes a family of numeric measurements. All the measurement related classes e.g. “Dimension” and “Tolerance” can be subsumed under the concept “Quantity”.

In contrast to “AbstractEntity”, the concept “ConcreteEntity” represents those particulars which can be located somewhere or particulars which have a particular location. In the similar manner as SUMO has “Object” and “Process” classes subsumed under the class “Physical” (Nile and Pease, 2001), the classes “Object” and “Event” are subsumed under the class “ConcreteEntity” as shown in figure 4.1. The “Object” and “Event” classes can also be referred as “endurants” and “perdurants” respectively as the latter two concepts have been categorized in the DOLCE taxonomy (Oberle et al., 2007). Endurants are wholly present at a particular time while perdurants may be partially present at a particular time (Gangemi et al., 2002). On the similar lines Highfleet defines the concepts “Object” and “Event” on the basis of time. The “Object” concept as described by the Highfleet, represents those concepts which endure through time whereas the concept “Event” represents those concepts which unfold through time. For example the ARO concepts assembly operation and assembly process can be subsumed under “Event” and the concepts product, and component can be subsumed under the concept “Object”.

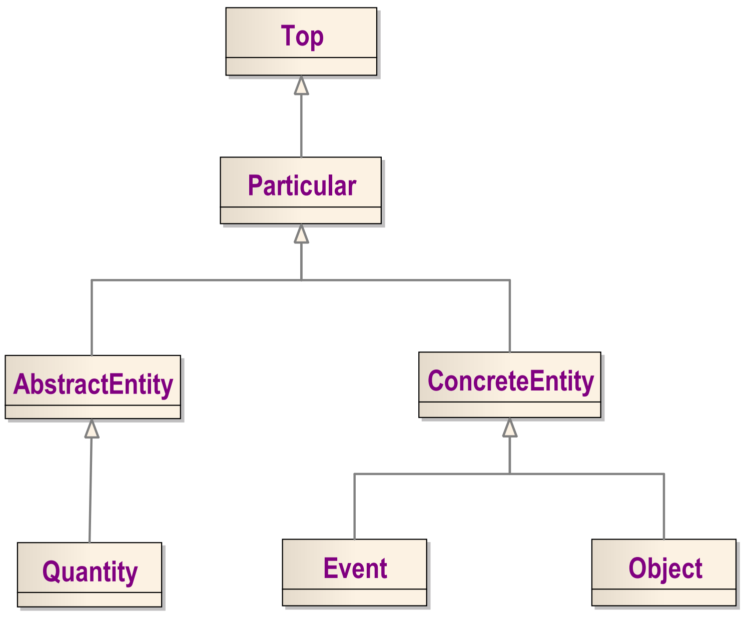


Figure 4.1: UML based representation of foundation level concepts

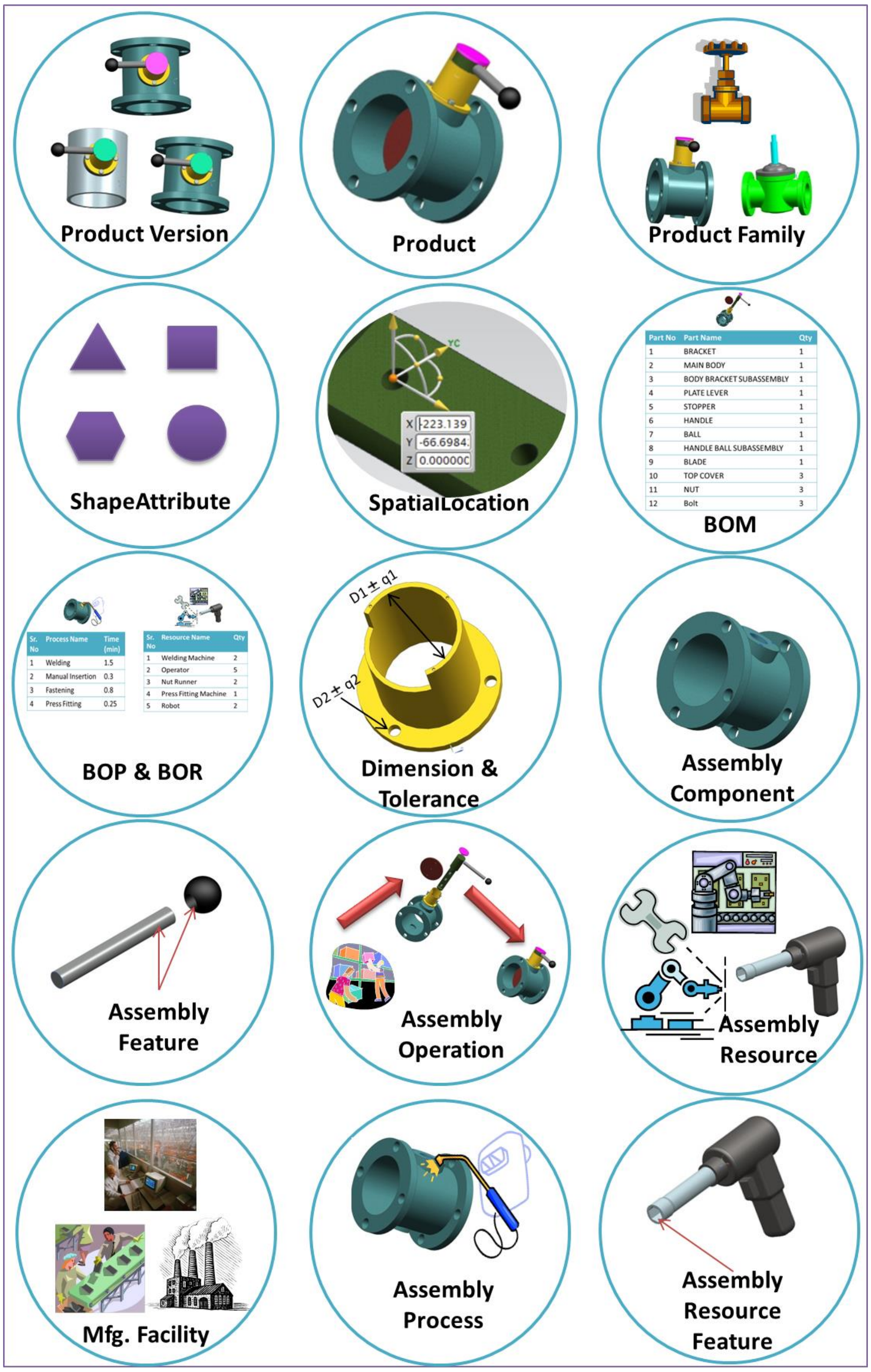
These foundation level concepts have been specialized to subsume ARO concepts. The next section discusses about ARO concepts in detail.

4.3 Overview of Key Assembly Reference Concepts

As discussed in chapter 3, the main purpose of introducing a set of reference concepts is to capture, represent, and share assembly knowledge. The reference concepts synthesize and capture various aspects of assembly design and assembly process planning knowledge. Key reference concepts identified for the assembly reference ontology are product, product version, product family, shape attribute, spatial location, Bill of Materials (BOM), Bill of Process (BOP), Bill of Resource (BOR), dimension, tolerance, assembly component, assembly feature, assembly operation, step, assembly resource, manufacturing facility, assembly process, and assembly resource feature. These concepts are linked with each other through the inter-class relationships and are connected with their parents and child classes through the generalization and specialization relations.

Figure 4.2 illustrates key ARO concepts with the help of a butterfly valve assembly example. For instance, the valve assembly in figure 4.2 is an instance of the reference concept product. Similarly, valve base is an example of assembly component and, the ball and handle form features (shown by arrows) are the instances of assembly features. The concept “Bill of Materials” (BOM) represents a list of assembly components with their quantities and is shown in figure 4.2 for the valve assembly product. Instances of other key reference concepts: product version, product family, BOP, BOR, dimension, assembly operation, assembly resource, manufacturing facility, assembly process, and assembly resource are also shown in figure 4.2.

This research work contributes towards the ARO and aims at capturing the semantics of a set of ARO concepts to support knowledge sharing across the assembly domain. However the ARO can be further extended to accommodate various other dimensions of assembly domain. The reference concepts identified for this research as shown in figure 4.2 are further explained in the following sections.



Part No	Part Name	Qty
1	BRACKET	1
2	MAIN BODY	1
3	BODY BRACKET SUBASSEMBLY	1
4	PLATE LEVER	1
5	STOPPER	1
6	HANDLE	1
7	BALL	1
8	HANDLE BALL SUBASSEMBLY	1
9	BLADE	1
10	TOP COVER	3
11	NUT	3
12	Bolt	3

Sr. No	Process Name	Time (min)
1	Welding	1.5
2	Manual Insertion	0.3
3	Fastening	0.8
4	Press Fitting	0.25

Sr. No	Resource Name	Qty
1	Welding Machine	2
2	Operator	5
3	Nut Runner	2
4	Press Fitting Machine	1
5	Robot	2

Figure 4.2: Examples of some of the key assembly reference concepts used in assembly reference ontology

4.3.1 Product

The term product has multifaceted interpretations and is commonly used in the product lifecycle domains. For example, in Teamcenter (PLM based software), a product is described as a single item or an assembly of items required to be manufactured. In Boothroyd’s DFA software system, both product and assembly have been used to refer to the product. However in context of the present work, the concept “Product” will refer to an assembled object.

A product carries key information and has multiple implications in various domains of the product lifecycle. For instance, a product designer would be more interested in the intended function of the product while an assembler might be concerned with possible assembly methods for the assembly of the product. The concept product is introduced to represent the information associated with it and its implications in assembly design and assembly process planning domains.

4.3.2 Product Version

In today’s competitive environment, a manufacturing system should be agile in order to accommodate the changes in products. A variation either triggered by the customer demand or product improvement requires changes in the product structure. Generally the term version is applicable where there occur variations or changes in the product over a time span (Elanchezhian, et al., 2005). For example, an updated instance and all the old instances of the same product would be referred to as versions of that product.

In context of the ARO, versions are very important as they carry product and process information. The term product version is introduced as a class in this research which refers to multiple versions of an assembled product. Information related to the history and past changes in a product can be represented through this class. Previously the concept part version has been used in the IMKS project to represent the versions of part for single piece part manufacturing. In IMKS, part version was described on the basis of definition provided by ISO/TS-10303-1022 (2004). However as the concept "product" has been used in this research to represent the assembled items, therefore the concept "product version" has been used for the assembly domain.

4.3.3 Product Family

The term product family refers to a group of products which share some common attributes. British standard BS 5191 (1975) describes "*product group" as a "number of products with one or more characteristics which make it convenient to combine them for planning and control processes*". The term product family is more frequently used in literature (Lit & Delchambre, 2003) as compared to product group, hence it has been used here in place of product group.

Various researchers have described product family concept from different common characteristics. For instance, grouping of products on the basis of, common function (Rekiek, et al., 1997) (Falkenauer & Delchambre, 1993), common form (Koren, 2010), similar routings and manufacturing processes (Fan & Liu, 1999), or common components (Danloy, et al., 1999) has been reported in the literature. These multiple interpretations can potentially imperil the interoperability across the product lifecycle domains, particularly the assembly design and assembly process planning domains.

As stated above, products can be grouped from assembly design perspective as well as from assembly planning point of view, hence the concept product family can be specialized to represent the assembly design and assembly process planning knowledge. Specialized classes in the form of "design product family" and "manufacturing product family" have been introduced as shown in figure 4.3. The knowledge associated with these two classes can be shared via the product family concept as discussed in section 3.4.2.2 of chapter 3.

The concept product family is subsumed under generic concept family and the MLO concept "Object" as shown in figure 4.3.

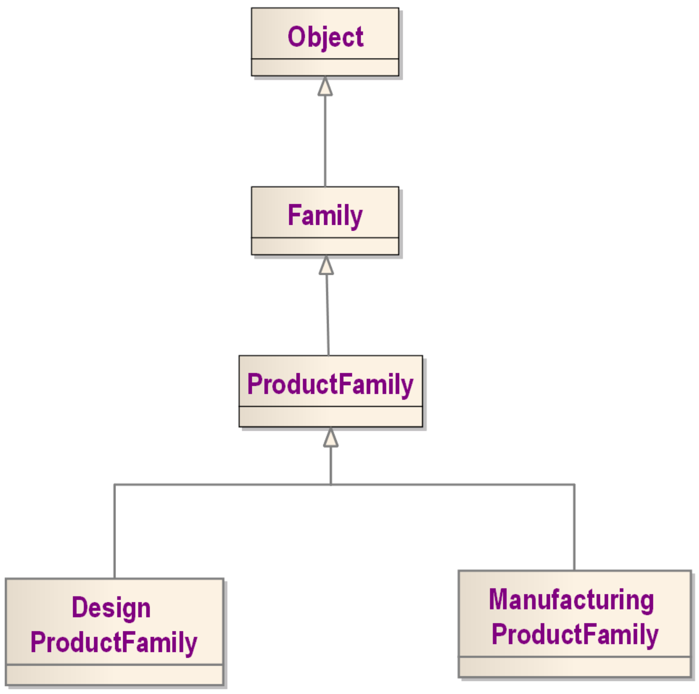


Figure 4.3: Product Family Concept Specialization

4.3.4 Shape Attribute

The shape attribute class represents the shapes of assembly components and assembly features that are important from the assembly domain point of view. This is because it helps to capture, represent and share the assembly knowledge associated with assembly components and features. For instance, the shape attribute class can support the capture of knowledge related to assembly fits as the latter are applicable to those assembly components and features which have circular shape e.g. holes and shafts. Further explanation of the shapes and their role in facilitating the assembly knowledge capture can be seen in section 5.3.

Shapes have been categorized based on the shape classification of Anderson (2007) who has provided a simple classification of closed 2D shapes and divided the latter into polygon and non-polygon shapes. The polygon shape is further specialized into triangle, square, pentagon, hexagon and other polygon shapes. Similarly the non-polygon shapes are mainly specialized into circular and non-circular shapes as shown in figure 4.4.

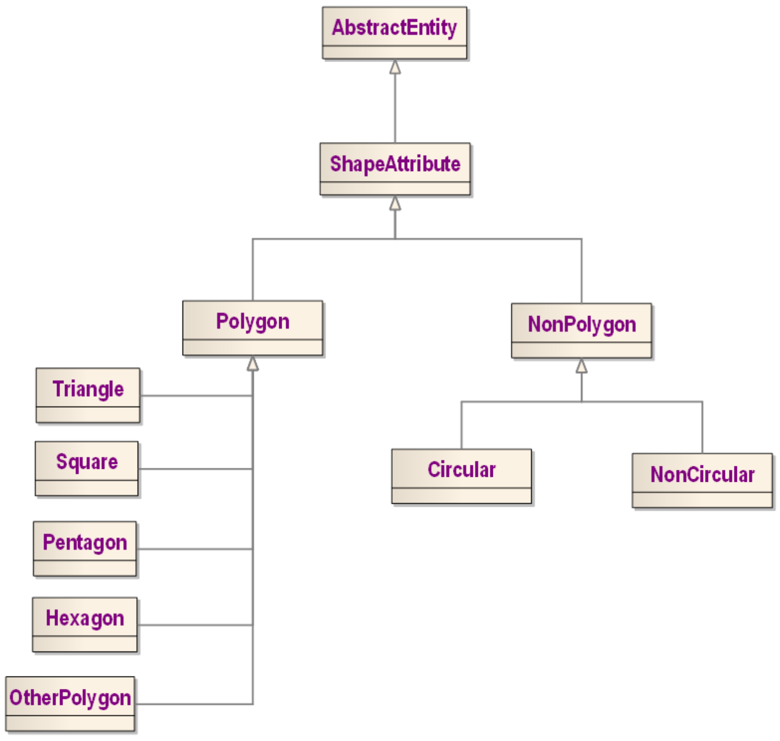
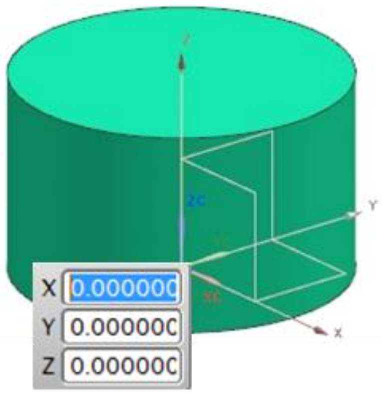


Figure 4.4: UML based representation of shape attributes

4.3.5 Spatial Location

The concept “spatial location” has been introduced in the ARO to capture and represent the location attributes of assembly components and features. This concept is particularly useful in the assembly domain because it determines the mating conditions of assembly features. For instance, when an assembly component having multiple assembly features is assembled with another assembly component having multiple assembly features, then the spatial location of these assembly features become very important. This implies that if there is a location incompatibility between the mating assembly features then it can result in component assemblability issues that will lead to the redesign of the mating components and features.

The location of an assembly object can be described by taking into account the reference point where it is located and the angle which it makes with a reference plane. These point and angular location concepts have been adapted from the concepts “reference line” and “reference point” described by Anjum (2011) to represent the co-ordinates of 3D objects. For instance, the extruded circular shape shown in figure 4.5 has a point location (X, Y, Z) at (0, 0, 0) where X, Y, and Z represent the distance of the object from a reference co-ordinate system. Similarly, the angular location of the object shown in figure 4.5 can be represented by assuming its central axis as a reference line and then measuring angles against X, Y, and Z axis. By taking the previous assumption, the angular location (Ax, Ay, Az) of the object shown in figure 4.5 would be (90, 90, 0) where Ax, Ay and Az represent the angles made by central axis of the object against X, Y and Z axis respectively. It is recognized that this kind of spatial location information is normally expected to come from the geometric modelling (e.g. CAD) systems.



X 0.00000
Y 0.00000
Z 0.00000

Figure 4.5: Representation of spatial location of an assembly feature which has circular shape attributes

As the spatial location concept has been described in terms of point location and angular location hence these concepts can be subsumed under spatial location class as shown in figure 4.6.

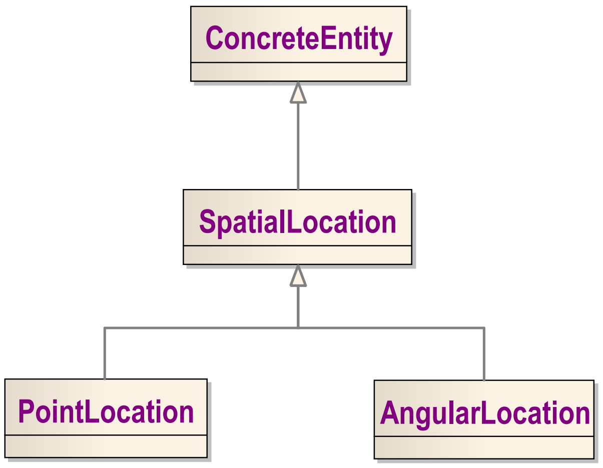
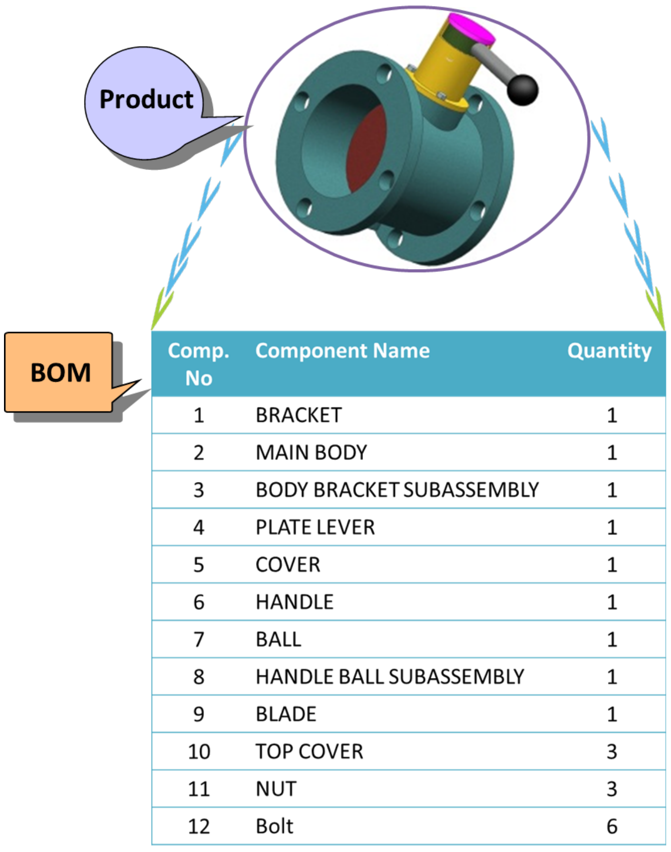


Figure 4.6: UML based representation of the concept spatial location

4.3.6 Bill of Materials (BOM)

Bill of Materials (BOM) is a core component of product lifecycle information management (Zhang, et al., 2010) and is a key concept for the assembly domain. BOM lists the components required to build a product as well as it carries information related to these components. There exist various definitions of BOM in the literature. For example, Chang et al. (1997) describe BOM as a list of components and raw materials along with their quantities. Jiao, et al. (2000) define BOM as a collection of items with parent child relationship. Zhong and NI (2010) believe that BOM not only comprises of list of items but also contains information associated with these items e.g. component number, the standard of tolerance etc. Based on the above mentioned description of BOM, it can be said that BOM carries two important concepts: components and their quantities. An example of BOM for butterfly valve assembly is shown in figure 4.7.



Comp. No	Component Name	Quantity
1	BRACKET	1
2	MAIN BODY	1
3	BODY BRACKET SUBASSEMBLY	1
4	PLATE LEVER	1
5	COVER	1
6	HANDLE	1
7	BALL	1
8	HANDLE BALL SUBASSEMBLY	1
9	BLADE	1
10	TOP COVER	3
11	NUT	3
12	Bolt	6

Figure 4.7: An example of BOM created for butterfly valve assembly

BOM is found in different forms and have multiple viewpoints (Chang, et al., 1997) (Jiao, et al., 2000). Although there exists various types of BOM in literature however Engineering Bill of Materials (EBOM), and Manufacturing Bill of Materials (MBOM) are the two most important categories (Vollmann, 1997) (Zhang, et al., 2010). EBOM comprises of list of items as described in assembly drawing (Xu, et al., 2008) (Tursi, et al., 2009) and is constructed on the basis of product design taking into account the functions of its components (Jiao, et al., 2000) (Chang, et al., 1997). However EBOM does not consider the manufacturing aspects hence it should not be used directly in assembly planning (Lee, et al., 2011) (Tursi, et al., 2009).

MBOM takes into account the manufacturing or assembly (process planning) aspects and is arranged according to the assembly plan of the product (Tursi, et al., 2009). MBOM comprises of list of all the materials along with their quantities required for a product to manufacture (Jones, et al., 2001) and is a different organization of EBOM which can be adapted for manufacturing purpose. As far as the structure of MBOM is concerned, it represents the hierarchical assembly groups based on the way they are assembled on shop floor (Chang, et al., 1997).

It is obvious from the above discussion that EBOM and MBOM are different in nature and have multiple implications in assembly design and assembly process planning domains. Hence BOM concept can be further specialized to EBOM and MBOM where EBOM and MBOM represent assembly design and assembly process planning domains respectively. The UML based concept specialisation of BOM is shown in figure 4.8. As BOM is recognized across product lifecycle domain hence it is termed as product lifecycle reference concept and has been subsumed under MLO concept Object.

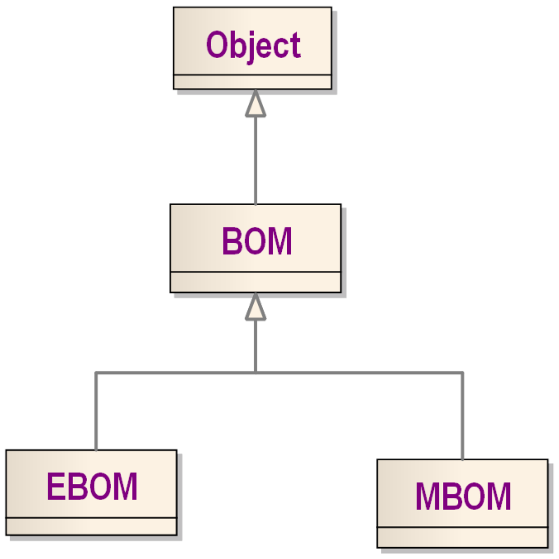


Figure 4.8: Concept specialization of BOM concept

One of the potential differences between EBOM and MBOM is the consideration of individual components. For instance, EBOM comprises of all the components which should be assembled to form a product (Hall and Jones, 2002), however components in MBOM of the same product may be different for a particular assembly system. This is because of some of the components which might have been purchased as subassemblies and are used as an individual component for product assembly. For example, if a car is being assembled at a particular assembly plant where engines are imported from somewhere else and are directly used in the assembly of car, then the assembler would not be interested in individual components of the engine. Hence MBOM for that particular assembly plant would be different from the EBOM.

From knowledge sharing point of view, these kinds of different interpretations of EBOM and MBOM may cause problems and consequently misunderstandings may develop. Hence it is important to define the structure of these BOM concepts which could be useful for a range of assembly systems. Once the semantics of BOM are captured, the knowledge associated with the assembly components can be shared across the assembly design and assembly process planning systems. A detail example case of multiple interpretations of MBOM concept has been discussed in chapter 5.

As discussed earlier, a BOM can be represented by a list of components and their quantities. Hence BOM can be related with the concept “ComponentList”. ComponentList represents the list of components required in the assembly of end product like butterfly valve and is a specialized concept of the foundation concept “list”. A UML based lightweight representation of BOM concept is shown in figure 4.9. The figure shows that BOM has a “hasComponentList” relationship with the concept “ComponentList”. Similarly, the concept “ComponentList” has “hasComponent” relation with the concept “Component”. BOM concept is subsumed under the MLO concept Object.

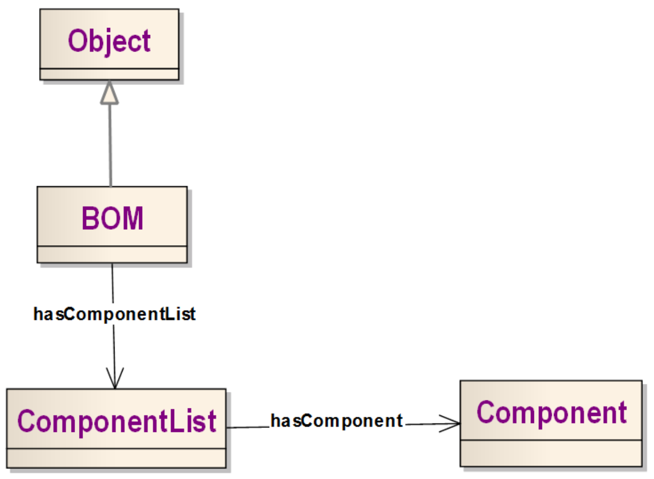


Figure 4.9: UML based lightweight representation of BOM concept.

4.3.7 Bill of Process (BOP)

The concept Bill of Process (BOP) has been identified to list the sequence of assembly processes for the production of a particular product. This concept has not been widely used in the literature although some researchers described it in their own perspectives. For instance, Zeng and Bin (2004) describe BOP as a data structure which represents the procedure of a production process while Park and Simpson (2008) describe it as a list of unit level activities in a production process and the corresponding time spent on each activity. On the similar lines Bauer, et al. (1991) defines BOP as a concept which "*describes the process steps involved in the production of a product*". From these descriptions, the author takes the view that a BOP basically describes the list of processes required to manufacture a unit product and can be used to represent the process information in the assembly domain. As BOP enlists the processes where the latter are instance of MLO class "Event", hence BOP can also be specialized under the concept "Event".

4.3.8 Bill of Resources (BOR)

Bill of Resources (BOR) has been introduced in the ARO to represent the required assembly resources to carry out the assembly processes for a particular product. Various researchers have defined BOR from their own view points. For instance, Hill (2012) describes BOR as "*a list of the machine time and labour time required to make one unit of a product*". Zobolas (2008) refers BOR as a database which represents the information related to the total required resources for the production of one unit of product. Sehgal (2009) believes that BOR represents resources just like BOM represents components to make a product. In the context of this research work and from the existing literature, BOR can be described as a list of resources required to produce a unit product. BOR can also be used to link the quantity and utilization time of the resources used.

4.3.9 Dimension

The term dimension is widely used in the product design and related domains to represent the measurements of various attributes of a product. For instance, Krulikowski (1998) defines dimension as "*a numerical value expressed in appropriate units of measure and used to define the size, location, orientation, form, or other geometric characteristics of a part*". In general dimensions are expressed in length and angular measurement functions. In the literature, various dimension related concepts e.g. linear dimensions, radial and arc dimensions, and angular dimension have been used to represent different kinds of dimensions (Byrnes, 2011) (Bennett and Siy, 2009) (Taylor, 2005). The concept dimension has also been used in various assembly domain related software systems e.g. Boothroyd's DFA system, NX 7.5, and Teamcenter, where it has been used to represent the measurement functions like length and angle.

The author takes the view that various length related dimensions can be classified under the concept length dimensions whereas angle related dimensions can be accommodated under the angular dimension concept. In this way both length and angle can be represented under the main concept dimension. Further classification of length and angular dimensions is possible however this is not required in the scope of this research. Examples of length dimensions can be: diameter of a hole, width across flats of a hexagonal nut, product height etc. Similarly, examples of angular dimensions may include the angle between the central axis of two different hole assembly features, angle between the reference X, Y, Z axis and the central axis of a shaft feature.

The concept dimension has been specialized from the concept quantity as shown in figure 4.10. As discussed in section 4.2, quantity is an MLO concept which represents all the measurement related information however the concept dimension has been introduced in the ARO to represent the length and angular measurement information only. The properties: length dimension and angular dimension have been subsumed under the dimension concept, which represent the length and angular measurements respectively. For instance, the length dimension can be represented in length units e.g. mm, cm etc. and angular dimension can be represented in angular measurement units e.g. degree, and radian. More details about the use of dimension concept can be found in chapter 5, and 6.

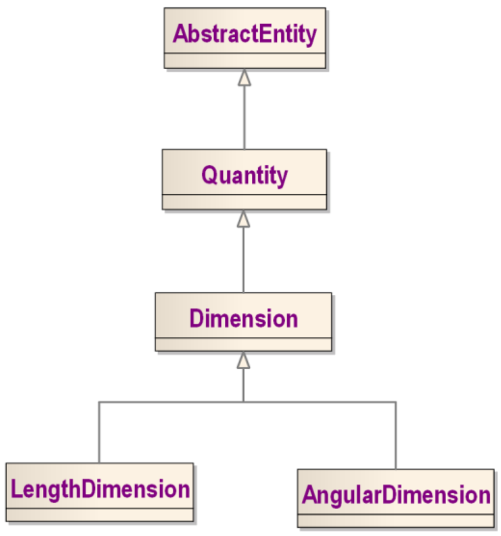


Figure 4.10: UML based representation of dimension

4.3.10 Tolerance

Before the assembly components and/or assembly features are brought in for the assembly purpose they are manufactured using different techniques based on the required dimensional accuracy and the tolerances. A range of dimensions or sizes are specified to permit the manufacturing of these assembly components and features. This range of sizes is determined by the basic size or dimension of the assembly component or feature and the tolerance. A basic size or dimension of the component/feature is the theoretical size or dimension of that component/feature from which limits (minimum and maximum allowable dimensions) are derived (Jensen, et al., 2001).

Whereas a tolerance is the maximum lower and upper deviations from the basic size (Nof, et al., 1997). Therefore a tolerance has two quantities: the lower quantity and the upper quantity. The upper quantity is added to the basic dimension to determine the maximum allowable dimension whereas the lower quantity is added to the basic dimension to get the minimum allowable dimension. The concept tolerance is important from the assembly point of view and can be used to deduce the information related to assembly fits and other mating relationships. The information related to mating relationships can be further exploited to determine the resulting assembly processes and the availability of assembly resources and manufacturing facility. A detail exploration of the concept tolerance has been discussed in chapter 5.

4.3.11 Assembly Component

The concept assembly component is a specialized form of the product lifecycle concept “component” and has been identified to support the representation and sharing of assembly knowledge. The term component has been found in the literature with different meanings and interpretations from its structure point of view. However majority of the sources explored, describe component either a single piece part or a subassembly. For example, standard ISO/TC 10303-224

(2003) defines component as: "*The component specifies either a Single_piece_part or another Manufactured_assembly used to define an assembly*". In the same way Molloy, et al. (1998) and Lohse (2006) describe component as either a single piece part or a subassembly used for building a product. Similarly, Siemens NX 7.5 assembly modeller and Teamcenter 8 also identify component as a single piece part or a subassembly. However Boothroyd's DFA 9.4 software system does not use the term component, rather it uses the terms part and subassembly to build assemblies.

Based on the above discussion, the author will use the following informal interpretations of the concepts, "Part", "Subassembly", and "Component" as follows.

(a) Part: refers to a single piece item used to build a product.

(b) Subassembly: refers to a collection of parts assembled together.

(c) Component: refers to either a part or a subassembly.

The above discussion about the component highlights its structure in terms of part and subassembly. However it can have different implications from the perspective of different domains. As the term component is a "product lifecycle" term and it can be applicable to domains such as service and disassembly as well so it has been specialized into assembly component as shown in the figure 4.11. The concept assembly component may have further different implications for assembly design and assembly process planning domains. For instance, in today's collaborative environment, functions of a manufacturing plant are diversified across the globe and are also heavily dependent on each other. The product designer would only be concerned with all those assembly components which are important from functional viewpoint whereas an assembler would only be worried about the assembly components which are significant from assembly viewpoint.

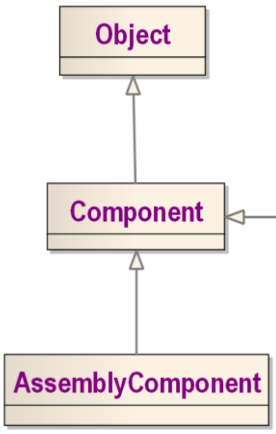


Figure 4.11: Lightweight UML based representation of assembly component.

4.3.12 Assembly Feature

Assembly feature is a specialized class of the generic concept feature. The Oxford dictionary defines feature as "*a distinctive attribute or aspect of something*". However most definitions of feature, found in the literature review, relate it with product lifecycle domain especially the design and manufacture. For example, Pratt and Wilson (1985) define feature as "*a region of interest on the surface of a part*". Rosen (1993) describes feature as "*meaningful abstractions of geometry that engineers use to reason about components, products, and processes*". Lenau and Mu (1993) believe that "*features are information sets that refers to aspects of form or other attributes of a part"*. It is obvious from the above discussion that a feature is a physical constituent of a component in context of design and manufacture but still it should not be constrained to a particular domain as it has been considered a generic concept in this research.

A more specialized class "FormFeature" can be introduced to replace feature. Form features are the "*features producing volumes*" (Rosen, 1993) or the "*features that relate to the shape or form of the part*" (Roller, 1989). However form features can carry design information and/or the manufacturing information (Hounsell, 1998) as well as other such domains. Hence form features may not be applicable to any particular product lifecycle domain and can be declared as a generic concept.

Form features can relate to products and resources therefore they can specialized into product feature and resource feature (the term resource feature is further explained in section 4.3.19). A product feature can have further implications in the design and manufacturing domains and could represent different aspects of these domains. For instance, a product feature for the design domain may be described as "*a parameterised geometrical entity used for building the CAD model*" (Molloy, et al., 1998) and may have the functional information as well. While product feature in context of the manufacturing domain can be described as the "*interpretation and, most importantly, the combination of form features from the viewpoint of manufacturing, assembly and inspection*" (Krause, et al., 1993) or "*a parameterised entity linked with one or several alternative manufacturing methods*" (Molloy, et al., 1998).

The concept product feature which carries design and manufacturing information can be further specialized into single piece part feature and assembly feature. The single-piece-part-feature concept deals with the design and manufacture of single piece parts and therefore it is not in the scope of this research. The concept assembly feature which carries assembly design and assembly process planning information is explored further in this research for the representation and sharing of assembly knowledge.

The assembly feature has different implications for assembly design and assembly process planning domains as it is evident from the definitions of assembly feature found in the literature as well. For example, Deneux (1999) defines assembly feature from design perspective as "*a generic solution referring to two groups of parts that need to be related by a relationship so as to solve a design problem".* Molloy, et al. (1998) describe assembly feature from assembly planning point of view as "*a parameterised entity linked with one or several assembly methods*". Comaa et al. (2003) define assembly feature from various viewpoints as "*any topological, geometrical, technological or functional information assigned to a face, a part or a sub-assembly, whose presence is inherent to the assembly process*".

In this research assembly feature has been specialized into various classes which can support the capture and sharing of assembly knowledge across the assembly design and assembly process planning domains. Hole and shaft assembly features provide basis for limits and fits, and are commonly found in literature. Hole assembly feature represents the female assembly feature and whereas shaft assembly feature represents the male assembly features. Hole and shaft assembly features have been explored in details in chapter 5 and 6.

Other subsumptions of assembly feature class include the plane mate assembly feature (Holland, 200) and alignment assembly feature (Shah, 2001). Plane mate assembly features represent those assembly features which have plane to plane mating relations as shown in figure 4.12a. The alignment feature represents the assembly features where the latter have their axis aligned with each other as shown in figure 4.12b.

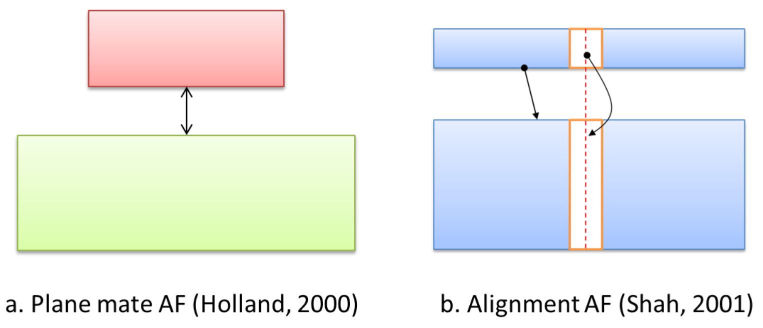

a. Plane mate AF (Holland, 2000)
b. Alignment AF (Shah, 2001)

Figure 4.12: Examples of plane mate and alignment assembly features

Other specializations of assembly features are handling assembly features (Holland, 1997) and tooling assembly features. Holland (1997) proposed that assembly feature could potentially be specialized into handling feature and connection feature. The handling feature represents the handling information while connection feature represents mating or connection information. Figure

4.13 shows two assembly components which have handling and connection features. However in this research, the concept connection feature has not been used and instead various instances of connection feature e.g. hole, shaft, plane mate feature, and alignment feature have been directly subsumed under the assembly feature concept.

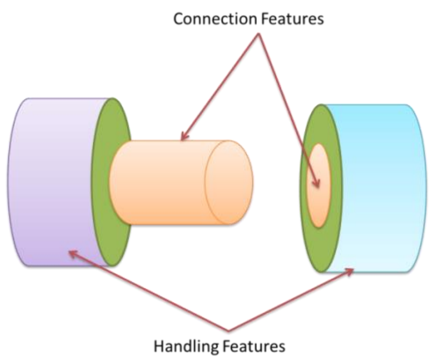


Figure 4.13: Handling and connection features

The concept tooling assembly feature has been used to represent those assembly features which actually interact with assembly resource features to accomplish the assembly task. An example of tooling assembly feature is shown in figure 4.14.

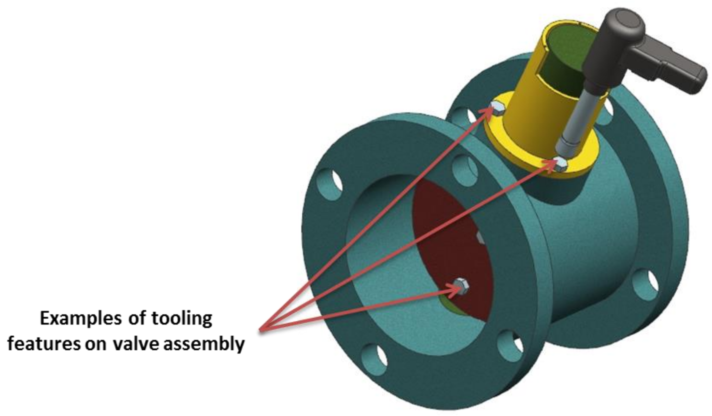


Figure 4.14: Examples of tooling features on the butterfly valve assembly

The nut runner (assembly resource) shown in figure 4.14 interacts with the tooling features to perform the fastening process. The concepts of handling and tooling features could be useful in the scenarios where products interact with assembly resources.

A complete concept specialization of assembly feature is shown in figure 4.15 where the assembly feature has been specialized into hole, shaft, plane mate, alignment, handling and tooling assembly features.

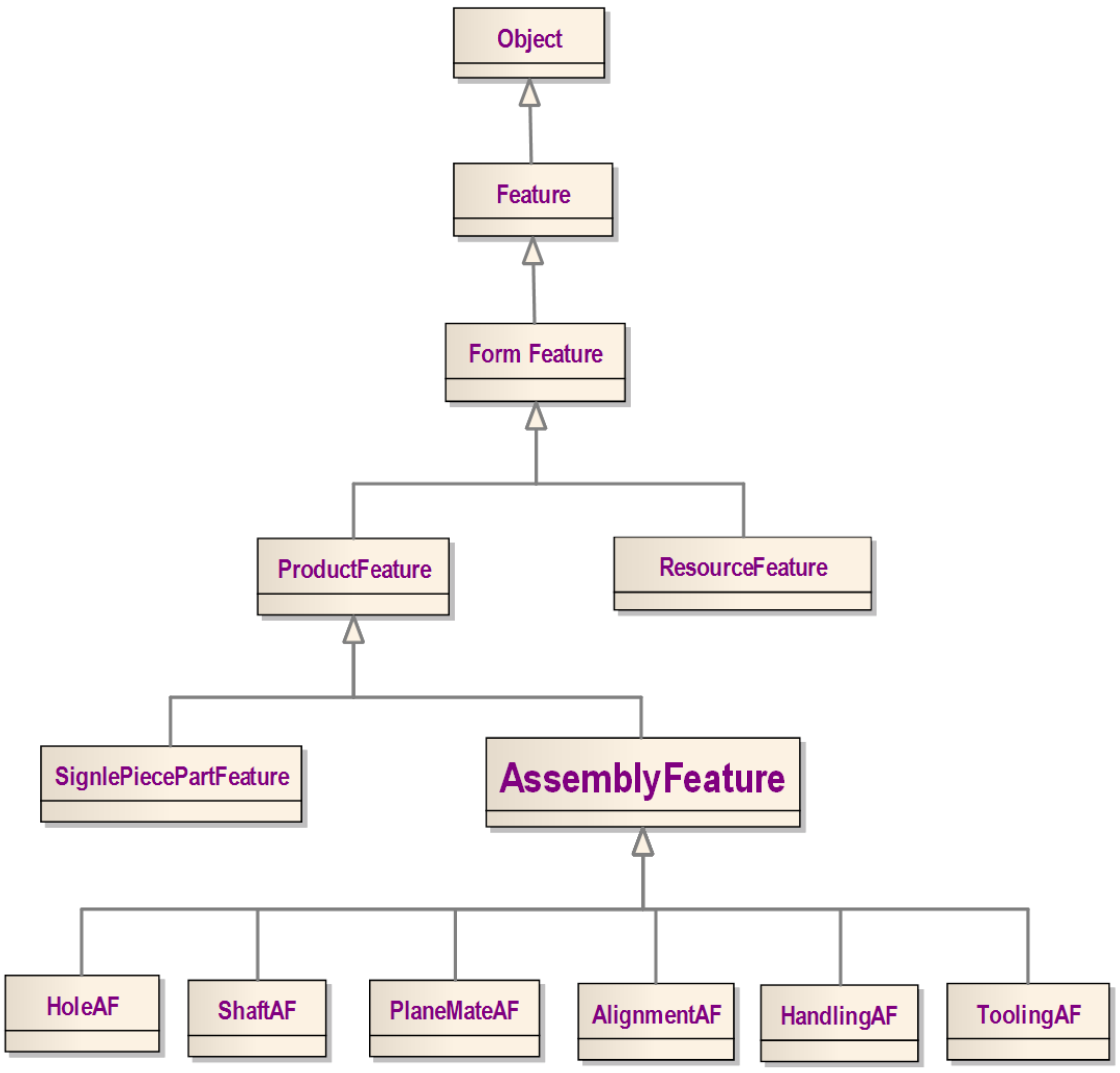


Figure 4.15: Concept specialization of assembly feature

4.3.13 Tolerance Type

As mentioned in section 4.3.10, the mating assembly features and components are specified with allowable tolerance quantities. These tolerance quantities depend upon the basic dimension of the assembly features and components. The tolerance quantities can be user specified or specified by different tolerance standards. One such tolerance standard is BS 4500 (1969) which provides different tolerance grades such as H8, H7, f7, k6, and p6. These tolerance grades vary with the basic dimension of assembly features and components.

In this research a concept “Tolerance Type” has been used to support the representation of these kinds of tolerance specific grades. It is important to understand that the concept “Tolerance Type” is different from the concept “Tolerance” as the “Tolerance Type” represents types (grades) of tolerance quantities whereas the “Tolerance” represents the tolerance quantities in some measurement units. A detailed exploration of the “Tolerance” and “Tolerance Type” concepts has been explained in section 5.3 of chapter 5.

4.3.14 Assembly Operation

Dictionary of engineering defines operation as “*a job usually performed in one location and consisting of one or more work elements*”. Operation is a generic term and can be applicable to multiple domains. The concept assembly operation can be used to represent the operations performed in the assembly domain and can be subsumed under the concepts operation and manufacturing operation as shown in the figure 4.16.

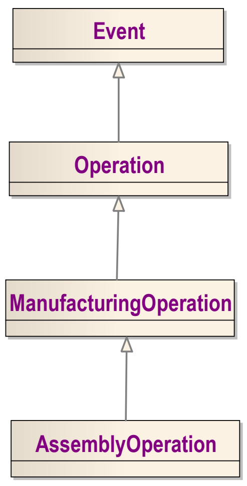


Figure 4.16: UML based representation of assembly operation

The author takes the view that an assembly operation may have one or more assembly processes. For instance, consider an assembly line where a product is being assembled. This assembly line can have various operations e.g. mounting and unmounting of a product on the assembly line and these operations could have assembly processes like fastening etc.

4.3.15 Assembly Process

Assembly process is a specialized class of the generic concept process. A process is "*an ordered set of activities with a defined goal*" (Lohse, 2006). The defined goal could be anything e.g. to get a degree, to win a gold medal or to make a product for consumers. This implies that the concept process is a generic concept which can be applicable to any domain. A more specialized concept "manufacturing process" is commonly used in context of the manufacturing domain. A manufacturing process has "creation of product" as its defined goal (Lohse, 2006). However a more appropriate definition of manufacturing process is provided in ISO 18629-43 (2006) which defines manufacturing process as "*structured set of activities or operations performed upon material to convert it from the raw material or a semi-finished state to a state of further completion*".

The above description of process and/or manufacturing process is related to a set of activities which are all time dependant. Therefore the generic concept process has been subsumed under the MLO concept "Event" instead of "Object" as shown in figure 4.17. Assembly process concept is introduced here as the most specialized concept in the hierarchy of figure 4.17 in order to represent the assembly process planning related information. The assembly process class is associated with the resource class as the latter helps to perform assembly processes.

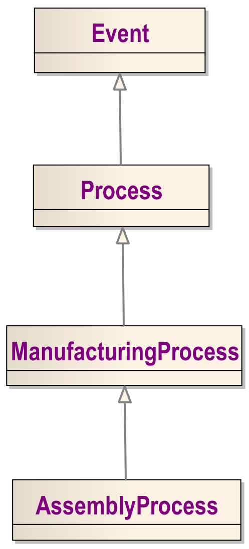


Figure 4.17: Lightweight representation of assembly process concept

4.3.16 Step

An operation may be further sub-divided into smaller elements. Each of these smaller elements is named as step in this research. The concept of step is introduced to facilitate the capture and representation of assembly process planning semantics. The concept of step has been defined by keeping in view the contact between a resource feature and a product feature. This implies that each time the resource feature makes a contact or mates with an assembly feature; it results in unique step each time.

One of the potential issues in the representation of assembly concepts is the fact that they have multiple representations and implications, and they are understood in different ways by different companies and the software systems. For example, the concepts assembly operation, assembly process and step have subtle similarities and differences. In particular the concepts assembly operation and assembly process are often confused with each other.

This research takes the view that the assembly operation concept is product dependant and may be different for different products. However the concept assembly process is considered independent of the product and different products may have same assembly processes. The concept step is a sub-activity of an operation and an operation may have multiple steps. For example consider the valve assembly shown in figure 4.18. Examples of assembly operations of this valve assembly could be: attach bracket with the main body, assemble cover with the bracket and main body. The possible assembly processes could be: welding, and fastening. Examples of step for the assembly operation "assemble cover with the bracket and main body" may include fastening of bolt1, fastening of bolt2 and fastening of bolt3.

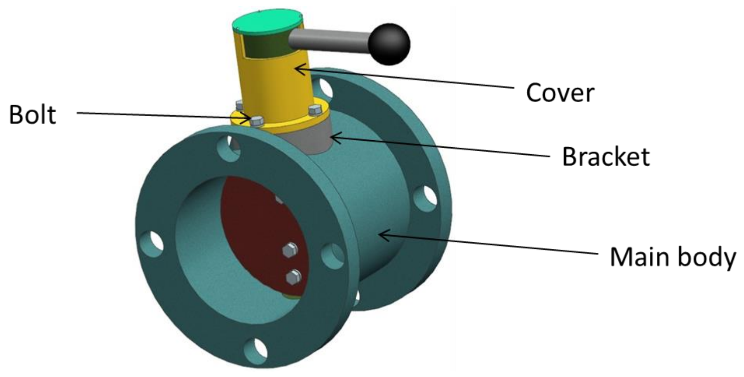


Figure 4.18: Complete butterfly valve assembly

4.3.17 Assembly Resource

Assembly resource is a specialized class of the generic concept resource and represents the resource information related to the assembly domain. In TeamCenter 8, the term resource refers to the items which are used to execute an operation or to perform a process. However in general, the term resource is open to multiple interpretations and meanings. For example, ISO 19115 (2003) defines resource as "*assets or means that fulfils a requirement*". Hence the term resource can be attributed to anything which satisfies requirement of any domain.

A more specialized class of resource is manufacturing resource. The term manufacturing resource has been explained by Usman (2012), who used it for the part machining. However manufacturing resource class may include the assembly and other manufacturing process resources as well. Hence the concept manufacturing resource can be further specialized to "assembly resource" where the later only represents the assembly related information.

A classification of assembly resources developed by Dorador (2001) is shown in figure 4.19. Assembly resource has three major subclasses named as transporting equipment, assembly equipment, and storage equipment. These three subclasses are further broken down to other subclasses as shown in figure 4.19. However, this classification is missing the concepts like human resources and the manufacturing facilities e.g. assembly plant etc. which can be considered as assembly resources as well (Lemaignan, et al., 2006).

This research work considers the human resources as a part of assembly resource concept however the term manufacturing facility has been considered as a different class and is discussed in the upcoming section 4.3.18. The concepts found in Dorador (2001)'s classification of assembly resource are domain specific and are therefore not included in assembly reference ontology. A lightweight representation of specialization of assembly resource concept for the assembly reference ontology is shown in figure 4.20. Assembly resource is shown as the most specialized class hence all the instances of assembly resources can be asserted directly under this class.

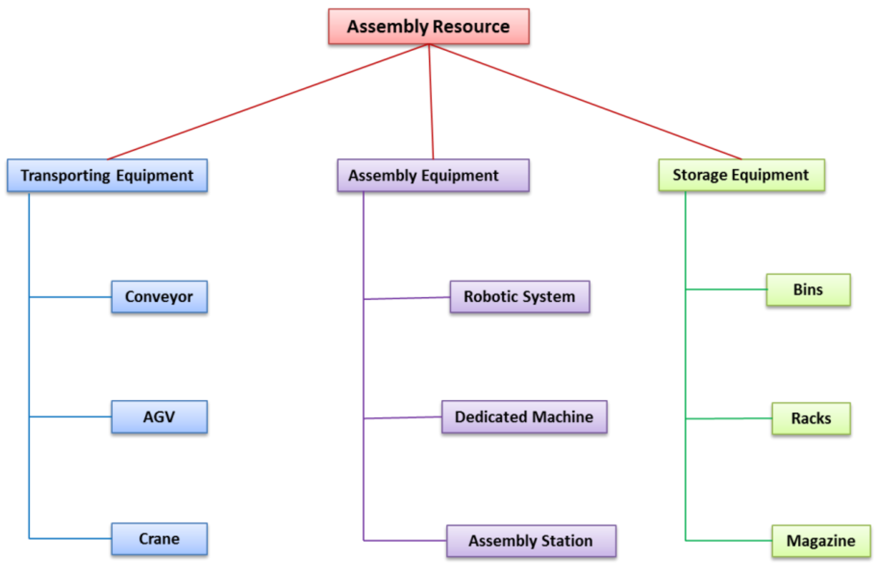


Figure 4.19: Assembly Resource Classification adapted from (Dorador, 2001)

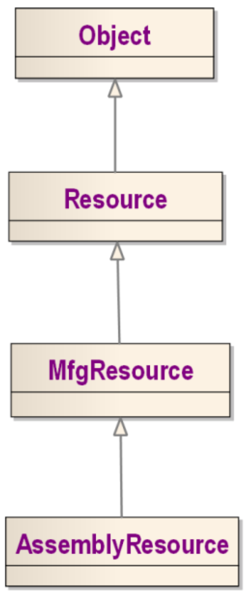


Figure 4.20: UML lightweight representation of assembly resource concept

4.3.18 Manufacturing Facility

The term manufacturing facility is commonly understood as a place for manufacturing of products. The structure of the concept manufacturing facility shown in figure 4.21, has been initially developed in MOSES project (Molina, et al., 1995). This was further extended by Zhao et al. (1999) with the addition of the Enterprise class. In the original model, the concept facility was used instead of manufacturing facility. However later on Usman (2012) has replaced the term facility with manufacturing facility. It is argued here that the same term can also be used for assembly reference ontology as the concepts like shop, cell station are commonly understood in manufacturing assembly as well. A more general class facility is introduced between the MLO concept Object and manufacturing facility. This would help if the ARO needs to extend or interact with other domains.

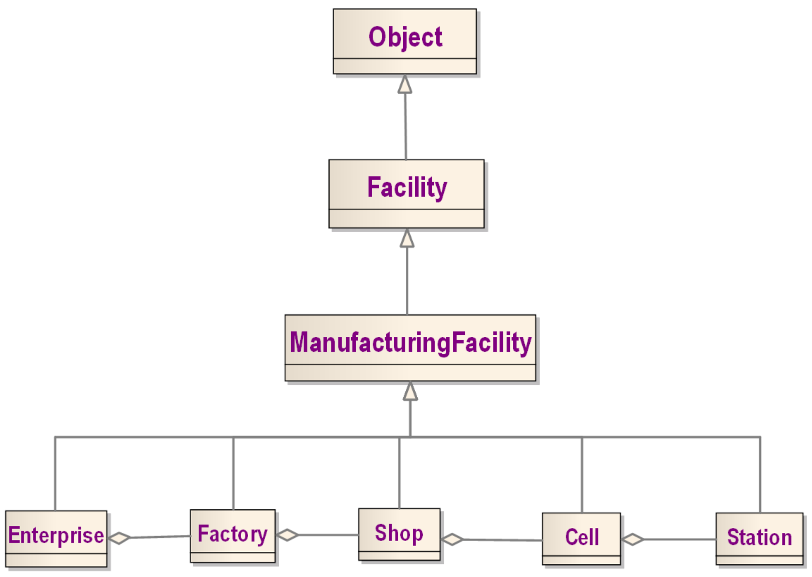


Figure 4.21: Lightweight representation of manufacturing facility adapted from (Zhao, et al., 1999)

As explained in Zhao et al. (1999), one or more stations aggregate to form a cell and one or more cells combine to form a shop. Similarly one or more shops aggregate to a factory and one or more factories combine to form an enterprise.

This structure of manufacturing facility can help to represent assembly process planning knowledge which can then be shared across other domains including the assembly design.

4.3.19 Assembly Resource Feature

As discussed in section 4.3.12, the concept of assembly feature supports the representation and sharing of assembly knowledge associated with the product. However in scenarios like flexible assembly line where it is important to find out the suitability of available assembly resources for the assembly of different variations of a product, it becomes imperative to take into account the resource product mating concepts and their relationships. In such situations, the concept of assembly resource feature can be used for resource features just like their counter part assembly features. In figure 4.22, an example of assembly resource feature is shown. This research work takes the view that the assembly resource feature refers to that resource feature which makes a contact with the product feature for purely assembly purpose. The hexagon socket of the nut runner shown in figure 4.22 can make contact with the product feature therefore it can be referred as assembly resource feature.

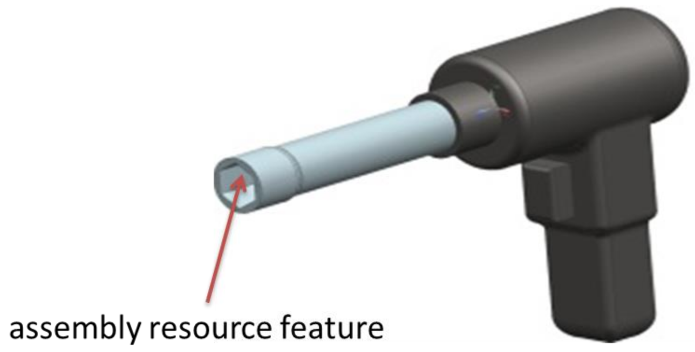


Figure 4.22: An example of assembly resource feature on a nut runner

The generalized and specialized concepts of assembly resource feature are shown in figure 4.23. Assembly resource feature concept has been subsumed under the more generic concepts: resource feature and manufacturing resource feature. The specialized classes: handling resource feature and tooling resource feature are the handling and tooling features on the assembly resource. A handling resource feature is the one which is used for handling activities e.g. moving, transportation etc. while a tooling resource feature is the resource feature which is used for assembly activities e.g. fastening, press fitting etc.

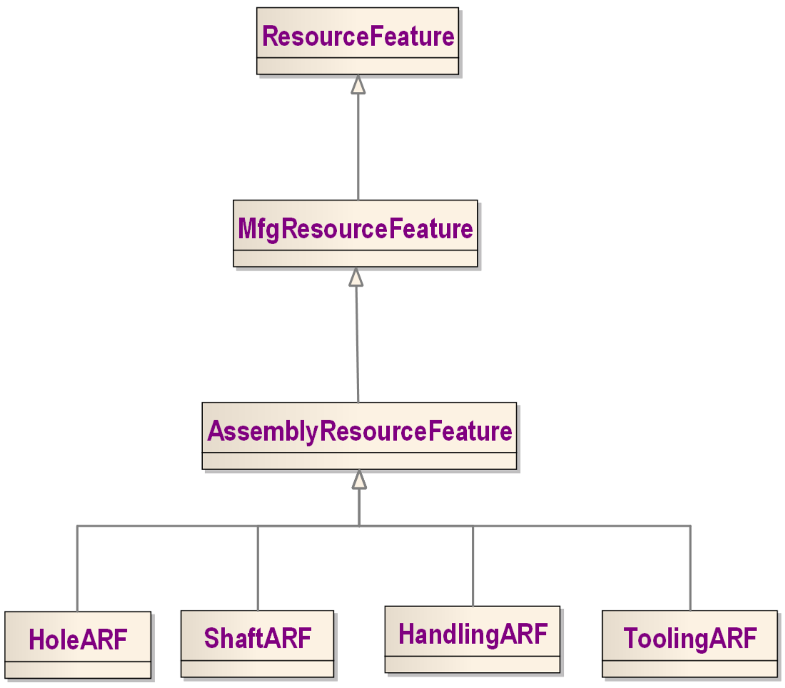


Figure 4.23: UML based representation of assembly resource feature.

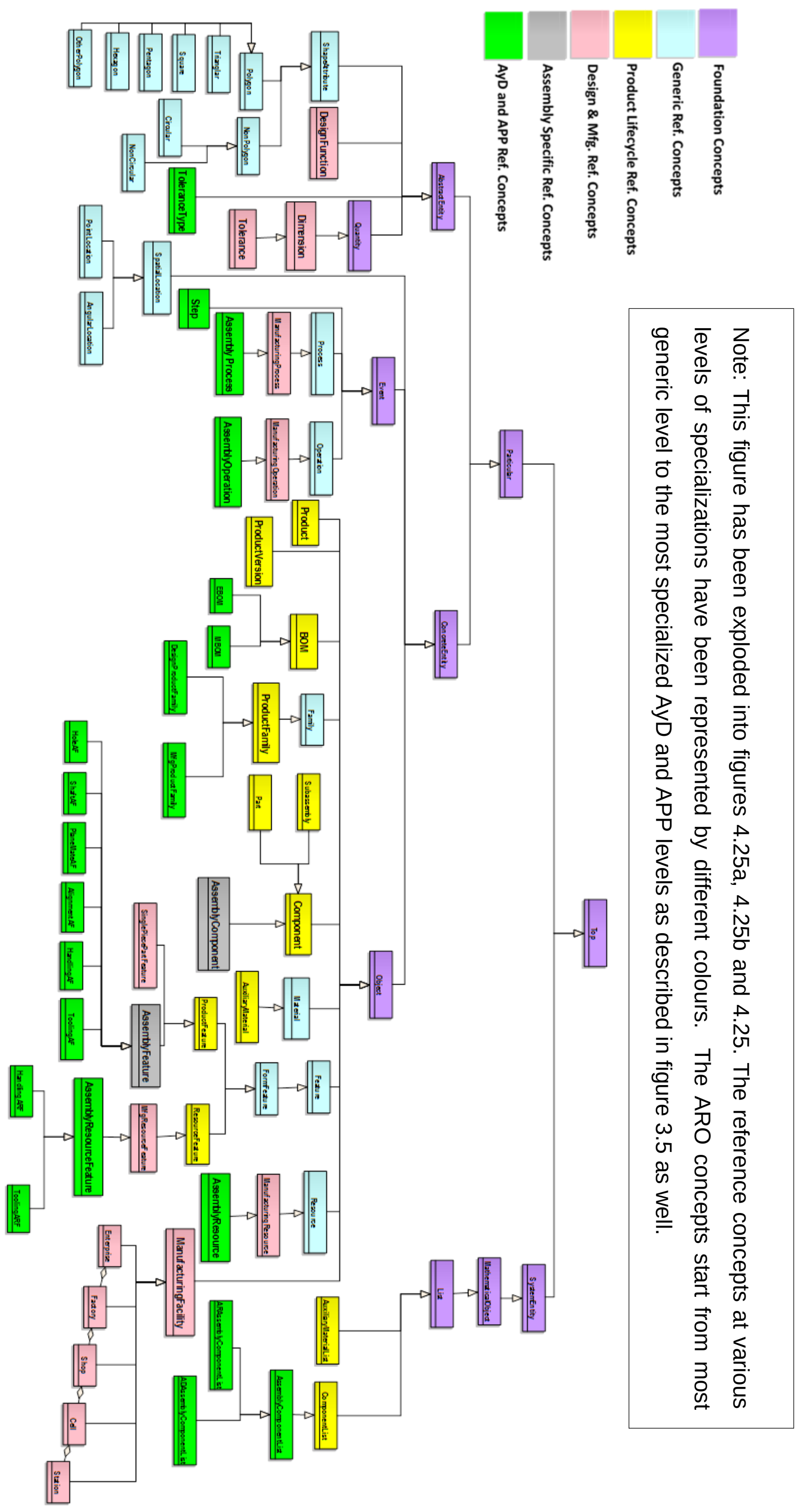
4.4 Combined Concept Specialization of ARO Concepts

Combined concept specialization of ARO is shown in Figure 4.24. However as the text in the figure for combined concept specialization is not legible so it has been broken down into three enlarged figures: figure 4.25a, figure 4.25b, and figure 4.25c. The concepts represented in purple boxes are the foundation concepts discussed in section 4.2. These concepts are then further specialized and have been linked with the assembly reference concepts via generalization relationships shown in the figure.

The concepts represented by sky blue boxes are the generic reference concepts for the ARO which have also been discussed in section 3.5 of chapter 3. The concepts represented by yellow boxes are the product lifecycle concepts whereas the concepts represented by pink boxes are the design and manufacturing reference concepts. Assembly specific reference concepts are represented by grey colour boxes, and assembly design and assembly process planning reference concepts are represented by green boxes as shown in the figures 4.24 and 4.25.

Concepts related to lists and its subsumptions in the form of auxiliary materials list and component list are also shown in figures 4.24 and 4.25c. These concepts have been represented by using the foundation concept ‘list’ that represents a group of ‘Top’ which may include the objects and events as well. These list related concepts are explained in section 5.2 of chapter 5.

Figures 4.24 and 4.25 give a complete overview of all the classes in the assembly reference ontology and their generalization relationships. However other than generalization relations, these classes also have inter-class relationships as well. These inter-class relationships are discussed in the following section.



Note: This figure has been exploded into figures 4.25a, 4.25b and 4.25. The reference concepts at various levels of specializations have been represented by different colours. The ARO concepts start from most generic level to the most specialized AyD and APP levels as described in figure 3.5 as well.

Figure 4.24: Combined concept specialization of the ARO

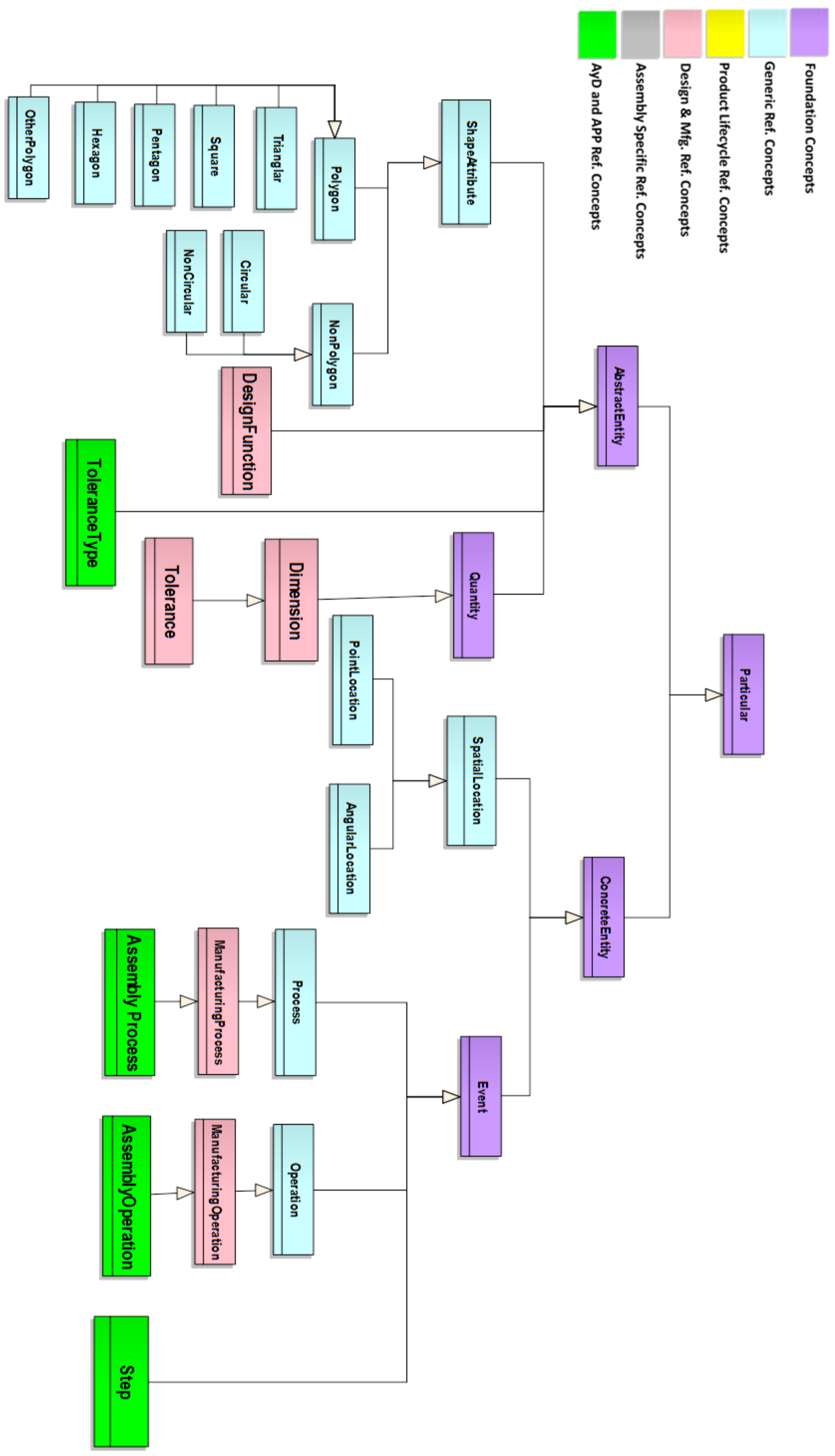

Foundation Concepts
Generic Ref. Concepts
Product Lifecycle Ref. Concepts
Design & Mfg. Ref. Concepts
Assembly Specific Ref. Concepts
AyD and APP Ref. Concepts
Particular
AbstractEntity
ConcreteEntity
ShapeAttribute
Polygon
Triangular
Square
Pentagon
Hexagon
OtherPolygon
NonPolygon
Circular
NonCircular
DesignFunction
ToleranceType
Quantity
Dimension
Tolerance
SpatialLocation
PointLocation
AngularLocation
Event
Process
ManufacturingProcess
Assembly Process
Operation
ManufacturingOperation
AssemblyOperation
Step

Figure 4.25a

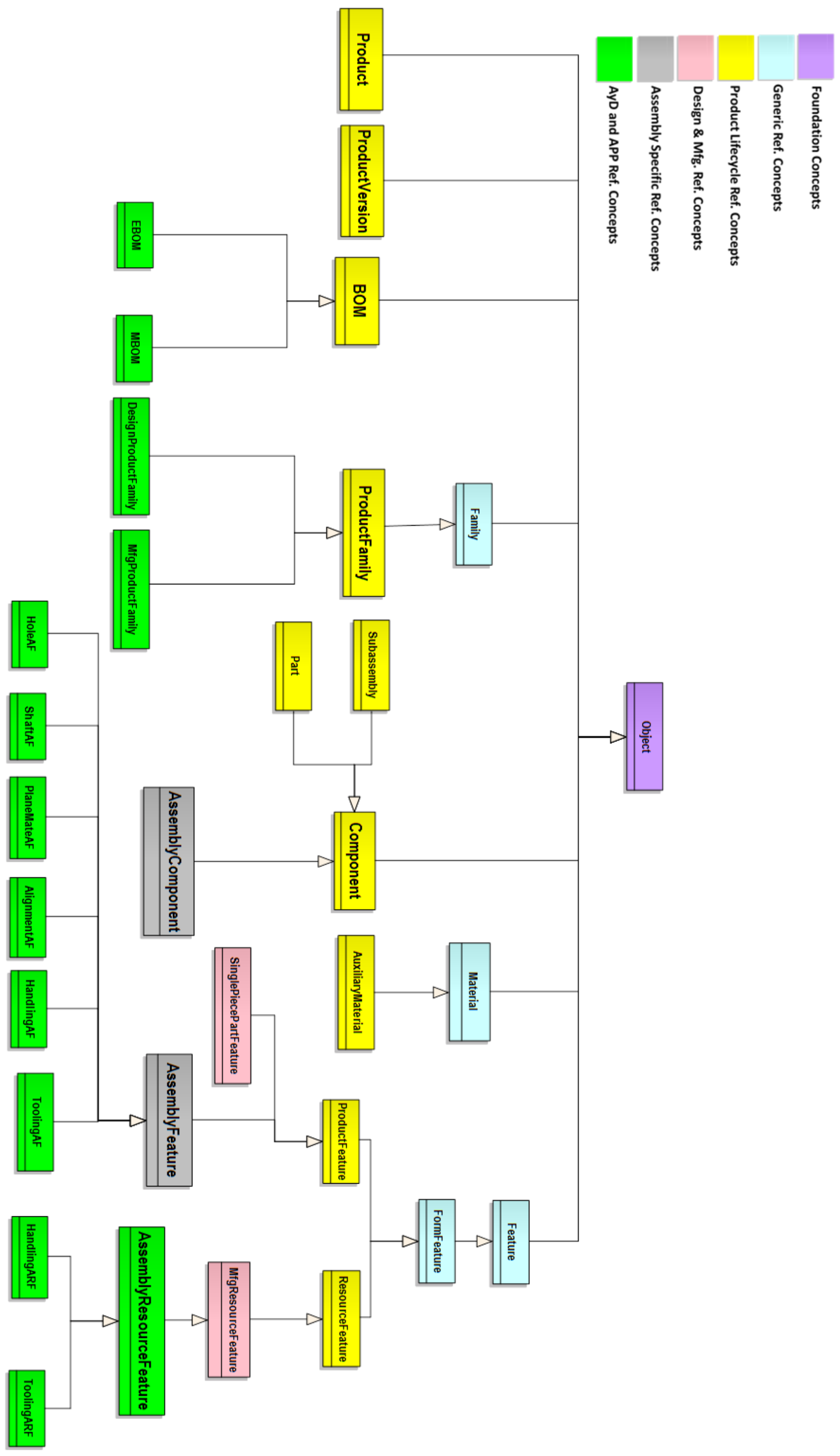

Product
ProductVersion
EBOM
MBOM
BOM
DesignProductFamily
MfgProductFamily
ProductFamily
Family
Part
Subassembly
Object
AssemblyComponent
Component
HoleAF
ShaftAF
PlaneMateAF
AlignmentAF
HandlingAF
ToolingAF
AssemblyFeature
SinglePiecePartFeature
AuxiliaryMaterial
Material
ProductFeature
FormFeature
Feature
HandlingARF
ToolingARF
AssemblyResourceFeature
MfgResourceFeature
ResourceFeature
AyD and APP Ref. Concepts
Assembly Specific Ref. Concepts
Design & Mfg. Ref. Concepts
Product Lifecycle Ref. Concepts
Generic Ref. Concepts
Foundation Concepts

Figure 4.25b

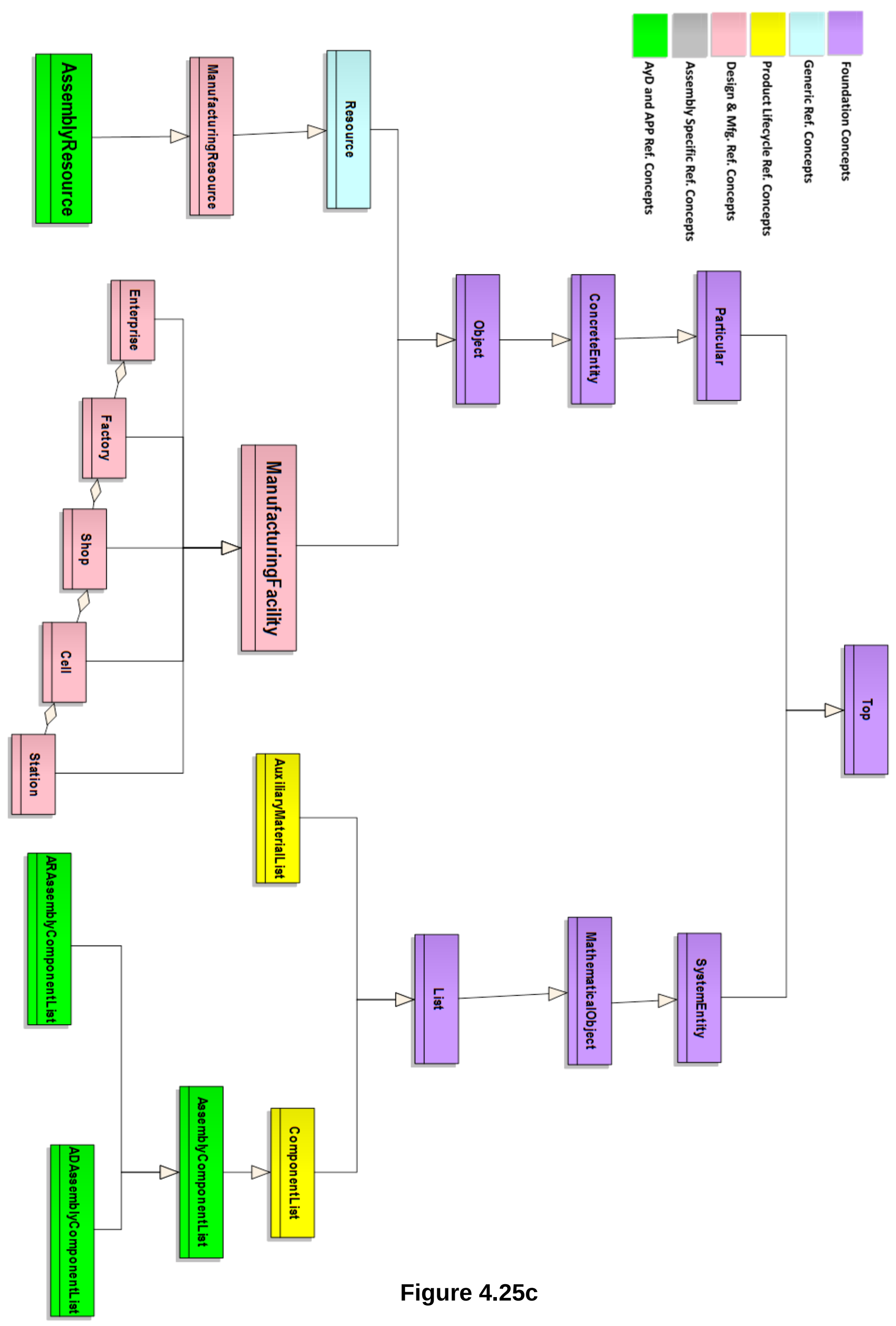


Figure 4.25c

Figure 4.25: Enlarged views of figure 4.24

4.5 Inter-Class Relations in ARO Concepts

This section discusses the inter-class relationships in the ARO which are also shown in figure 4.26 as well. Inter-class relationships help to represent the knowledge associated with the concepts and are extensively used in axioms and queries to infer, constrain and retrieve the assembly knowledge. Key inter-class relations are explained below.

In section 4.3.1, the class “Product” was informally defined as an assembly of items. Hence the product has a set of items or components as part of its definition. So a relation called “hasComponent” has been defined to link the “Product” class with the “Component” class. Similarly a product would also have BOM where the latter is actually a list of components. A relation “hasBOM” is established between product and BOM classes. Similarly the relations “hasBOP” and “hasBOR” has been used to link the product with the BOP and BOR respectively. As products are upgraded, they are assigned different product versions. Hence a relation “hasProductVersion” is placed between “Product” and “ProductVersion” class.

The class “AssemblyComponent” is associated with the class “AssemblyFeature” via the relation “hasAssemblyFeature”. The class “AssemblyFeature” has been associated with the classes “ShapeAttribute”, and “SpatialLocation” with the relations “hasShape” and “hasLocation” as shown in figure 4.26. These relations represent the shape and geographic locations of assembly feature and they have been further discussed in chapter 5.

The relation “matesWith” associate the class assembly feature with itself. This means that it is held between the mating assembly features. Similarly the relation “hasFitWith” define the fits relationship with the mating assembly features. For an assembly feature to be fit in another assembly feature, it requires to be with in the dimensional specification. The dimension and tolerance are provided to ensure that the mating features assemble together properly. This kind of knowledge is captured using the relations “hasDimension” and “hasTolerance” as shown in figure 4.26.

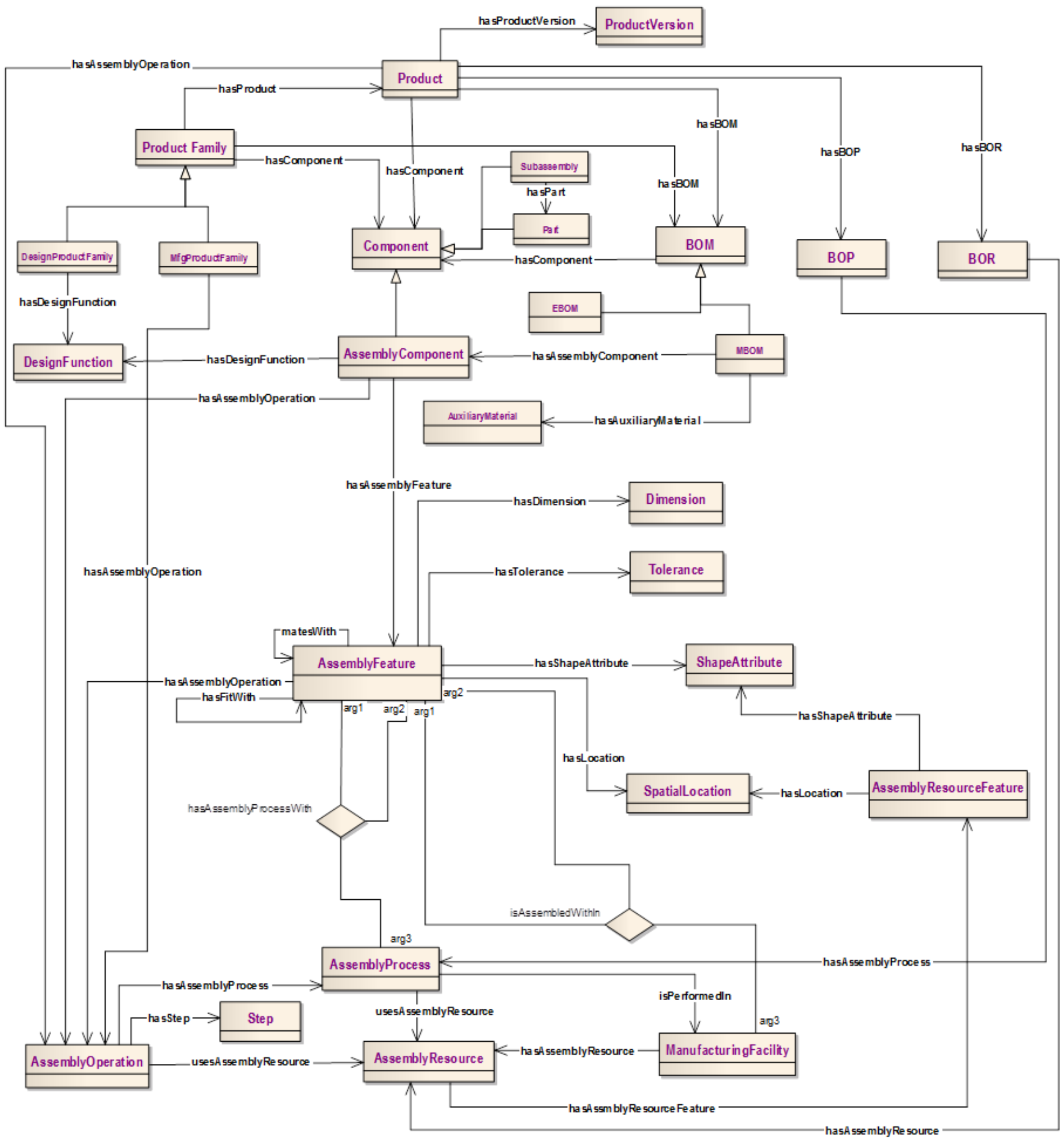


Figure 4.26: A UML based representation of interclass relationships in the ARO

Assembly operations and processes use assembly resources to accomplish the assembly of a product. Hence a relation “usesAssemblyResource” is defined between these classes. A manufacturing facility has assembly resources and this relation has been represented by “hasAssemblyResource” which holds between the manufacturing facility class and assembly resource class.

Ternary relations “hasAssemblyProcessWith” and “isAssembledWithIn” have been defined between the classes assembly feature, assembly process and with the assembly feature, manufacturing facility classes as shown in the figures. These relations have been further explored in chapter 5. The relation “hasDesignFunction” is defined to represent the link between the classes like design product family, assembly component and the design function class.

The relations represented in figure 4.26, show only an abstract view of the complex relationships involved in the product assembly domain. Therefore these relations and the associated classes need to be further explored. The relationships related to the concepts MBOM and assembly feature has been further explored in chapter 5. MBOM and assembly feature concepts are important from intra-domain and inter-domain assembly knowledge sharing point of view and hence present comprehensive examples of representation and sharing of assembly knowledge.

4.6 Formalization of ARO

This section briefly discusses the formalization process of the assembly reference ontology. As discussed in section 3.3, it is required to use a formal language for the representation and sharing of assembly knowledge. KFL which is a common logic based ontological approach is being explored to formalize the ARO. The formalization process consists of declaring the ARO concepts as properties, declaration of their mutual relationships, functions and the use of axioms to apply constraints and infer the new knowledge. The formalization of ARO is briefly explained as follows.

4.6.1 Declaring Properties

The term property refers to any taxonomic component when writing ontology in KFL. The reference concepts identified for the ARO has been declared as properties in the KFL. Properties can be declared in the KFL by using the following format.

```
:Prop AssemblyFeature

:Inst Type

:sup ProductFeature

:name "Assembly Feature class"

:rem "Assembly features are related with each other
from assembly design and assembly process planning
perspectives".
```

Each line in the above format is called a directive and it starts with a colon. The term prop in the first directive refers to the property. AssemblyFeature has been declared as a property which is a subsumption of the class ProductFeature. The second directive starts with "Inst" which stands for "instance of" and is used to declare the type of instantiation of properties.

The third directive starts from "sup" which actually describes the super-property of the concept declared in the first directive. In the above example, it can be said that AssemblyFeature has super property ProductFeature. The fourth and fifth directives are optional and represent additional information related to the declared property. For example, name of the properties is sometime abbreviated to simplify their representation. In such cases the full name may be written in the name directive. Similarly remarks may be added to describe any other information which might be considered necessary.

Furthermore the specialized classes of assembly features e.g. hole AF, shaft AF etc., and other ARO classes can be declared using the same format. A complete declaration of properties in the ARO can be found in appendix B.1.

4.6.2 Declaring Relations

The concept super-property only represents parent child relationships. However there are other relations between the classes or properties (see section 4.5 for more details) which need to be captured. These relations can be declared in KFL by using the following directives.

```
:Rel hasTolerance
:Inst BinaryRel
:Sig Object Tolerance
:Arg "Assembly Feature" "Tolerance"
:name "has tolerance".
```

Like properties, relations require first three fields from the above declaration e.g. "Rel", "Inst", and "Sig", however the last two fields "Arg", and "name" are optional. The first directive starts from "Rel" which is actually the name of relation e.g. "hasTolerance". The second directive starts from "Inst" that describes the type of relation held between the properties. The following instances of relations are acceptable in KFL.

UnaryRel: relates one property through this relation

BinaryRel: relates two properties in a relation

TernaryRel: relates three properties in a relation

QuaternaryRel: relates four properties in a relation

QuinaryRel: relates five properties in a relation

Relation: can relate any number of properties in a relation

The relation "hasTolerance" is an example of "BinaryRel" which relates two properties "Object" and "Tolerance" as can be seen in the second directive of the relation declaration. The third field in the relation declaration is "Arg" which stands for arguments. "Arg" directive describes the properties which are related with each other e.g. "AssemblyFeature", and "Tolerance". The last two directives are optional and may be added to facilitate the modeller and/or the user.

The tolerance related relations have been further explored in chapter 5 where they have been used to constrain and deduce assembly design related information.

4.6.3 Declaring Functions

Functions in KFL provide additional entities from different parameters. For example, all the measurements related information can be captured with the help of functions. The function is declared in KFL by using the following directives.

```
:Fun m

:Inst UnaryFun

:Sig RealNumber -> LengthDimension
```

The first directive starts with "Fun" which stands for function, defines the name of function to be declared. The second directive starts with "Inst" similar to property and relation declaration and refers to the type of function e.g. unary function, binary function etc. The third field "Sig" stands for signature and describes the arguments to the function of left hand side of the arrow shown above and the instantiated property on the right hand side of the arrow. In the above mentioned function “m” which needs only one argument to return it length dimension.

Functions have been further explored as necessary part of the formalization process and they have been used in chapter 5 and 6 to capture the assembly domain semantics. The next section discusses about the application of axioms.

4.6.4 Applying the Axioms

In addition to classes, relations and functions, formal ontologies consist of axioms which can support the representation and sharing of assembly knowledge. KFL axioms comprises of constraints and rules which are explained below.

4.6.4.1 Constraints

Constraints in KFL are represented by Integrity Constraints (ICs). The last directive of a constraint axiom starts with “:IC” followed by its type e.g. soft, hard, and a comment in inverted commas. The comment is shown to the user as a warning if any of the ICs is violated. An example of constraint to capture the semantics of MBOM concept is presented in the following axiom.

```
(=> (MBOM ?mbom)

       (exists (?aclist)

       (and (AssemblyComponentList ?aclist)

      (hasAssemblyComponentList ?mbom ?aclist))))

:IC hard "Every MBOM should have assembly component list."
```

All variables (shown in the above axiom) are represented by question mark (?) sign. The axiom starts with parentheses followed by an equal and greater than

(=>) sign. The constraint enforces that every MBOM should have assembly component list. It suggests that whenever there exists an instance of MBOM there should also exist an instance of assembly component list and the instance of MBOM should have that instance of assembly component list. These kinds of constraints can improve the model accuracy by preventing the wrong assertions in the database and by forcing the users to assert correct information. Thus constraints help to define the true semantics of concepts.

Various other constraints applied in the ARO have been explained in chapter 5. However in relation to the MBOM semantics and its specialized concepts, more constraints have been applied to demonstrate the understanding of concept specialization in section 5.2.3.

4.6.4.2 Rules

Rules are axioms which infer new knowledge from the existing knowledge. They are different from constraints as they add new knowledge instead of preventing any inconsistent statement. In KFL, rules look similar to the constraints except the last directive which starts with ":rem" instead of "IC". An example of a KFL rule which can be used to deduce new knowledge is explained below.

```
(<= (hasDimensionWithTolerance ?p ?q ?tol)

      (and (Object ?p)

             (Dimension ?q)

             (Tolerance ?tol)

             (hasDimension ?p ?q)))

:rem "The relation hasDimensionWithTolerance can be deduced from
the relations hasTolerance and hasDimension for the given
dimension and tolerance of an object."
```

The rule states that the relation hasDimensionWithTolerance can be deduced from the relations hasDimension and hasTolerance. This suggests that if clauses having hasDimension and hasTolerance relations are true then the statement having hasDimensionWithTolerance is also true for a given object ?p with dimension ?q and tolerance ?tol. Hence rules are can be used to deduce new information from the existing information. A further exploration of rules has been done in chapter 5.

4.7 Summary

A set of assembly reference ontology concepts and their mutual relationships have been identified and explored in this chapter. These reference concepts have been taken from different domains however most of these concepts have their links with the assembly domain. The ARO concepts have been related with the foundation concepts (provided by Highfleet) to create the formal assembly reference ontology.

The ARO is neither a purely design ontology nor it is only aimed at the assembly planning aspects, rather it provides intermediate concepts which have been associated by using the ARO relationships to support the representation and sharing across the assembly design and assembly process planning domains. The chapter described a brief but in-depth exploration of the ARO concepts and their relationships by providing the informal definitions of the key reference concepts used in this ontology. At the end, a combined representation of all the ARO concepts has been discussed to give an overall view of the ontology.

The informal definitions and intuitions of assembly reference concepts can be formally defined by using the formal language (knowledge frame language (KFL) in this research). An overview of the formalization process has been explored by using the properties, relations, functions, and axioms. The detailed exploration of a selected set of reference concepts, and how they can be exploited to represent and share assembly knowledge is discussed in the next chapter.

CHAPTER 5

EXPLORING MBOM AND ASSEMBLY FEATURE CONCEPTS FOR ASSEMBLY KNOWLEDGE SHARING

5.1 Introduction

This chapter discusses in detail the key ARO concepts: Manufacturing Bill of Materials (MBOM) and assembly feature for the representation and sharing of assembly knowledge. These concepts provide examples of intra-domain and inter-domain assembly knowledge sharing as they consider some domain specific assembly design and assembly process planning concepts. The MBOM (Assembly Process Planning Reference Concept) and assembly feature (Assembly Specific Reference Concept) have been chosen keeping in view their multiple interpretations and implications in assembly design and assembly process planning domains. Section 5.2 and its sub-sections explain the MBOM concept and three of its different interpretations. This section demonstrates how these multiple interpretations of the MBOM concept can be captured and represented using a formal ontological approach.

Section 5.3 explains the assembly feature concept and its implications for assembly design and assembly process planning domains. The assembly feature concept is associated with other ARO concepts such as dimension, tolerance, shape attribute, assembly process, assembly resource and manufacturing facility. Although most of the assembly feature related ARO concepts have been informally defined in chapter 4, this section discusses the complex relationships between these concepts and how these concepts can be formalized to support knowledge sharing across the assembly design and assembly process planning domains.

5.2 Exploration of MBOM Concepts

As discussed in section 4.3.6 of chapter 4, Engineering Bill of Materials (EBOM) and Manufacturing Bill of Materials (MBOM) are two sub-concepts of Bill of Materials (BOM) which represent the Assembly Design (AyD) and Assembly Process Planning (APP) domains respectively. However within the AyD and APP domains, EBOM and MBOM may also have different meanings and interpretations, and potentially hinder the knowledge sharing across the AyD and APP domains. The process of knowledge sharing within the AyD and APP domains is termed as intra-domain AyD knowledge sharing and intra-domain APP knowledge sharing. The case of APP intra-domain knowledge sharing has been investigated through the example of three different interpretations of MBOM.

5.2.1 Informal Description of MBOM Concepts

Different interpretations of MBOM have been found in the literature and they have been analysed to investigate the intra-domain knowledge sharing scenario. The first informal definition of MBOM is based on Stark (2011)'s interpretation of MBOM. Stark (2011) describes MBOM as a list of items in EBOM and other things needed to make a product e.g. machine oil etc. Stark (2011) also describes EBOM items as objects which make up the product from a design viewpoint. The additional bit from manufacturing or assembly point of view are the things which facilitate the making of product assembly and these may include, for example, machine oil for lubrication, paint and tape to mark the floor.

The second MBOM (informal) definition is based on the interpretation from Hirata (2009)'s work. Hirata (2009) describes that generally MBOM does not include items such as paint, tape and some small items like bolts, nuts whereas these small items are described As Required (AR) items on engineering drawings. Although Hirata (2009) concludes that these items should be part of

MBOM however his general description of MBOM excludes these items from MBOM. It is interesting to note that the second interpretation of MBOM is considerably different from that of first one. The interpretation of MBOM based on Stark (2011) has been termed as MBOMs and the interpretation from Hirata (2009)'s work as MBOMh.

The author of this thesis adds third interpretation of MBOM called MBOMi. The informal definition of MBOMi is based on author's understanding of the MBOM concepts gained from the existing literature. The MBOMi is described as a list of assembly components which are directly used in building a product assembly. For example, MBOMi contains all the assembly components of a valve assembly shown in figure 5.1 including the small components such as nuts and bolts. However the items used indirectly to make the product are not included in the MBOMi item list. The examples of such indirect items are oil, paint and tape.

Once the informal definitions of these MBOM concepts are in place, the next step is to represent these concepts in UML to facilitate the visualization of all the MBOM and associated concepts. Other related concepts used in the informal definitions of MBOM are the different types of lists of items or objects which are required to assemble the product. Two main types of lists of items have been identified. These are: (1) list of items used directly to build the product, and (2) list of items used indirectly to build the product. The first type of list is called an assembly component list and the second type of list is named as an auxiliary material list.

The Assembly Specific Reference Concept "assembly component" has already been described in section 4.3.11 of chapter 4. The Product Lifecycle Reference Concept "auxiliary material list" has auxiliary materials where the latter are the indirect materials used to build a product. For instance, machine oil, paint, and tape are not directly used in the assembly of a product hence they are termed as auxiliary materials in this research. The examples of assembly components and auxiliary materials are shown in figure 5.1.

The assembly component list has been specialized into two further concepts which are: “As Required (AR) assembly component list” and “As Designed (AD) assembly component list”. AR assembly component list represents small and standard components which are described as AR assembly components on the assembly drawing. It implies that they are not purchased through the Material Requirement Planning (MRP) process instead they are acquired as bulk. The AD assembly components are those assembly components which are not AR assembly components and are purchased through the MRP process. The examples of AR and AD assembly components are shown in figure 5.1.

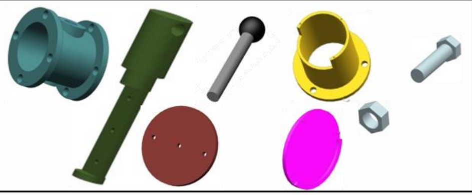
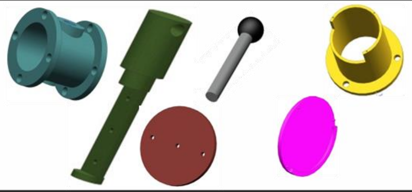
Concept Description	Concept Name	Concept Examples
Items directly used in assembly of product.	Assembly Component	
Items described as A/R (as required) on assembly drawing.	AR Assembly Component	
Items other than A/R items.	AD Assembly Component	
Items indirectly used in assembly of product.	Auxiliary Material	

Figure 5.1: Description of concepts used in the definitions of MBOM concepts

5.2.2 UML based Representation of MBOM Concepts

The AD and AR assembly component list concepts are Assembly Process Planning Reference Concepts and have been introduced to capture and differentiate the semantics of MBOMs, MBOMh, and MBOMi. It is now possible to represent these MBOM concepts with a set of related concepts described above. The more appropriate informal definitions of MBOMs, MBOMh, and MBOMi can be described in terms of a list of assembly components, AD assembly components, AR assembly components and auxiliary materials. In the first step these concepts would be used in UML based representation of MBOM concepts and consequently will be formalized using a heavyweight ontological approach.

Subsequently the informal definitions of three MBOM concepts can be put into words as follows:

a. **MBOMs:** is a list of assembly components which include the AD assembly components and AR assembly components, and a list of auxiliary materials.

b. **MBOMh:** is a list of assembly components (which are AD assembly components) excluding the AR assembly component list and excluding the auxiliary materials list.

c. **MBOMi:** is a list of assembly components which include AD assembly components and AR assembly components, but MBOMi excludes the list of auxiliary materials.

The UML based representation of MBOMs, MBOMh, and MBOMi is shown in figure 5.2. Figure 5.2a shows that MBOMs has both the assembly component list and auxiliary materials list. The constraint attached with MBOMs reflects that whenever MBOMh exists, it should always have auxiliary material list. Figure 5.2b shows the UML based representation of MBOMh. Two different constraints have been attached with MBOMh which dictate that it should not have AR assembly component list and auxiliary material list. Figure 5.2c shows the representation of MBOMi where one constraint has been applied to show that MBOMi should not have the auxiliary material list as part of its definition.

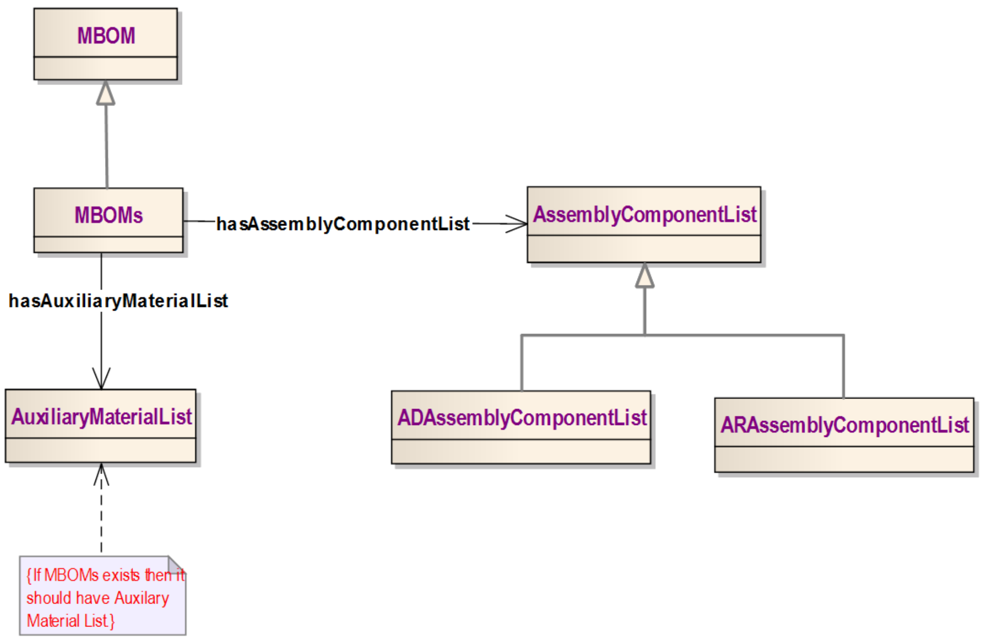


Figure 5.2a: UML based lightweight representation of MBOMs

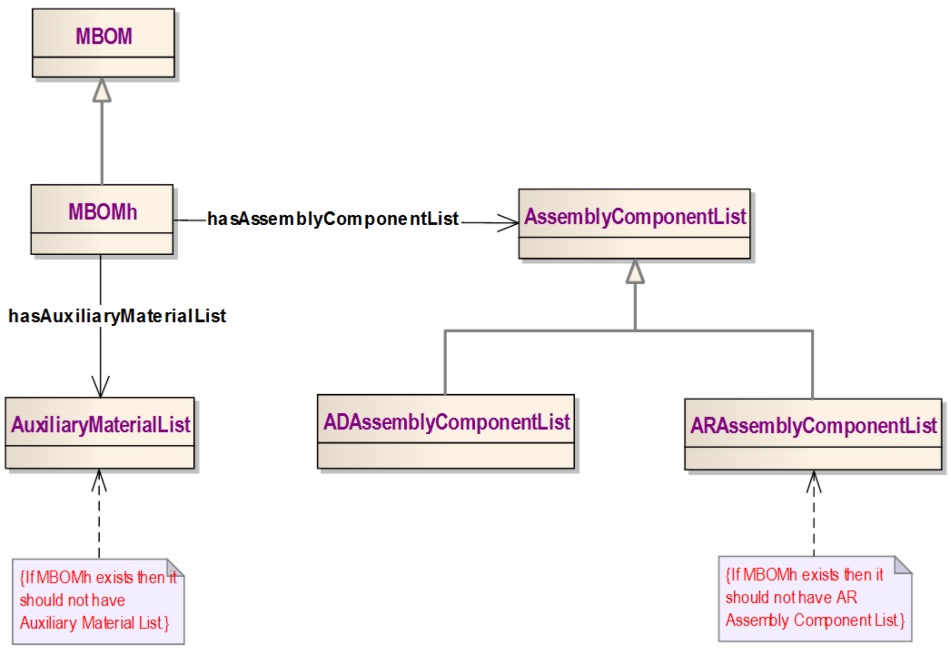


Figure 5.2b: UML based lightweight representation of MBOMh

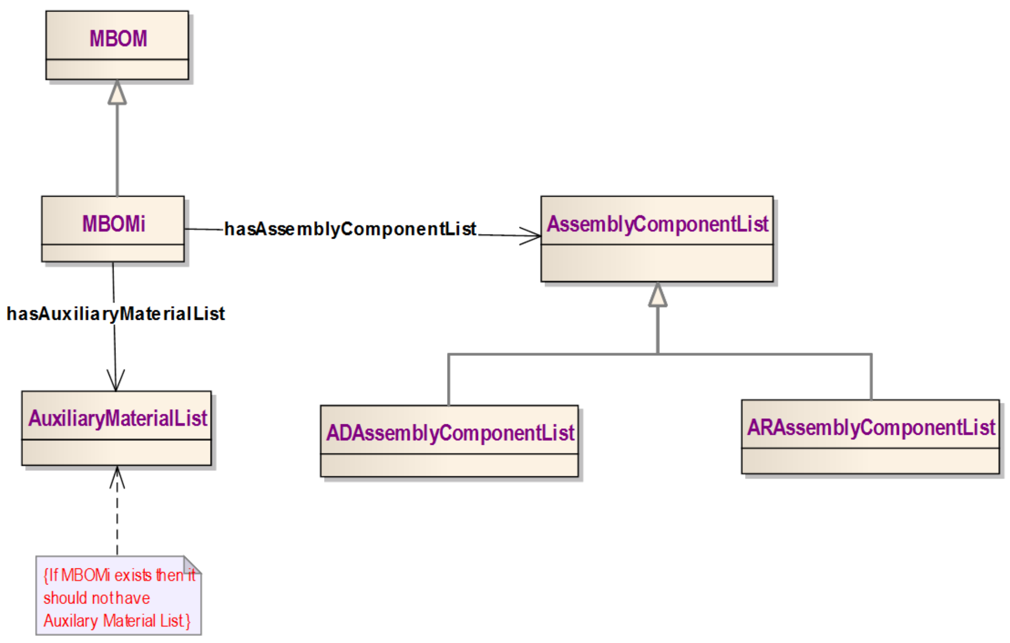


Figure 5.2c: UML based lightweight representation of MBOMi

Figure 5.2 : UML based lightweight representation of three MBOM concepts.

5.2.3 Formalization of MBOM Concepts

The formalization process has been briefly described in section 4.6, where the semantics of the MBOM concept were constrained by the application of an axiom (see section 4.6.4.1). In this section, the definitions of specialized classes of MBOM e.g. MBOMs, MBOMh, and MBOMi are formally captured. The formalization process starts with the declaration of properties. For example, MBOMs property has been declared in KFL as follows:

```
:Prop MBOMs

:Inst Type

:sup MBOM

:name "Manufacturing Bill of Materials based on
Stark, (2011)'s definition"

:rem "MBOMs has assembly component list as well as
auxiliary materials list."
```

The name and remark directives are optional and can be added to facilitate the modeller. Similarly two other MBOM properties can be declared using the same KFL format. Now the properties: assembly component list and auxiliary materials list represent the list of assembly components and auxiliary materials. They have been specialized under the ULO property “list” where the latter describes a series of particulars. For instance, the declaration of auxiliary material list in KFL is as follows:

```
:Prop AuxiliaryMaterialList

:Inst Type

:sup List
```

Similarly the other list related properties: assembly component list, AD assembly component list, and AR assembly component list have been declared in KFL. Other concepts such as auxiliary material and assembly component are also captured and can be found in appendix B.1.

Most of the relations associated with MBOM are instances of binary relation. For example, MBOMs has the “hasAuxiliaryMaterialList” relation with auxiliary material list which is actually a binary relation. The relation can be declared as follows:

```
:Rel hasAuxiliaryMaterialList

:Inst BinaryRel

:Sig BOM AuxiliaryMaterialList
```

However the relation between all the instances of property “list” and instances of properties like assembly component and auxiliary material has variable arity. For example, an assembly component list may have any number of assembly components and the assembly components may vary depending upon the product for which the MBOM has been created. The only variable arity relation in KFL is the relation “item” which would now be used instead of relations like “hasAssemblyComponent”. Due to this limitation, the relations “hasAssemblyComponent”, and “hasAuxiliaryMaterial” have been replaced with the ULO relation “item”.

Axioms have been applied to constrain and capture the semantics of concepts already declared as properties. For example, the following two axioms have been applied to capture and constrain the semantics of the concept “assembly component list”:

```
(=> (AssemblyComponentList ?l)
      (exists (?c)
            (and (AssemblyComponent ?c)
                   (item ?l ?c))))
:IC hard "Every assembly component list consists of at least one assembly component within the list."


(=> (AssemblyComponentList ?l)
       (not (exists (?other)
                (and (item ?l ?other)
                 (not (AssemblyComponent ?other))))))
:IC hard "Every assembly component list should consist of exclusively assembly components that make up the list."
```

The first axiom dictates that an instance of assembly component list should have at least one instance of assembly component. It implies that the system would not accept any list which does not have at least one assembly component. However there are still possible chances that the system accepts a list which has one or more instances of assembly component as well as instances of other concepts like auxiliary material. The second axiom averts such attempts and makes the system accept instances of assembly components only for the property “assembly component list”.

The concepts “AD assembly component list” and “AR assembly component list” are mutually exclusive meaning that no single instance of AD assembly component can be found in AR assembly component list and no single instance of AR assembly component list should be found in AD assembly component list. This can be captured by applying the following axioms.

```
(=> (ADAssemblyComponentList ?l)

       (not (exists (?x)

            (and (ARAssemblyComponent ?x)

                               (item ?l ?x)))))

:IC hard "Every AD assembly component list should not
consist of AR assembly components."


(=> (ARAssemblyComponentList ?l)

       (not (exists (?x)

            (and (ADAssemblyComponent ?x)

                               (item ?l ?x)))))

:IC hard "Every AR assembly component list should not
consist of AD assembly components."
```

The first axiom (mentioned above) says that AD assembly component list cannot have any AR assembly component whereas the second axiom dictates that AR assembly component list cannot have AD assembly component list. Similarly semantics related to auxiliary material list has been captured using these kinds of axioms.

So far, constraints have been applied to assembly component lists and auxiliary material list. The constraints shown in the UML diagrams in figure 5.2 can also be represented in KFL by using axioms. A summary of such axioms is shown in figure 5.3.

Concept	Axioms applied to constrain the semantics of MBOM concepts	Examples

MBOMs	`(=> (MBOMs ?mboms) (exists (?amlist) (and (AuxiliaryMaterialList ?amlist) (hasAuxiliaryMaterialList ?mboms ?amlist))))`
	`:IC hard` **`"Every MBOMs should have auxiliary material list."`**	

MBOMh	`(=> (and (MBOMh ?mbh) (AuxiliaryMaterialList ?aml)) (not (hasAuxiliaryMaterialList ?mbh ?aml)))`
	`:IC hard` **`"MBOMh should not have AuxiliaryMaterialList."`**	

MBOMh	`(=> (and (MBOMh ?mbh) (ARAssemblyComponentList ?aacl)) (not (hasARAssemblyComponentList ?mbh ?aacl)))`
	`:IC hard` **`"MBOMs should not have ARAssemblyComponentList."`**	

MBOMi	`(=> (and (MBOMi ?mbi) (AuxiliaryMaterialList ?aml)) (not (hasAuxiliaryMaterialList ?mbi ?aml)))`
	`:IC hard` **`"MBOMi should not have AuxiliaryMaterialList."`**	

Figure 5.3: Example of four different axioms applied to constrain the meanings of MBOMs, MBOMh and MBOMi.

The first axiom attaches a constraint on MBOMs that it should always have auxiliary material list. This suggests that whenever a user will try to populate an instance of MBOMs, the system will demand an assertion of auxiliary material list as well. It is important to note that a constraint was applied on MBOM (parent class of MBOMs) which dictated that whenever MBOM exists, it should have an assembly component list (see section 4.6.4.1). This implies that the same constraint should also be applicable on MBOMs and other specialized classes of MBOM. Specifically with reference to MBOMs, it should have an assembly component list and an auxiliary material list.

The second axiom dictates that the MBOMh concept should not have an auxiliary material list. This axiom would not allow the users to assert instances of auxiliary materials in the MBOM knowledge base. In the same fashion, the third axiom averts any attempt to assert instances of AR assembly components. This complies with the definition of MBOMh that it should not have auxiliary materials and AR components as discussed in the previous sections.

The fourth axiom applies a constraint on MBOMi that it should not accept any instance of auxiliary material. This suggests that MBOMi consists of AD and/or AR assembly component lists.

Once the semantics of all the MBOM concepts are formally defined, they can be experimentally evaluated by instantiation. The experimental investigation of these concepts is described in detail in chapter 6.

5.3 Exploration of Assembly Feature and Related Concepts

5.3.1 Introduction

This section explores the Assembly Specific Reference Concept assembly feature for assembly knowledge representation and sharing. An example of a journal bearing assembly has been considered to explain and demonstrate the capture and sharing of assembly knowledge. This section comprises of three main parts: (a) an informal description of assembly feature, and related concepts and relationships, (b) a UML based lightweight representation of these concepts and relationships, and (c) a formalization of these concepts and relationships. Although the ARO concepts have been defined in chapter 4, however this section further elaborate some of those concepts for the assembly knowledge sharing scenario.

5.3.2 Informal Description of Concepts and Relationships

5.3.2.1 Assembly Feature

An assembly feature is a key ARO concept which carries vital assembly information and hence is considered important from an assembly knowledge sharing point of view. As discussed in section 4.3.12, an assembly feature carries assembly design as well as assembly process planning related information hence it has been explored in detail to demonstrate the knowledge representation and sharing aspects across the assembly domain.

In the context of this work, the possible design implication of the assembly feature is the tolerancing and fits related information and from the APP perspective assembly feature can be linked to assembly processes, assembly resources and manufacturing facility. Hence it can be said that an assembly feature of a particular dimension, has a relationship in the form of assembly fits with another assembly feature and these assembly features have a specific assembly process through which they can be joined together. The term assembly feature has been used instead of the terms component because the component could have more than one joining contact with each other and this can be problematic when capturing knowledge or even retrieving knowledge or a query for a specific assembly joint. A typical example is shown in figure 5.4 where a component “journal bearing bush” has two assembly features A and B.

This bush would be joined with a bearing housing using the assembly feature A and with a shaft using the assembly feature B.

Although assembly feature has been specialized into various classes (see section 4.3.12), in relevance to this scenario hole Assembly Feature (AF) and shaft Assembly Feature (AF) have been considered.

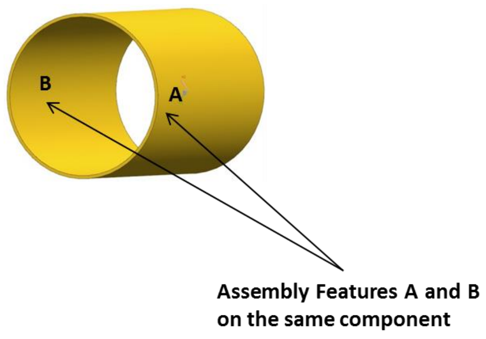


Figure 5.4: Multiple assembly features on the journal bearing bush component

5.3.2.2 Tolerance Standard BS 4500

Tolerance standards enable designers to specify tolerance quantities which depend upon the dimensions of mating assembly features. The limits and fits standard BS 4500 (1969) classifies tolerance types such as H8, H7, f7, k6, p6 etc. and subsequently suggests the type of assembly fits. The tolerance types for the hole AF starts with the capital letter while tolerance types for the shaft AF starts with the small letter.

For example, H8 and H7 are tolerance quantities which are applicable to holes only, while f7, k6 and p6 are tolerance quantities applicable to shafts only. These standard tolerance categories are represented with the concept “Tolerance Type” in this research. The Assembly Design Reference Concept “Tolerance Type” support the capture of design knowledge related to tolerance standard BS 4500 and has been frequently used in the upcoming sections. A selected set of tolerance quantities for hole and shaft assembly features is shown in table 5-1.

Table 5-1: BS 4500 hole and shaft tolerances (BS 4500, 1969)

		Clearance Fit		**Transition Fit**		**Interference Fit**	
Basic Dimension		**Tolerance**		**Tolerance**		**Tolerance**	
Over	**To**	**H8**	**f7**	**H7**	**k6**	**H7**	**p6**
mm	**mm**	**Multiple of 0.001mm**	**Multiple of 0.001mm**	**Multiple of 0.001mm**	**Multiple of 0.001mm**	**Multiple of 0.001mm**	**Multiple of 0.001mm**
-	**3**	+14 0	-6 -16	+10 0	+6 0	+10 0	+12 +6
3	**6**	+18 0	-10 -22	+12 0	+9 +1	+12 0	+20 +12
6	**10**	+22 0	-13 -28	+15 0	+10 +1	+15 0	+24 +15
10	**18**	+27 0	-16 -34	+18 0	+12 +1	+18 0	+29 +18
18	**30**	+33 0	-20 -41	+21 0	+15 +2	+21 0	+35 +22
30	**50**	+39 0	-25 -50	+25 0	+18 +2	+25 0	+42 +26
50	**80**	+46 0	-30 -60	+30 0	+21 +2	+30 0	+51 +32

80	**120**	+54	-36	+35	+25	+35	+59
		0	-71	0	+3	0	+37
120	**180**	+63	-43	+40	+28	+40	+68
		0	-83	0	+3	0	+43
180	**250**	+72	-50	+46	+33	+46	+79
		0	-96	0	+4	0	+50
250	**315**	+81	-56	+52	+36	+52	+88
		0	-108	0	+4	0	+56
315	**400**	+89	-62	+57	+40	+57	+98
		0	-119	0	+4	0	+62
400	**500**	+97	-68	+63	+45	+63	+108
		0	-131	0	+4	0	+68

To demonstrate the use of the tolerance table, consider hole and shaft AF having a dimension of 52 mm. Let us suppose that the tolerance specification for the hole and shaft is H7/p6. It is necessary to determine the maximum and minimum allowable dimensions of the hole and shaft AF.

Now to determine the maximum and minimum allowable dimensions, we need to find out the upper and lower quantities of respective tolerances of the hole and shaft AF. First consider the case of hole AF.

HoleAF dimension = 52 mm

HoleAF tolerance type = H7

Upper tolerance quantity for 52 mm hole dimension = (30*0.001) mm = 0.030 mm

(Note: See 52 mm dimension against H7 tolerance in table 5-1)

Lower tolerance quantity for 52 mm hole dimension = (0*0.001) mm = 0 mm

Maximum allowable dimension for hole AF = hole dimension + upper tolerance quantity = 52 mm + 0.030 mm = **52.030 mm**

Minimum allowable dimension for hole = hole dimension + lower tolerance quantity = 52 mm + 0 mm = **52 mm**

Similarly the maximum and minimum allowable dimensions for shaft AF can be calculated as follows:

Shaft dimension = 52 mm

Shaft tolerance type = p6

Upper tolerance quantity for 52 mm shaft dimension = (51*0.001) mm = 0.051 mm

(Note: See 52 mm dimension against p6 tolerance in table 5-1)

Lower tolerance quantity for 52 mm shaft dimension = (32*0.001) mm = 0.032 mm

Maximum allowable dimension for shaft = shaft dimension + upper tolerance quantity = 52 mm + 0.051 mm = **52.051 mm**

Minimum allowable dimension for shaft = shaft dimension + lower tolerance quantity = 52 mm + 0.032 mm = **52.032 mm**

The minimum and maximum allowable dimensions are determined to allow a range of assembly features to be manufactured with varying dimensions. These assembly features are then assembled together with different types of fits where the latter are dependent upon minimum and maximum allowable dimensions. So the types of tolerance determine the types of fits and this is discussed in the next section.

5.3.2.3 Assembly Fits

A fit is the dimensional relationship between the mating components (Dictionary of Engineering, 2003). There are three types of fits commonly found in the literature which are: (a) clearance fit, (b) transition fit, and (c) interference fit. These types of fits are decided on the basis of functional requirements of mating components. The types of fits as found in BS 4500 (1969) and other literature sources are specified as follows:

(a) Clearance Fit: A hole and shaft has clearance fit with each other if the minimum allowable dimension of a hole is larger than the maximum allowable dimension of a shaft.

(b) Transition Fit: A hole and shaft has transition fit with each other if the minimum allowable dimension of a hole is smaller than the maximum allowable dimension of a shaft, and the maximum allowable dimension of a hole is larger than the minimum allowable dimension of a shaft.

(c) Interference Fit: A hole and shaft has interference fit with each other if the maximum allowable dimension of a hole is smaller than the minimum allowable dimension of a shaft.

Figure 5.5 shows that the “bearing housing” has interference fit with “bearing bush” (assembly feature A) and “bearing bush” (assembly feature B) has clearance fit with the “shaft”. This suggests that in all cases the maximum allowable dimension of bearing housing should always be smaller than the minimum allowable dimension of the bearing bush (assembly feature A) in order to have the interference fit. Similarly the minimum allowable dimension of bearing bush (assembly feature B) should always be larger than the maximum allowable dimension of the shaft in order to have clearance fit.

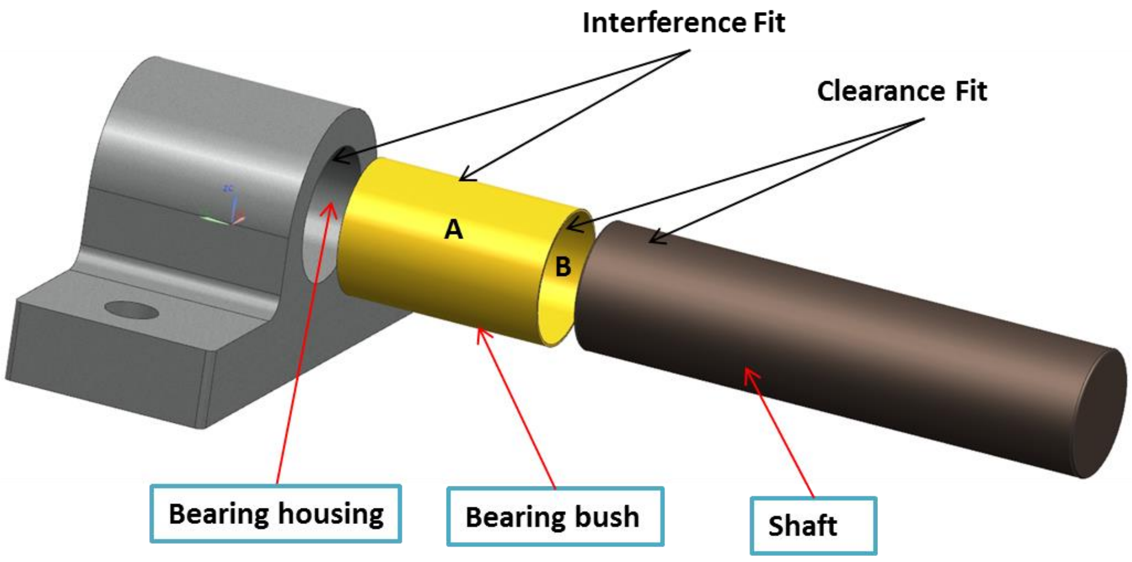


Figure 5.5: Types of fits in journal bearing assembly

In addition to the types of fits, the term "allowance" is also used when specifying the fits. An allowance is the minimum clearance or maximum interference between the hole and the shaft (Jensen, et al., 2001). The minimum clearance is the least difference between the minimum allowable dimension of the hole and the maximum allowable dimension of the shaft. Whereas the maximum interference is the difference between minimum allowable dimension of the hole and maximum allowable dimension of the shaft. The minimum clearance is always positive while the maximum interference is always negative.

5.3.3 UML Based Representation of Concepts and Relationships

This section explains the visual representation of assembly feature and related concepts and relationships in UML. In this work the UML diagram represents classes and their mutual relationships. As the focus of this research is to investigate the knowledge sharing aspects across the AyD and APP domains hence the UML based representation has been illustrated from design and planning perspective separately. The following sections describe them in detail.

5.3.3.1 Representation of Assembly Design Perspective

The assembly design perspective of hole and shaft assembly as explained in section 3.3 and section 5.3.2.1 is related to the tolerance, and assembly fits. The UML based representation of these concepts is explained in the following sections.

5.3.3.1.1 UML Based Representation of Tolerance

Figure 5.6 shows assembly feature concept and its link to the tolerance related concepts. These concepts have been connected with each other via different association relationships as shown in the figure. The concept “AssemblyFeature” has been associated with the concept “Tolerance” through a binary relation “hasTolerance” and a ternary relation “has DimensionWithTolerance”. The Design Reference Concept “Tolerance” itself is specified by two measurement quantities and these quantities have been captured using the function “tolerance” (not shown in figure 5.6) in KFL based modelling and would be discussed in section 5.3.4 as the functions are not displayed in UML class diagrams.

The ternary relation “hasDimensionWithTolerance” is a combination of the binary relations “hasDimension” and “hasTolerance”. The relation “hasDimension” represents the basic dimension of assembly feature and links the latter with the ARO concept “Dimension”. Hence the relation “hasDimensionWithTolerance” represents the basic size of the assembly feature along with the tolerance. The purpose of this relation is to help a user if he/she is interested to get basic dimension along with tolerance. The first argument in this relation is assembly feature followed by the second argument dimension and the tolerance.

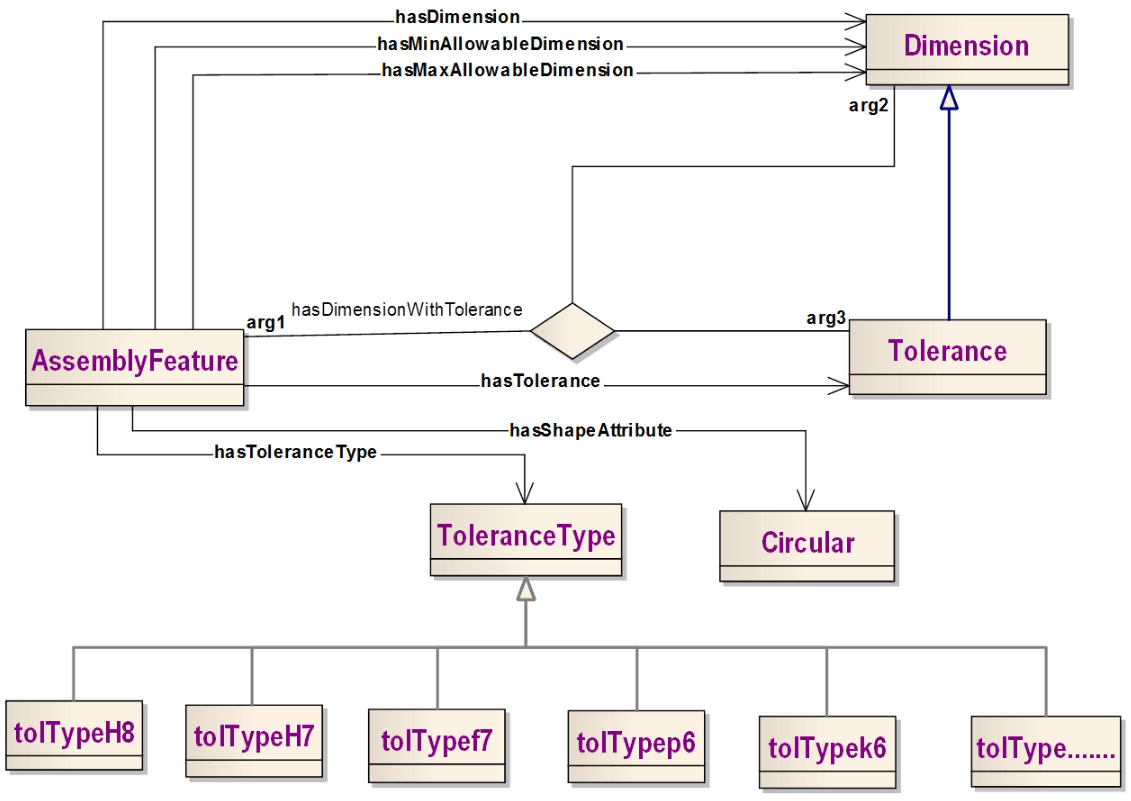


Figure 5.6: UML based representation of design perspective of assembly feature

To represent the limits e.g. the maximum and minimum allowable dimensions of assembly features, two binary relations “hasMinAllowableDimension” and “hasMaxAllowableDimension” have been used. These relations link the concept “AssemblyFeature” with the ARO concept “Dimension” as shown in figure 5.6.

This work also considers the representation of tolerance standard BS 4500 and the class “ToleranceType” has been used to represent the different grades of tolerances in BS 4500. A selected set of tolerance types (H8, H7, f7, p6, k6) has been considered and it is argued that other tolerance types can be similarly represented. The “ToleranceType” class provides users an option to specify the intended tolerance grade or type which would subsequently lead towards the recommended fits type and their production consequences. The instances of standard tolerance types are applicable to circular shapes of hole and shaft assembly feature therefore a specialized class of shape attribute has been associated with assembly feature as shown in figure 5.6.

5.3.3.1.2 UML Based Representation of Assembly Fits

A second aspect of the design perspective of assembly features are the assembly fits which are the dimensional relationships among the mating assembly features e.g. hole AF and shaft AF. As assembly fits are the dimensional relationships between assembly features, they have been represented by relationships instead of making separate classes for them. Three binary relations: hasClearanceFitWith, hasTransitionFitWith, and hasInterferenceFitWith have been specified between the classes: “HoleAF” and ShaftAF as shown in figure 5.7.

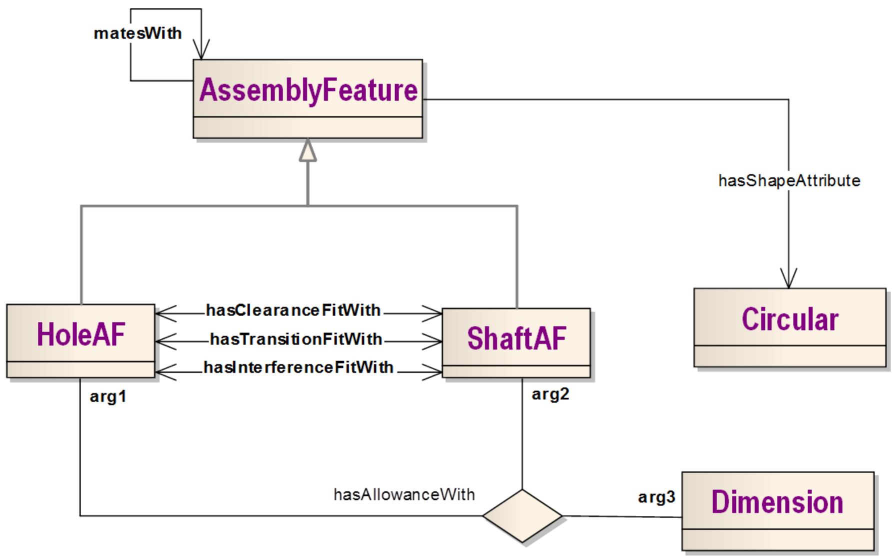


Figure 5.7: UML based representation of assembly fits

The binary relations between hole AF and shaft AF are bidirectional which implies that these relations can be applicable on both sides. For example both of the following claims would be true for “hasClearanceFitWith” relation:

HoleAF hasClearanceFitWith ShaftAF.

OR

ShaftAF hasClearanceFitWith HoleAF.

A ternary relation "hasAllowanceWith" has also been specified to represent the allowance between the hole AF and the shaft AF. The relationship is ternary because it relates the hole AF, the shaft AF and the dimension. The dimension in this case represents the minimum clearance or maximum interference (allowance) in terms of some measurement units.

Another binary relationship matesWith has been identified which links assembly feature with another assembly feature or hole AF with the shaft AF and vice versa. This relation specifies a special condition for the assembly fits relationships and the latter will only be held between the mating assembly features. For example the relations hasClearanceFitWith, hasTransitionFitWith, and hasInterferenceFitWith are only valid if the hole AF mates with shaft AF and vice versa.

Assembly feature class has also been related to the class circular through a binary relation hasShapeAttribute as shown in the figure. This association asserts that the fits relationships are held between hole and shaft AF only if they have circular shape attributes.

5.3.3.1.3 Combined UML Based Representation of Assembly Design Perspective

A combined representation of concepts and relationships related to design perspective of assembly feature is shown in figure 5.8. All of the design related concepts e.g. dimension, tolerance and tolerance type are shown in the diagram along with their relationships. However it is important to recognize that the axioms which actually constrain and infer the semantics of these concepts and relationships cannot be completely represented in UML based diagrams.

For instance, it is general practice that both of the tolerance quantities are represented in the same units of measurement. For instance if the lower quantity of tolerance is expressed in millimetres (mm) then the upper quantity should also be specified in mm. However these kinds of constraints may not be captured using the lightweight representation and therefore requires formal ontological approach. Section 5.3.4 discusses the formalization of these concepts and relationships in detail.

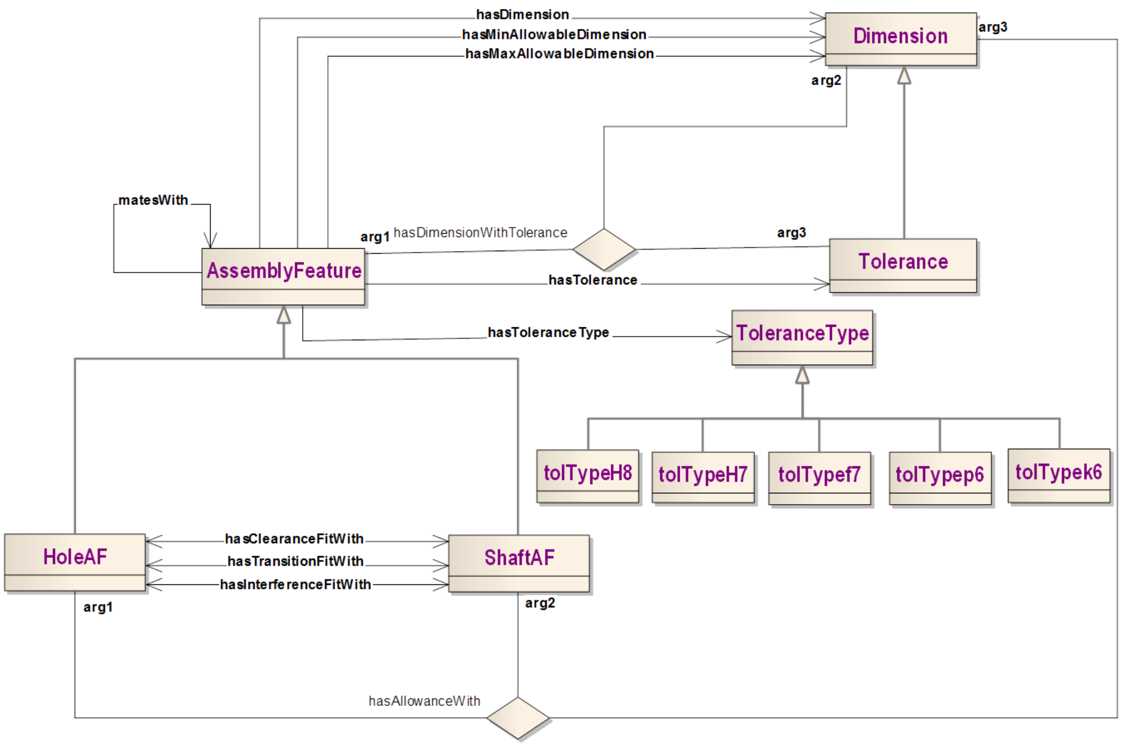


Figure 5.8: Combined representation of design perspective of assembly feature

5.3.3.2 Representation of Assembly Process Planning Perspective

The APP perspective of mating assembly features include the concepts which support the capture of APP knowledge and are specified in the form of manufacturing facility (Manufacturing Reference Concept) , assembly process

(APP Reference Concept) and assembly resource (APP Reference Concept) classes as shown in figure 5.9. The classes shown in figure 5.9 are linked with each other by one generalization relation, two ternary relations and three binary relations. The hole AF and shaft AF classes are linked with assembly feature class through generalization relation suggesting that hole AF and shaft AF are both specialized classes of assembly feature class.

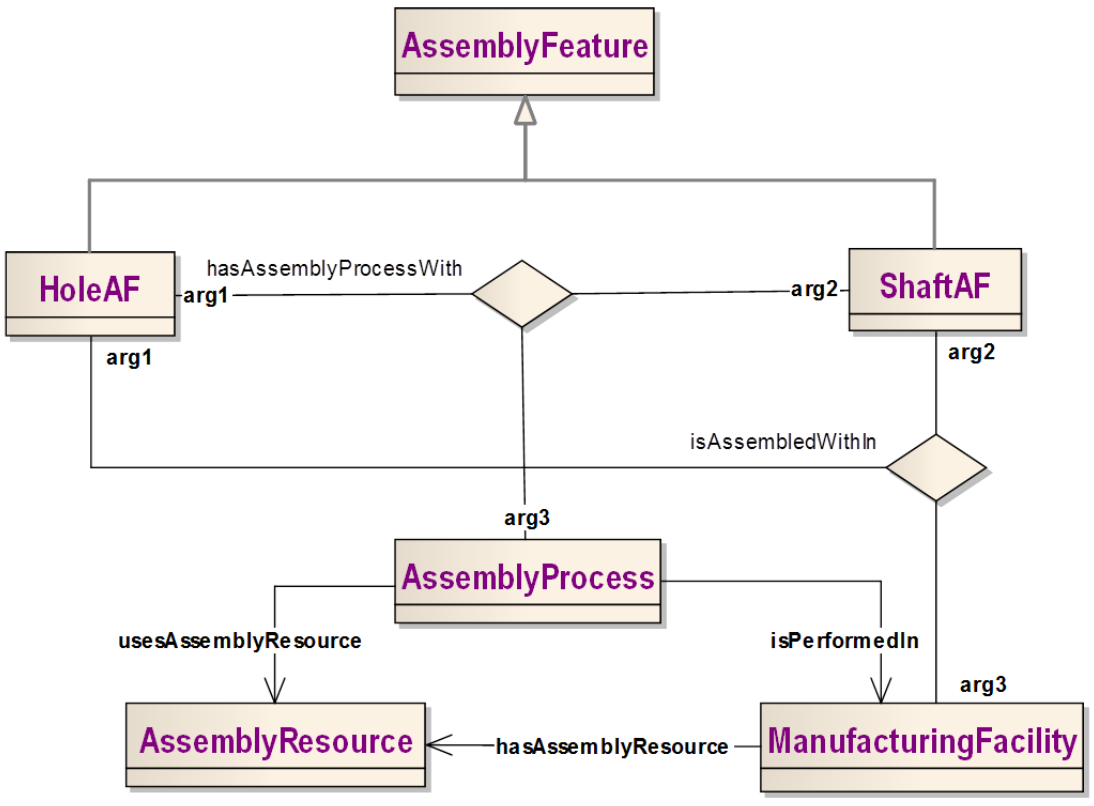


Figure 5.9: Assembly process planning perspective of assembly feature

The ternary relation “hasAssemblyProcessWith” associates hole AF, shaft AF, and the assembly process classes. This relation is useful to infer the possible assembly processes for a particularly type of assembly fit. The second ternary relation “isAssembledWithIn” links hole AF, shaft AF, and the manufacturing facility classes and can be used to evaluate whether a pair of hole and shaft assembly features can be assembled in a particular manufacturing facility or not.

Three binary relations shown in figure 5.9 are: “isPerformedIn”, “usesAssemblyResource”, and “hasAssemblyResource”. Assembly process class is linked with manufacturing facility class and assembly resource class via the relations “isPerformedIn” and “usesAssemblyResource” respectively.

The “isPerformedIn” relationship can be used to evaluate the manufacturing facility for the occurrence of a particular assembly process. The “usesAssemblyResource” relation helps to sort out a set of assembly resources for a particular assembly process. Finally the “hasAssemblyResource” relation is held between manufacturing facility and assembly resource classes and is used to find out the assembly resources for a particular manufacturing facility.

5.3.4 Formalization of Concepts and Relationships

The UML based class models provide only basic information about the classes and the relationships amongst them. However they are unable to constrain the semantics of concepts and are incapable of deducing new knowledge. The KFL based formalization compensates these deficiencies by applying Integrity Constraints (ICs) and inference rules. The following sections discuss in details the formalization of the ARO concepts for knowledge capture and knowledge sharing.

5.3.4.1 Formalization of Assembly Design Related Concepts

This section is divided into following three sub-sections:

- Formalization of Tolerance
- Formalization of Assembly Fits
- Formalization of Assembly Allowance

5.3.4.1.1 Formalization of Tolerance

As described in section 4.3.10, a tolerance is represented by two quantities which are lower quantity and upper quantity. The class “Tolerance” is subsumed under the ARO class “Dimension” as discussed in section 5.3.3.1.1. To capture the semantics of tolerance, it has been declared as a property (class) called “Tolerance” and as a function named as “tolerance”. The declaration of tolerance as property and as function are expressed in KFL as follows:

```
:Prop Tolerance

:Inst Type

:sup Dimension



:Fun tolerance

:Inst BinaryFun

:Sig Dimension Dimension -> Tolerance
```

The above expression states that the function tolerance takes two quantities to return the Tolerance where the latter has already been declared as a property. The function “tolerance” is applicable to user defined tolerance, however as this work also considers the BS 4500 (1969) limits and fits standard therefore various tolerance types have also been declared as functions separately. For instance the H8 function has been declared in KFL as follows:

```
:Fun H8

:Inst BinaryFun

:Sig Dimension Dimension -> Tolerance
```

In addition to these tolerance functions, a property “ToleranceType” has been specified to represent the type of tolerance e.g. H8, f7 etc. The “ToleranceType” property has been specialized into tolTypeH8, tolTypeH7, tolTypef7, tolTypek6, tolTypep6 classes which specify various types of tolerance. Similarly a relation “hasToleranceType” has been defined which links the assembly features with the type of tolerance. These tolerance type properties and relations help to retrieve the intended tolerance quantities from the knowledge base.

To measure the user defined and standard tolerance quantities, different measure functions have been defined which are: metre (m), centimetre (cm), millimetre (mm), and micron. These measure functions have been declared as unary functions in KFL to represent the dimension of assembly features. For instance, mm measure function is declared in KFL as follows:

```
:Fun mm

:Inst UnaryFun

:Inst MeasureFun

:Sig RealNumber -> LengthDimension
```

Similarly the other measure functions micron, cm, and m have been defined in KFL. To make these measure functions convertible with each other, following assertions have been added.

```
(measureMultiple m 100 cm)

(measureMultiple cm 10 mm)

(measureMultiple mm 1000 micron)
```

The functions defined so far allow the measurement parameters (lower and upper quantities of tolerance) to be used in reasoning. And various constraints have been applied to constrain the semantics of tolerance. For instance, as general practice, the first quantity in the tolerance function should always be less than the second quantity. This constraint can be applied in the following manner.

```
(=> (hasTolerance ?x (tolerance ?q1 ?q2))

      (and (Object ?x)

            (Dimension ?q1)

            (Dimension ?q2)

            (measureLT ?q1 ?q2)))

:IC hard "The lower deviation quantity of a Tolerance must
always be less than its upper deviation quantity."
```

The above axiom states that variable ?q1 which is a lower quantity should always be less than the variable ?q2 which is upper quantity, for any object ?x. This axiom will prevent any the assertion of any instance of ?q1 which is larger than an instance of ?q2.

Similarly as a general practice, the upper and lower quantities of tolerance should always be specified in same units of measurement. This constraint can be captured using the following axiom.

```
(=> (and (hasTolerance ?x (tolerance (?mfunc1 ?num1) (?mfunc2 ?num2)))

                   (Object ?x)

                   (returnProp ?mfunc1 ?q1)

                   (returnProp ?mfunc2 ?q2))

                   (= ?mfunc1 ?mfunc2))

:IC hard "Only quantities of the same kind are allowed to participate
in the tolerance function."
```

The above axiom dictates that only quantities of same kind of measurement function should be allowed in specifying the tolerance function. This would prevent any attempt to specify tolerance quantities with different measure functions.

Once the semantics of tolerance are constrained, the inference rules can be applied to deduce new knowledge from the existing knowledge. For instance, the minimum and maximum allowable dimensions of mating features can be determined by adding the lower and upper quantities of tolerance as discussed in section 4.3.10 and section 5.3.2.2. For instance, the inference rule for determining the minimum allowable dimension is as follows:

```
(<= (hasMinAllowableDimension ?p ?q+lower)
      (and (Object ?p)
             (hasDimensionWithTolerance ?p ?q (?tol ?lower ?upper))
             (measurePlus ?q ?lower ?q+lower)
             (or (= ?tol tolerance)
                   (= ?tol f7)
                   (= ?tol H8)
                   (= ?tol k6)
                   (= ?tol H7)
                   (= ?tol p6))))
:rem "The Object ?p has a minimum allowable dimension which is
equivalent to its nominal dimension plus its lower deviation."
```

The above axiom suggests that the minimum allowable dimension for an object ?p can be obtained by adding the lower tolerance quantity to its basic dimension. The tolerance can be user defined tolerance or one of the standard tolerance mentioned in table 5-1. Similarly the inference rule has been applied to deduce the maximum allowable dimension.

Now sometimes it may be useful to find a range of dimensions on the basis of a given tolerance. To determine a value or a range of values of dimensions acceptable for a given tolerance following axiom has been applied to infer the allowable dimensional value or range in mm measure function.

```
(<= (hasAllowableDimensionalValueOrRange ?p (mm ?valueOrInterval))
      (and (Object ?p)
      (hasMinAllowableDimension ?p (mm ?num1))
      (hasMaxAllowableDimension ?p (mm ?num2))
      (inInterval ?valueOrInterval (interval in ?num1 ?num2 in))))
:rem "The Object ?p has an allowable dimension or an allowable range
of dimensions in millimetres dictated by its minimum and maximum
allowable dimensions."
```

The above axiom dictates that the acceptable dimensional value or range of dimensional values of any object e.g. hole AF or shaft AF should be in between the minimum and the maximum allowable dimensions. Similarly, rules have been applied for other measure functions e.g. cm, m and micron.

Now to deduce the values of lower and upper quantities for standard tolerance functions: H8, H7, f7, p6, and k6, it is required to specify the shape attribute, basic dimension and tolerance type of the assembly feature. It is similar to manually picking the values of lower and upper quantities from table 5-1 while seeing them against the basic dimension value and the tolerance type. For example, the values of lower and upper quantities for a hole AF having basic dimension 52 mm and tolerance type H7 would be 0 mm and 0.030 mm (see example in section 5.3.2 for more details). The inference rule to deduce this kind of knowledge would be as follows:

```
            (<= (hasTolerance ?h (H7 ?q1 ?q2))
                  (and (HoleAF ?h)
                        (hasShapeAttribute ?h ?circular)
                        (Circular ?circular)
```

```
                                    (= ?q1 (mm 0))
                                    (= ?q2 (mm 0.030))
                                    (hasDimension ?h ?q)
                                    (measureLT (mm 50) ?q)
                                    (measureLTE ?q (mm 80))
                                    (hasToleranceType ?h ?t)
                                    (tolTypeH7 ?t)))
:rem "A hole should have tolerance (H7 (0 mm to 0.030 mm)) when it has
a dimension range 50-80 mm."
```

The above rule deduces that any hole AF ?h having basic dimension in between 50 mm and 80 mm would have tolerance (H7 (0 mm) (0.030 mm)). The other conditions applied to this rule are that the tolerance type should be H7, the shape attribute should be circular and types of AF should be hole AF. It suggests that if hole AF is replaced with shaft AF having the same specifications e.g. tolerance type H7, the knowledge base would not show any tolerance quantities as H7 is only applicable to holes.

Similarly, inference rules have been applied to capture the standard tolerance quantities with assembly features having dimensional range from 0 mm to 500 mm as shown in table 5-1. The complete set of axioms can be found in appendix B.3.1.

5.3.4.1.2 Formalization of Assembly Fits

The informal description and representation of assembly fits and related concepts have been discussed in sections 5.3.2.1 and 5.3.3.1.2. It was described that fits are the dimensional relationship between the mating objects e.g. hole AF and shaft AF. Three different types of relations were identified which are: hasClearanceFitWith, hasTransitionFitWith, and hasInterferenceFitWith. These relations were considered bidirectional and were held between two mating objects.

In KFL the relation hasClearanceFitWith can be specified as follows:

```
:Rel hasClearanceFitWith

:Inst BinaryRel

:Inst IrreflexiveBR

:Inst IntensionalRel

:Sig Object Object

:supRel hasFitWith
```

The above relation hasClearanceFitWith is identified as an instance of binary relation as it is held between two classes. The relation is also an instance of irreflexive binary relation implying that the same object cannot have clearance fit with itself. There has to be two different objects to have clearance fit with each other. The relation hasClearanceFitWith is also declared as an instance of intensional relation meaning that the hasClearanceFitWith cannot be asserted in knowledge base and can only be used in inference rules and queries. Hence the knowledge base would determine the assembly fit relation between a hole and a shaft depending upon the type of tolerance and the basic dimensions of mating components. Similarly the relations hasTransitionFitWith and hasInterferenceFitWith has been declared in KFL.

A hole AF and a shaft AF having a particular dimension with a specific tolerance can be assessed to find out the type of fit they would have with each other. For example, the following axiom specifies when a hole AF and a shaft AF should have clearance fit with each other.

```
(<= (hasClearanceFitWith ?h ?s)

            (and (HoleAF ?h)

            (hasShapeAttribute ?h ?circular)

            (Circular ?circular)

            (ShaftAF ?s)

            (hasShapeAttribute ?s ?circular)

            (or (matesWith ?h ?s)

                (and (hasDimension ?h ?d1)

                     (hasDimension ?s ?d2)

                     (= ?d1 ?d2)

                        (matesWith ?h ?s)

                        (hasToleranceType ?h ?tolH8)

                        (hasToleranceType ?s ?tolf7)

                        (tolTypeH8 ?tolH8)

                        (tolTypef7 ?tolf7)))

                  (hasMinAllowableDimension ?h ?holeMinDim)

                  (hasMaxAllowableDimension ?s ?shaftMaxDim)

                  (measureLT ?shaftMaxDim ?holeMinDim)))

:rem "A hole has clearnce Fit With a shaft if the minimum allowable
dimension of hole is larger than the maximum allowable dimension of
shaft."
```

To understand the above axiom, it can be broken down into two parts. The first part is based on the assumption that the tolerance function is user defined function. The knowledge base would deduce the minimum and maximum allowable dimensions on the basis of user defined tolerance quantities. However the user defined tolerance quantities can be associated to any assembly feature consequently any hole can have clearance fit with any shaft even if they are not going to mate with each other. To counter this problem a relation matesWith has been added in the axiom to constrain the hasClearanceFitWith relation to mating assembly features only.

The second part of the axiom which specifies the standard tolerance quantities has been annexed with the first part with an operator OR implying that an assembly feature can have either a user defined tolerance or standard tolerance information provided by the knowledge base. The BS 4500 (1969) takes the hole and shaft sizes as equal and then applies tolerance on these quantities to make their sizes different. Therefore constraint of equal dimensions is also placed in the axiom e.g. the system would only apply standard tolerance on those assembly features which have equal dimensions.

In BS 4500 there are different pairs of tolerance types which can be applied to mating components e.g. H8 and f7, H7 and k6, H7 and p6. When H8 and f7 are applied on a hole AF and shaft AF, this would result in clearance fit (see sheet 4500A in BS 4500). Hence the tolerance pair H8 and f7 have been specified in the axiom to infer hasClearanceFitWith relation between hole AF and shaft AF.

As discussed in section 5.3.2.3, for a clearance fit, the minimum allowable dimension of hole AF should be larger than the maximum allowable dimension of shaft AF. This condition has also been specified at the bottom of the above axiom. Similarly, the semantics of interference and transition fit have been captured using the inference rules (see appendix B.3.1 for more details).

5.3.4.1.3 Formalization of Assembly Allowance

As discussed in section 5.3.2.3, the allowance is the minimum clearance or maximum interference between a hole and a shaft or vice versa. The minimum clearance or maximum interference between the mating features (hole and shaft) would exist when an instance of a hole having minimum allowable dimension mates with an instance of a shaft having maximum allowable dimension. The minimum clearance or maximum interference can simply be calculated by subtracting the minimum allowable dimension of hole, and the maximum allowable dimension of shaft. If the resulting figure is positive (i.e. there exists some space between a hole and shaft) then it is the minimum clearance case. However if the resulting figure is negative implying that the size of shaft is larger than that of hole then it is the maximum interference case.

To capture the semantics of allowance, the following relation has been declared in KFL.

```
:Rel hasAllowanceWith

:Inst TernaryRel

:Inst IntensionalRel

:Sig Object Object Dimension
```

The relation has been declared as an instance of intensional relation which implies that the relation cannot be asserted in the knowledge base as a fact and would only be used in inference rules and queries.

Now to the relation hasAllowanceWith has been used in the following inference rule in order to capture the semantics of allowance from the existing knowledge in the knowledge base.

```
(<= (hasAllowanceWith ?h ?s ?allowance)

                  (and (HoleAF ?h)

                        (hasShapeAttribute ?h ?circular)

                        (Circular ?circular)
```

```
                    (ShaftAF ?s)

                    (Dimension ?allowance)

                    (hasClearanceFitWith ?h ?s)

                    (hasMinAllowableDimension ?h ?qmin)

                    (hasMaxAllowableDimension ?s ?qmax)

                    (measureMinus ?qmin ?qmax ?allowance)))
:rem "A hole hasAllowanceWith a shaft if the minimum allowable
dimension of hole is larger than the maximum allowable dimension of
shaft which is left intensionally and assembly features having
clearance fit have always positive allowance."
```

The above mentioned axiom specifies that if a hole AF and a shaft AF have clearance fit, the allowance between the mating AF would be the difference between the minimum allowable dimension of hole and the maximum allowable dimension of the shaft. Similar rules have been applied to find out the allowance in case of transition fits and interference fits. For more details please see appendix B.3.1.

5.3.4.2 Formalization of Assembly Process Planning Related Concepts

Most of the concepts formalized so far capture design knowledge. This design knowledge not only influences the design decisions but also affects the APP consequences. Hence design systems should be linked with APP systems in order to assess the assemblability issues during the design stage. This section considers the formalization of assembly process, assembly resource, and manufacturing facility concepts to capture APP knowledge.

Assembly process related information can be deduced for a particular type of fit using the inference rules. For example, a hole AF could be assembled with a shaft AF using the press fitting or shrink fitting assembly process when they have interference fit with each other. This type of knowledge can be captured by using the following axiom.

```
(<= (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)

                  (and (HoleAF ?hole)

                        (ShaftAF ?shaft)

                        (or (PressFitting ?assemblyprocess)

                            (ShrinkFitting ?assemblyprocess))

                         (hasInterferenceFitWith ?hole ?shaft)))

:rem "A hole can have press fitting or shrink fitting assembly process
if it has interference fit with the shaft."
```

The above axiom also links design knowledge with the assembly planning knowledge. Similarly axioms have been applied to deduce the possible assembly processes when assembly features have clearance and transition fits. Now once the assembly process is known, the associated assembly resources can be deduced from the knowledge base using the following inference rule.

```
(<= (usesAssemblyResource ?assemblyprocess ?assemblyresource)

      (and (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)

             (HoleAF ?hole)

             (ShaftAF ?shaft)

             (PressFitting ?assemblyprocess)

             (PressFitMachine ?assemblyresource)

             (hasAssemblyResource ?manufacturingfacility       ?
              ?assemblyresource)

            (ManufacturingFacility ?manufacturingfacility)))

:rem "Press fit assembly process can use press fit machine as assembly
resource."
```

The above axiom states that if the possible assembly process is press fitting, then the associated assembly resource would be press fit machine provided that the latter is available in a particular manufacturing facility. Similar rules can be applied to deduce other assembly resources used by different assembly processes. Alternatively it is also possible to determine whether a pair of hole and shaft features can be assembled in a particular manufacturing facility. This kind of knowledge is important when making decisions during the design stage of the product. For example if the default manufacturing facility cannot be used for the product assembly then decisions like purchasing new assembly resources or outsourcing can be considered during the design stage.

The following axiom deduces whether a hole and shaft features can be assembled in a specific manufacturing facility or not.

```
      (<= (isAssembledWithIn ?hole ?shaft ?manufacturingfacility)

            (and (HoleAF ?hole)

            (ShaftAF ?shaft)

            (ManufacturingFacility ?manufacturingfacility)

            (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)

            (isPerformedIn ?assemblyprocess ?manufacturingfacility)

            (hasAssemblyResource              ?manufacturingfacility
            ?assemblyresource)

            (usesAssemblyResource                    ?assemblyprocess
            ?assemblyresource)))

:rem "A hole can be assembled with a shaft in a manufacturing facility
if the later has the required assembly resource."
```

The conditions applied in the above rule for the assembly of hole and shaft feature in a particular manufacturing facility include: the features should have assembly process with each other, that assembly process should be performed in that manufacturing facility, manufacturing facility should have the required assembly resources, and that assembly process should be able to use those assembly resources.

Once the assembly design and assembly process planning related knowledge is formalized, it can be experimentally investigated by loading the ontology in IODE and subsequently asserting facts and making queries. This is explained in chapter 6.

5.4 Summary

This chapter has explored the two main ARO concepts, those of MBOM and assembly feature for assembly knowledge representation and sharing. During the exploration of the MBOM concepts, it has been found that MBOM can have multiple interpretations. Three different interpretations of MBOM have been explained with the example of a butterfly valve assembly. These MBOM interpretations were first informally defined and then represented using the UML diagrams. Finally these concepts have been formalized using the KFL based formal ontological approach.

The second part of the chapter has discussed the assembly feature concept to support knowledge sharing across the assembly domains. It is argued that the assembly feature concept has different implications for the assembly design and assembly process planning domains. It has been explored that how assembly feature is linked with tolerance and assembly fits related assembly design knowledge and with assembly process, assembly resource and manufacturing facility related assembly process planning knowledge. The chapter has also considered tolerance standard BS 4500 and how the concepts related to tolerance standard can be associated with the assembly feature concept.

At first the concepts have been represented in UML to delineate the relationships between these concepts. After that these concepts have been formalized using the KFL based ontological approach. KFL based constraints and rules have been successfully exploited to capture the semantics of the concepts and relationships considered in this chapter.

CHAPTER 6

EXPERIMENTAL INVESTIGATION

6.1 Introduction

This chapter explains the design of the experimental system and explores the experimental investigation of key aspects of the research work proposed in this thesis. The experiments reported in this chapter validate the following research claims described in chapter 3, 4, and 5 of the thesis.

- A set of assembly reference concepts can support assembly knowledge sharing
- The ARO contributes towards the understanding of the need for concept specializations
- The ARO concepts can be used to relate the assembly design and assembly process planning knowledge
- The ARO concepts can provide a base for the development of application specific ontologies for the assembly domain.

This chapter also explores a case study in the automotive sector to test the applicability of the ARO in the wider scope. This chapter is organized as follows:

Section 6.2 briefly describes the design of the experimental system for the ARO framework. Section 6.3 explains the experiments performed to validate the research work proposed in this thesis. Section 6.4 presents a case study in the automotive sector. Finally section 6.5 presents the summary of the chapter 6.

6.2 Design of the Experimental System

The design of the experimental system for the ARO requires the selection of the ontology representation language and the respective appropriate software tool. Knowledge Frame Language (KFL) a Common Logic (CL) based ontological formalism has been used to represent the ARO in computational form. The KFL supports the creation of heavyweight ontologies by providing the syntax and the semantics. Chapter 4 and 5 have provided formal definitions of the key ARO concepts using the KFL code that includes properties, relations, functions and axioms. However the code developed in the KFL requires experimental evaluation.

The KFL code has been written in Notepad++ which is a free source code editor (Ho, 2011). The KFL files written in Notepad++ are then loaded into the experimental software tool IODE (please see appendix A for more detail). The IODE at this point checks the code structure and integrity constraints applied in the code. If the code is correctly structured and integrity constraints are not violated, IODE creates a database for the loaded KFL file.

Once the ontology file is loaded into the IODE, facts can be asserted into the database. Integrity constraints are triggered if the fact assertions violate any of them. Once the facts have been asserted, the ontology can be evaluated by making queries. The whole experimental investigation process is visually shown in figure 6.1.

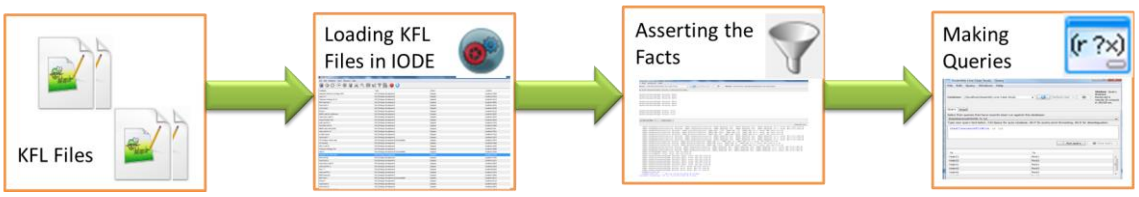


Figure 6.1: Implementation of the ARO for experimental investigation

6.3 Experimentation

A number of experiments have been performed to evaluate the ARO framework. These experiments have been identified to validate various aspects of research framework proposed in this thesis. These experiments have been categorised into following three main experiments.

Experiment 1: To evaluate the MBOM semantics by testing the integrity constraints and to investigate a case of intra-domain assembly process planning knowledge sharing.

Experiment 2: To identify the assembly design related information by using the ARO concept: assembly feature, and the standard tolerance information formalized in chapter 5.

Experiment 3: To evaluate the assembly process planning consequences of product assembly during the assembly design stage and to investigate a case of inter-domain assembly knowledge sharing.

These experiments are explained in the following sections.

6.3.1 Evaluating the MBOM Semantics by Testing the Integrity Constraints and Investigating a Case of Intra-Domain Assembly Process Planning Knowledge Sharing

The purpose of this experiment is to validate the semantics of MBOM interpretations as discussed in section 5.2. This experiment also helps to understand the use of concept specialization levels to support formal representation of concepts and to provide a link for knowledge sharing (section 3.4.2.2). This experiment will demonstrate the assertions of facts to verify that whether the knowledgebase

- Allows the fact assertions when they satisfy the formal definition of the concepts.
- Does not allow the fact assertions when they do not satisfy the formal definition of the concepts.
- Reports the reasons of fact assertions which do not satisfy the formal definition of the concepts.
- Shows that the violation of formal definition is applicable to the related specialized concepts.

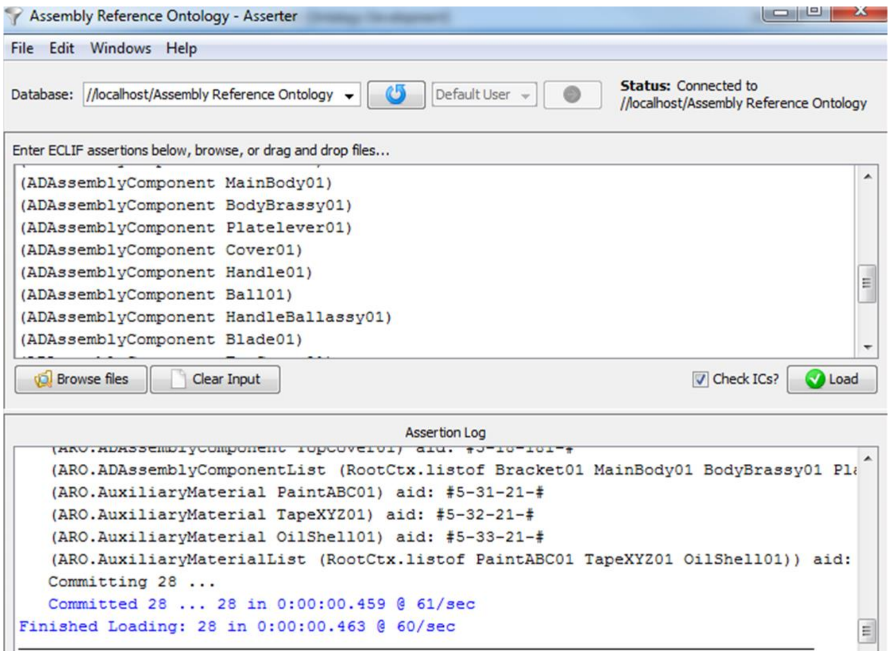
The facts related to the valve assembly (see section 4.3.6 and section 5.2) have been asserted to evaluate the semantics of MBOM. Table 6-1 shows a summary of these facts. The ARAssemblyComponent and ADAssemblyComponent are specialized classes of assembly components as explained in section 5.2. The auxiliary materials are the materials indirectly used to support the assembly of components. The As Required (AR) assembly component list, As Designed (AD) assembly component list and auxiliary materials list have been instantiated with the help of “listof” function as shown in the table 6-1.

Because the facts displayed in table 6-1 do not violate any constraint (please refer to section 5.2.3 for respective constraints) therefore these facts have been successfully asserted in the database as shown in the figure 6.2. Once these facts have been asserted they can be used to test the semantics of MBOM and its specialized classes e.g. MBOMs, MBOMh, and MBOMi.

The constraint attached with MBOM states that it should have an assembly component list. This constraint should also work for the specialized classes of MBOM (e.g. on MBOMs, MBOMh, and MBOMi). For example if an instance of MBOMs is asserted in the database without asserting AD or AR assembly component lists, the system will display an error message reporting that every MBOM should have assembly component list as shown in figure 6.3.

Table 6-1: Instances of assembly components, auxiliary materials and their corresponding lists for butterfly valve assembly.

Classes	Instances
ARAssemblyComponent	Nut01 Nut02 Nut03 Nut04 Nut05 Nut06 Bolt01 Bolt02 Bolt03 Bolt04 Bolt05 Bolt06
ARAssemblyComponentList	(listof Nut01 Nut02 Nut03 Nut04 Nut05 Nut06 Bolt01 Bolt02 Bolt03 Bolt04 Bolt05 Bolt06)
ADAssemblyComponent	Bracket01 MainBody01 BodyBrassy01 Platelever01 Cover01 Handle01 Ball01 HandleBallassy01 Blade01 TopCover01
ADAssemblyComponentList	(listof Bracket01 MainBody01 BodyBrassy01 Platelever01 Handle01 Ball01 HandleBallassy01 Blade01 TopCover01)
AuxiliaryMaterial	PaintABC01 TapeXYZ01 OilShell01
AuxiliaryMaterialList	(listof PaintABC01 TapeXYZ01 OilShell01)



Database: //localhost/Assembly Reference Ontology

Status: Connected to //localhost/Assembly Reference Ontology

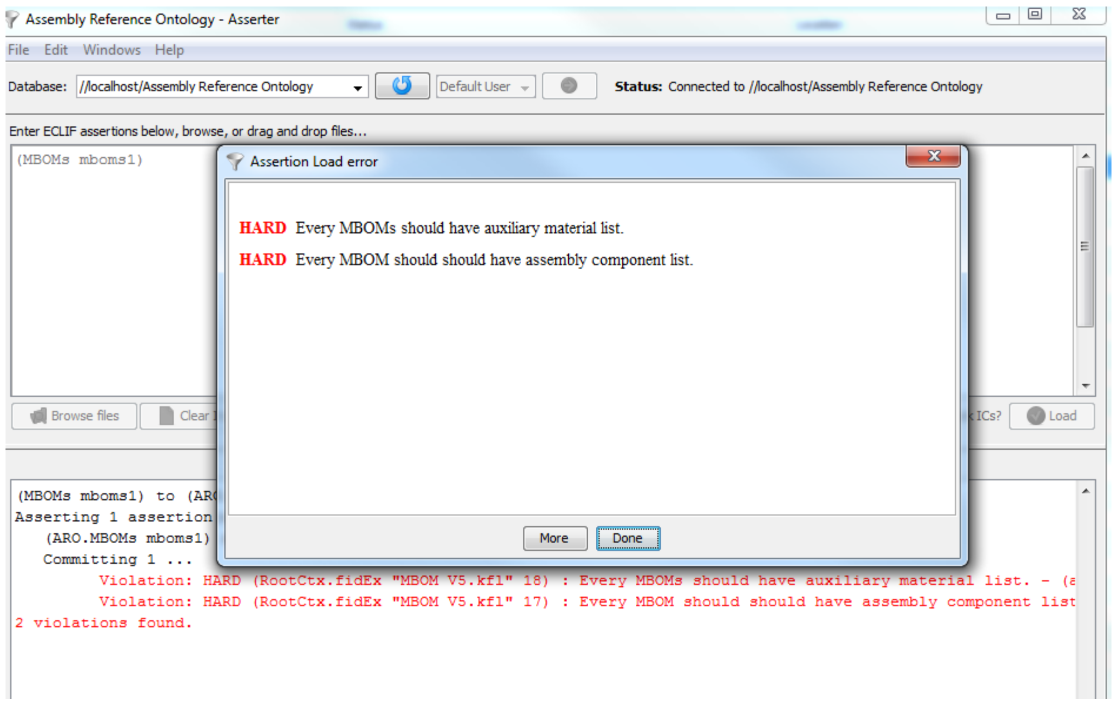
Enter ECLIF assertions below, browse, or drag and drop files...

```
(ADAssemblyComponent MainBody01)
(ADAssemblyComponent BodyBrassy01)
(ADAssemblyComponent Platelever01)
(ADAssemblyComponent Cover01)
(ADAssemblyComponent Handle01)
(ADAssemblyComponent Ball01)
(ADAssemblyComponent HandleBallassy01)
(ADAssemblyComponent Blade01)
```

[✓] Check ICs?

```
   (ARO.ADAssemblyComponentList (RootCtx.listof Bracket01 MainBody01 BodyBrassy01 Pl
   (ARO.AuxiliaryMaterial PaintABC01) aid: #5-31-21-#
   (ARO.AuxiliaryMaterial TapeXYZ01) aid: #5-32-21-#
   (ARO.AuxiliaryMaterial OilShell01) aid: #5-33-21-#
   (ARO.AuxiliaryMaterialList (RootCtx.listof PaintABC01 TapeXYZ01 OilShell01)) aid:
   Committing 28 ...
   Committed 28 ... 28 in 0:00:00.459 @ 61/sec
Finished Loading: 28 in 0:00:00.463 @ 60/sec
```

Figure 6.2: Facts of assembly component, auxiliary materials and their corresponding lists are successfully asserted



Database: //localhost/Assembly Reference Ontology

Default User

Status: Connected to //localhost/Assembly Reference Ontology

Enter ECLIF assertions below, browse, or drag and drop files...

```
(MBOMs mboms1)
```

HARD Every MBOMs should have auxiliary material list.

HARD Every MBOM should should have assembly component list.

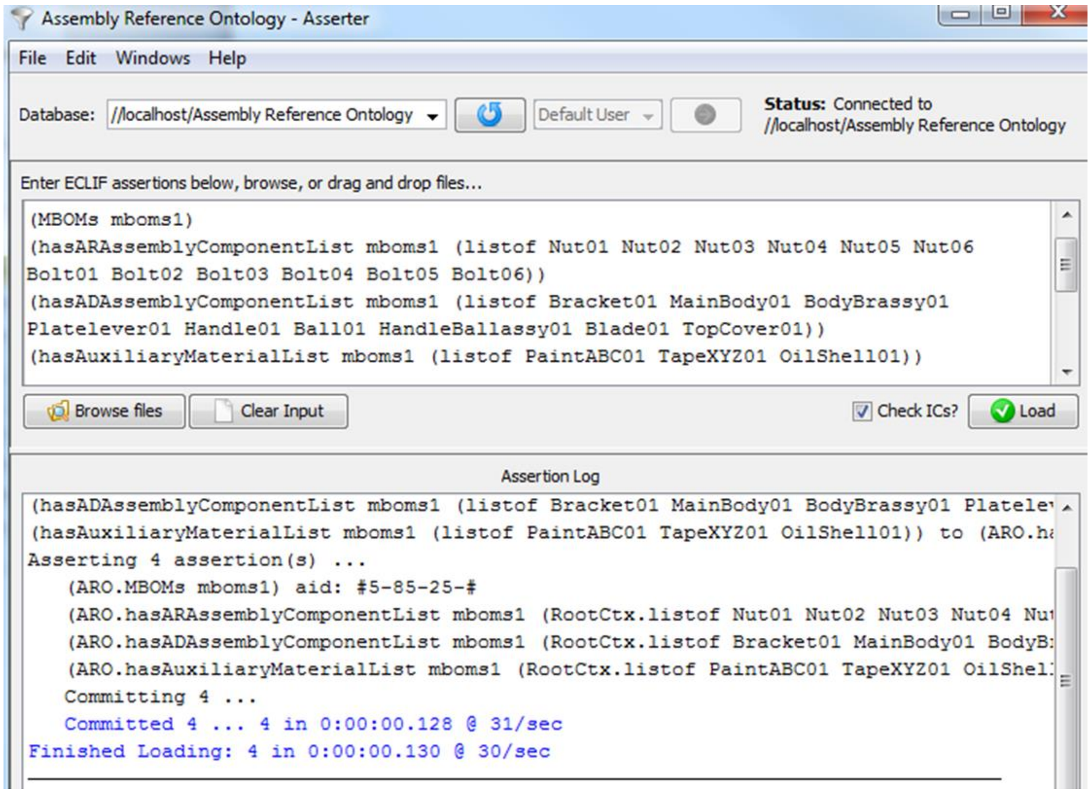
```
(MBOMs mboms1) to (AR
Asserting 1 assertion
   (ARO.MBOMs mboms1)
   Committing 1 ...
        Violation: HARD (RootCtx.fidEx "MBOM V5.kfl" 18) : Every MBOMs should have auxiliary material list. - (a
        Violation: HARD (RootCtx.fidEx "MBOM V5.kfl" 17) : Every MBOM should should have assembly component list
2 violations found.
```

Figure 6.3: MBOMs fact assertion without assembly component list and auxiliary material list

The IC violation caused due to the absence of assembly component list in figure 6.3 suggests that any instance of a specialized level of a concept which does not satisfy the formal definition of its parent class will also violate the constraints applied on the parent class. However as the parent classes are more generic as compared to their child classes therefore more constraints can be applied on the child classes which can then be used for specific applications.

Figure 6.3 also shows the IC violated due to the lack of auxiliary material list. This is because there is an IC attached with the MBOMs (MBOMs is specialized from MBOM) as well which states that whenever an MBOM exists, it should also have auxiliary material list (please see section 5.2.3). Therefore when MBOMs is asserted with the assembly component lists and auxiliary materials lists the system will accept the MBOMs facts assertions as shown in the figure 6.4.

It is to be noted that MBOMs asserted without assembly component list violates IC due to its parent class MBOM. Whereas MBOMs asserted without auxiliary material list is due to the IC applied on the concept itself. However MBOMs can be instantiated successfully when asserted with AR assembly component list or AD assembly component list or both (as both of these concepts are specialized from the assembly component list) and the auxiliary material list as shown in figure 6.4.


Assembly Reference Ontology - Asserter
File Edit Windows Help
Database: //localhost/Assembly Reference Ontology
Default User
Status: Connected to //localhost/Assembly Reference Ontology
Enter ECLIF assertions below, browse, or drag and drop files...
Browse files
Clear Input
Check ICs?
Load
Assertion Log

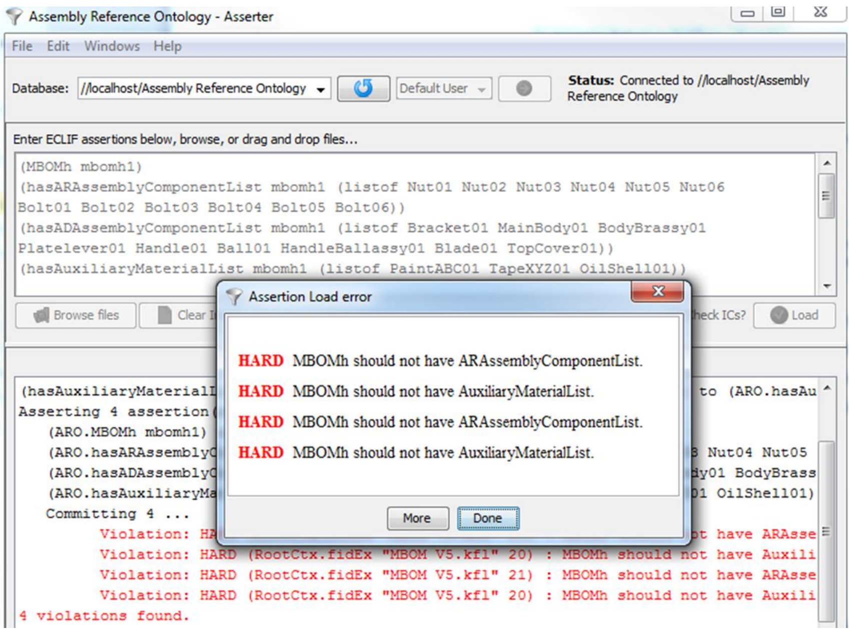
```
(MBOMs mboms1)
(hasARAssemblyComponentList mboms1 (listof Nut01 Nut02 Nut03 Nut04 Nut05 Nut06
Bolt01 Bolt02 Bolt03 Bolt04 Bolt05 Bolt06))
(hasADAssemblyComponentList mboms1 (listof Bracket01 MainBody01 BodyBrassy01
Platelever01 Handle01 Ball01 HandleBallassy01 Blade01 TopCover01))
(hasAuxiliaryMaterialList mboms1 (listof PaintABC01 TapeXYZ01 OilShell01))
```

```
(hasADAssemblyComponentList mboms1 (listof Bracket01 MainBody01 BodyBrassy01 Platele
(hasAuxiliaryMaterialList mboms1 (listof PaintABC01 TapeXYZ01 OilShell01)) to (ARO.ha
Asserting 4 assertion(s) ...
   (ARO.MBOMs mboms1) aid: #5-85-25-#
   (ARO.hasARAssemblyComponentList mboms1 (RootCtx.listof Nut01 Nut02 Nut03 Nut04 Nu
   (ARO.hasADAssemblyComponentList mboms1 (RootCtx.listof Bracket01 MainBody01 BodyB
   (ARO.hasAuxiliaryMaterialList mboms1 (RootCtx.listof PaintABC01 TapeXYZ01 OilShel
   Committing 4 ...
   Committed 4 ... 4 in 0:00:00.128 @ 31/sec
Finished Loading: 4 in 0:00:00.130 @ 30/sec
```

Figure 6.4: Facts asserted for MBOMs

Similarly whenever instances of MBOMh and MBOMi are asserted without the assembly component list, the system displays an error message suggesting to include the assembly component list with the instances of MBOMh and MBOMi.

If the facts in figure 6.4 are asserted for MBOMh, the system will display the error as shown in figure 6.5. This is because of the axioms applied on MBOMh to constrain its semantics (please refer back to figure 5.3). These constraints actually avert any attempt made to assert the AR assembly component list and the auxiliary material list as evident from figure 6.5. However when MBOMh is asserted with AD assembly component list only, the system accepts it as shown in figure 6.6.

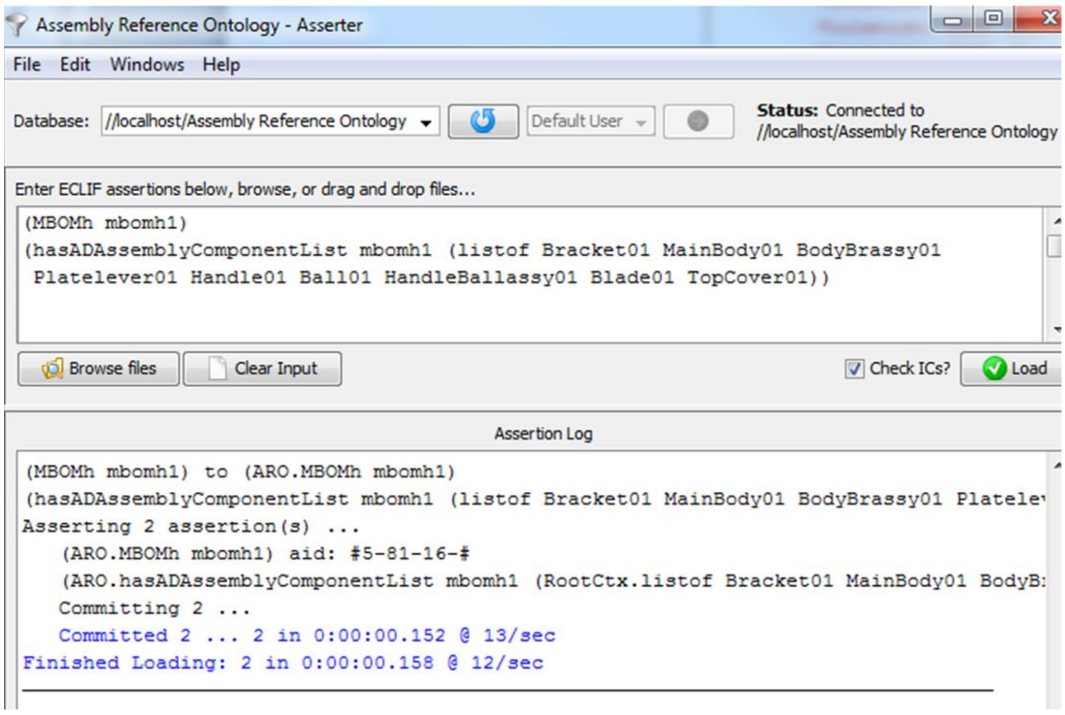

Assembly Reference Ontology - Asserter
File Edit Windows Help
Database: //localhost/Assembly Reference Ontology
Default User
Status: Connected to //localhost/Assembly Reference Ontology
Enter ECLIF assertions below, browse, or drag and drop files...
Browse files
Clear I
heck ICs?
Load
Assertion Load error
More
Done

```
(MBOMh mbomh1)
(hasARAssemblyComponentList mbomh1 (listof Nut01 Nut02 Nut03 Nut04 Nut05 Nut06
Bolt01 Bolt02 Bolt03 Bolt04 Bolt05 Bolt06))
(hasADAssemblyComponentList mbomh1 (listof Bracket01 MainBody01 BodyBrassy01
Platelever01 Handle01 Ball01 HandleBallassy01 Blade01 TopCover01))
(hasAuxiliaryMaterialList mbomh1 (listof PaintABC01 TapeXYZ01 OilShell01))
```

```
HARD MBOMh should not have ARAssemblyComponentList.
HARD MBOMh should not have AuxiliaryMaterialList.
HARD MBOMh should not have ARAssemblyComponentList.
HARD MBOMh should not have AuxiliaryMaterialList.
```

```
(hasAuxiliaryMaterialI                                          to (ARO.hasAu
Asserting 4 assertion(
   (ARO.MBOMh mbomh1)
   (ARO.hasARAssemblyC                                          3 Nut04 Nut05
   (ARO.hasADAssemblyC                                          dy01 BodyBrass
   (ARO.hasAuxiliaryMa                                          01 OilShell01)
   Committing 4 ...
         Violation: HA                                          ot have ARAsse
         Violation: HARD (RootCtx.fidEx "MBOM V5.kfl" 20) : MBOMh should not have Auxili
         Violation: HARD (RootCtx.fidEx "MBOM V5.kfl" 21) : MBOMh should not have ARAsse
         Violation: HARD (RootCtx.fidEx "MBOM V5.kfl" 20) : MBOMh should not have Auxili
4 violations found.
```

Figure 6.5: IC violations caused due to AR assembly component list and auxiliary material list when asserting the facts for MBOMh

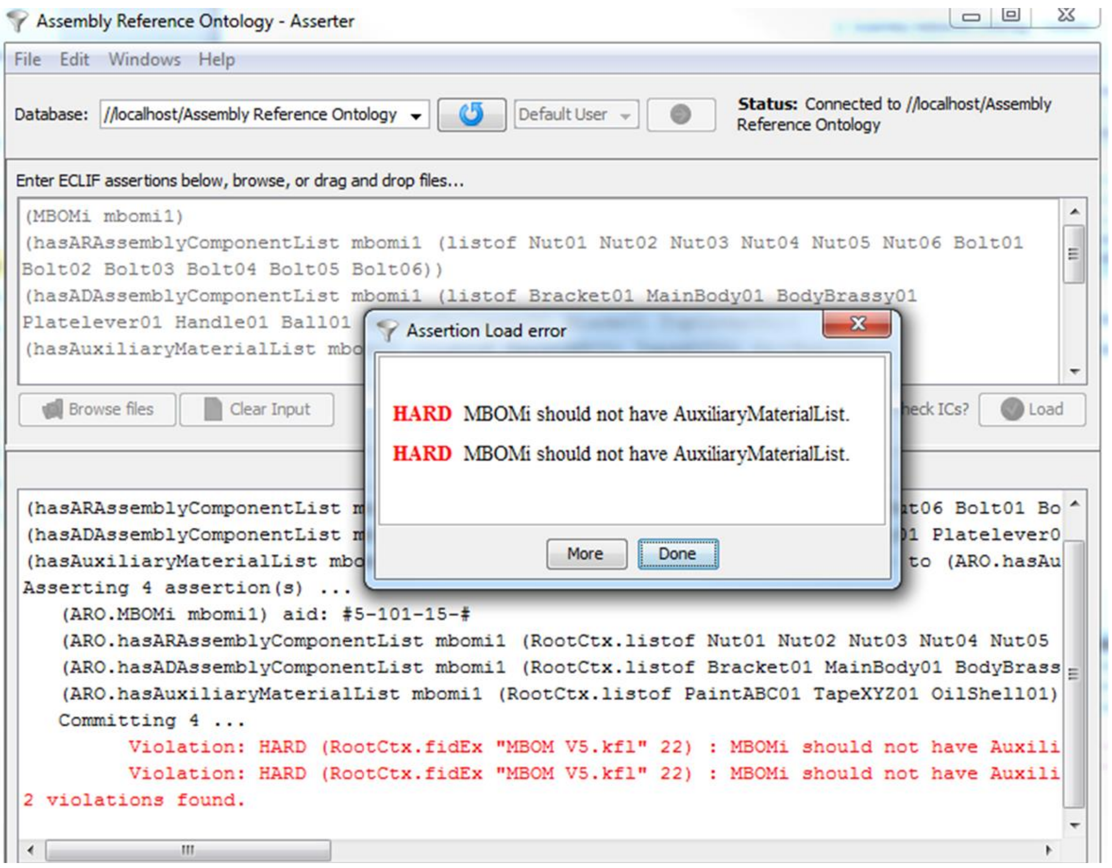

Assembly Reference Ontology - Asserter
File Edit Windows Help
Database: //localhost/Assembly Reference Ontology
Default User
Status: Connected to //localhost/Assembly Reference Ontology
Enter ECLIF assertions below, browse, or drag and drop files...
Browse files
Clear Input
Check ICs?
Load
Assertion Log

```
(MBOMh mbomh1)
(hasADAssemblyComponentList mbomh1 (listof Bracket01 MainBody01 BodyBrassy01
 Platelever01 Handle01 Ball01 HandleBallassy01 Blade01 TopCover01))
```

```
(MBOMh mbomh1) to (ARO.MBOMh mbomh1)
(hasADAssemblyComponentList mbomh1 (listof Bracket01 MainBody01 BodyBrassy01 Platele
Asserting 2 assertion(s) ...
   (ARO.MBOMh mbomh1) aid: #5-81-16-#
   (ARO.hasADAssemblyComponentList mbomh1 (RootCtx.listof Bracket01 MainBody01 BodyB
   Committing 2 ...
   Committed 2 ... 2 in 0:00:00.152 @ 13/sec
Finished Loading: 2 in 0:00:00.158 @ 12/sec
```

Figure 6.6: Successful assertion of MBOMh facts

Finally if the facts in figure 6.4 are asserted for MBOMi, the system will also display an error message. The IC violation in this case has been observed because auxiliary material list was also asserted for MBOMi. It is clear in figure 5.3 (of chapter 5) that MBOMi should not have auxiliary material list. Hence the system has returned the expected results by not allowing the assertion of auxiliary material list for MBOMi as shown in figure 6.7.



Database: //localhost/Assembly Reference Ontology
Default User
Status: Connected to //localhost/Assembly Reference Ontology

Enter ECLIF assertions below, browse, or drag and drop files...

```
(MBOMi mbomi1)
(hasARAssemblyComponentList mbomi1 (listof Nut01 Nut02 Nut03 Nut04 Nut05 Nut06 Bolt01
Bolt02 Bolt03 Bolt04 Bolt05 Bolt06))
(hasADAssemblyComponentList mbomi1 (listof Bracket01 MainBody01 BodyBrassy01
Platelever01 Handle01 Ball01
(hasAuxiliaryMaterialList mbo
```

Assertion Load error

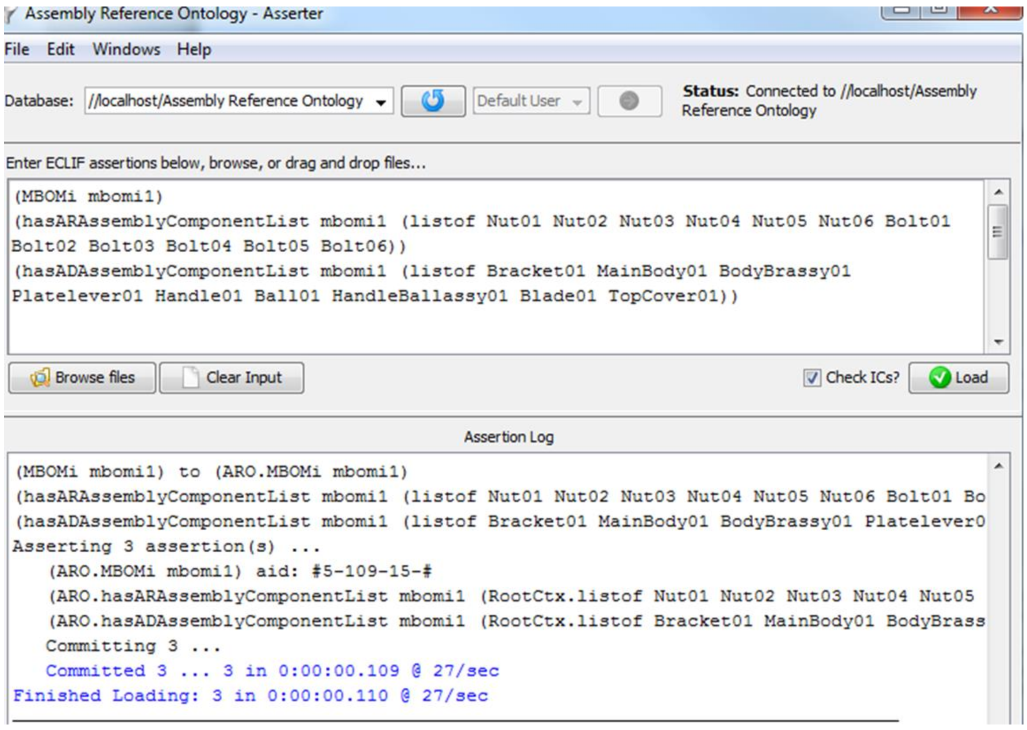
HARD MBOMi should not have AuxiliaryMaterialList.

HARD MBOMi should not have AuxiliaryMaterialList.

```
(hasARAssemblyComponentList m                                              t06 Bolt01 Bo
(hasADAssemblyComponentList m                                              01 Platelever0
(hasAuxiliaryMaterialList mbo                                              to (ARO.hasAu
Asserting 4 assertion(s) ...
   (ARO.MBOMi mbomi1) aid: #5-101-15-#
   (ARO.hasARAssemblyComponentList mbomi1 (RootCtx.listof Nut01 Nut02 Nut03 Nut04 Nut05
   (ARO.hasADAssemblyComponentList mbomi1 (RootCtx.listof Bracket01 MainBody01 BodyBrass
   (ARO.hasAuxiliaryMaterialList mbomi1 (RootCtx.listof PaintABC01 TapeXYZ01 OilShell01)
   Committing 4 ...
        Violation: HARD (RootCtx.fidEx "MBOM V5.kfl" 22) : MBOMi should not have Auxili
        Violation: HARD (RootCtx.fidEx "MBOM V5.kfl" 22) : MBOMi should not have Auxili
2 violations found.
```

Figure 6.7: IC violated due to assertion of auxiliary material list for MBOMi

Now if the auxiliary material list is removed and the AR assembly component list and AD assembly component list are asserted, the system will accept the assertion as shown in figure 6.8. It is evident from these assertions that the system only accepts those facts which comply with the formal definitions of the concepts. Once the facts have been successfully asserted, queries can be made to further validate the semantics of MBOM concepts.


Assembly Reference Ontology - Asserter
File Edit Windows Help
Database: //localhost/Assembly Reference Ontology
Default User
Status: Connected to //localhost/Assembly Reference Ontology
Enter ECLIF assertions below, browse, or drag and drop files...
Browse files
Clear Input
Check ICs?
Load
Assertion Log

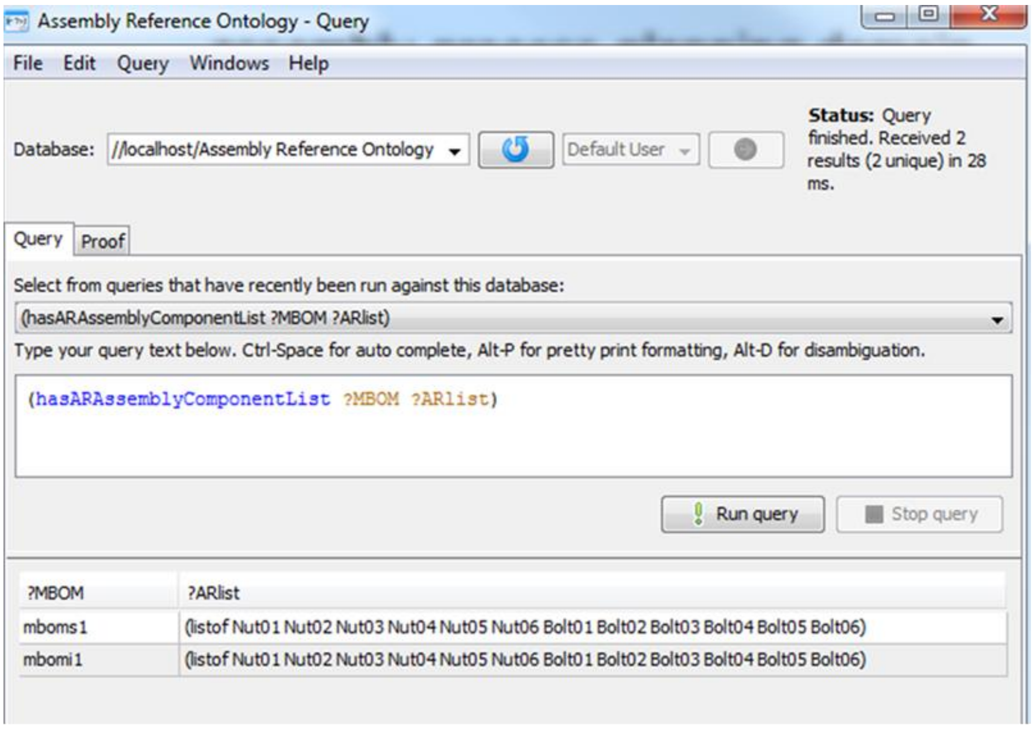
```
(MBOMi mbomi1)
(hasARAssemblyComponentList mbomi1 (listof Nut01 Nut02 Nut03 Nut04 Nut05 Nut06 Bolt01
Bolt02 Bolt03 Bolt04 Bolt05 Bolt06))
(hasADAssemblyComponentList mbomi1 (listof Bracket01 MainBody01 BodyBrassy01
Platelever01 Handle01 Ball01 HandleBallassy01 Blade01 TopCover01))
```

```
(MBOMi mbomi1) to (ARO.MBOMi mbomi1)
(hasARAssemblyComponentList mbomi1 (listof Nut01 Nut02 Nut03 Nut04 Nut05 Nut06 Bolt01 Bo
(hasADAssemblyComponentList mbomi1 (listof Bracket01 MainBody01 BodyBrassy01 Platelever0
Asserting 3 assertion(s) ...
   (ARO.MBOMi mbomi1) aid: #5-109-15-#
   (ARO.hasARAssemblyComponentList mbomi1 (RootCtx.listof Nut01 Nut02 Nut03 Nut04 Nut05
   (ARO.hasADAssemblyComponentList mbomi1 (RootCtx.listof Bracket01 MainBody01 BodyBrass
   Committing 3 ...
   Committed 3 ... 3 in 0:00:00.109 @ 27/sec
Finished Loading: 3 in 0:00:00.110 @ 27/sec
```

Figure 6.8: Successful assertion of facts for MBOMi

For instance, if a query is made to find out the MBOM having AR assembly component list, it will return the results for MBOMs and MBOMi as shown in figure 6.9 (a). The query does not show MBOMh because the AR assembly component list was not allowed to be asserted. The next query asks to find out instances of MBOM which have AD assembly component list. The system returns instances of MBOMs, MBOMh and MBOMi along with the instance of AD assembly component list as shown in figure 6.9 (b). This is because all MBOM specialized classes have AD assembly component list. The last query shown in figure 6.9 (c) is made to find out an instance of MBOM which have auxiliary material list. The system returns an instance of MBOMs along with an instance of auxiliary material list. This shows that only MBOMs has auxiliary material list.

The queries results suggest that the only common list found between MBOMs, MBOMh and MBOMi is the AD assembly component list. This implies that AD assembly component list can provide a link between all three MBOM classes which can subsequently support assembly knowledge sharing across the assembly process planning domain.


Assembly Reference Ontology - Query
File Edit Query Windows Help
Database: //localhost/Assembly Reference Ontology
Default User
Query Proof
Run query
Stop query

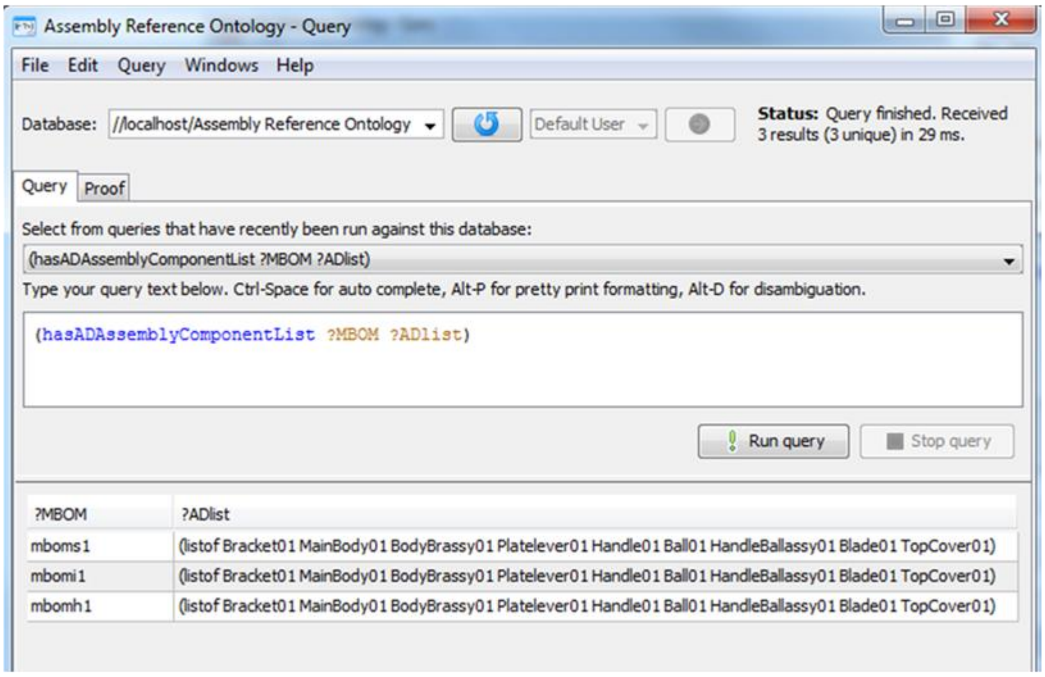
Status: Query finished. Received 2 results (2 unique) in 28 ms.

Select from queries that have recently been run against this database:
(hasARAssemblyComponentList ?MBOM ?ARlist)
Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(hasARAssemblyComponentList ?MBOM ?ARlist)
```

?MBOM	?ARlist
mboms1	(listof Nut01 Nut02 Nut03 Nut04 Nut05 Nut06 Bolt01 Bolt02 Bolt03 Bolt04 Bolt05 Bolt06)
mbomi1	(listof Nut01 Nut02 Nut03 Nut04 Nut05 Nut06 Bolt01 Bolt02 Bolt03 Bolt04 Bolt05 Bolt06)

(a) Query for AR assembly component list


Assembly Reference Ontology - Query
File Edit Query Windows Help
Database: //localhost/Assembly Reference Ontology
Default User
Query Proof
Run query
Stop query

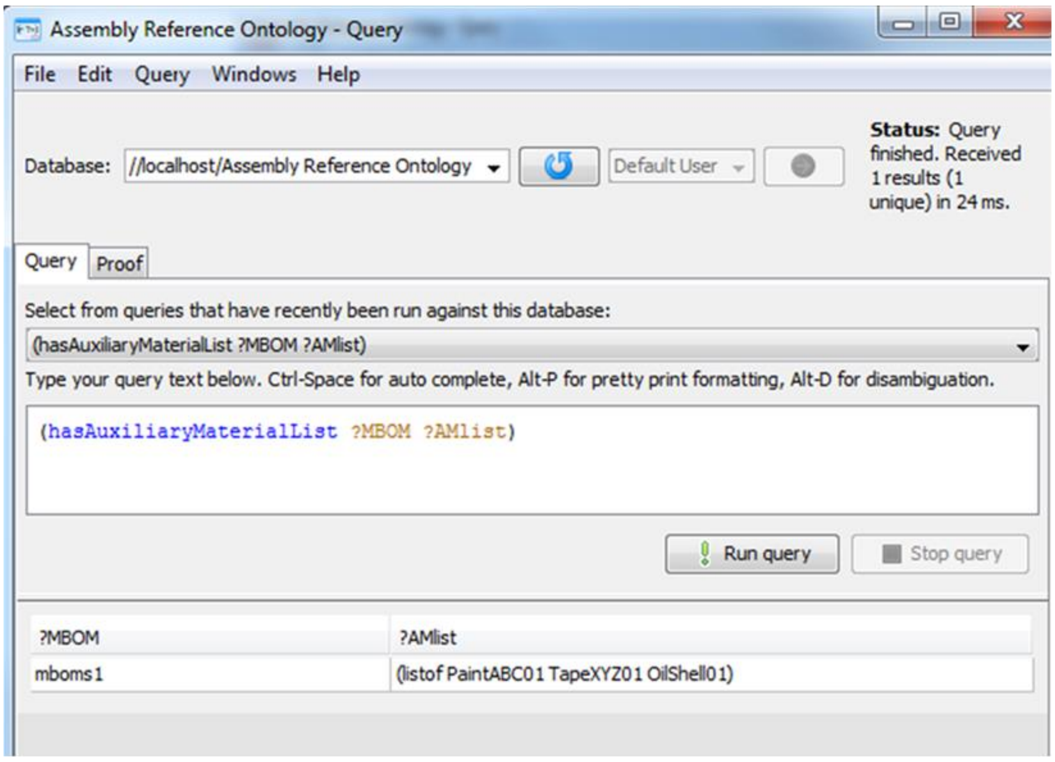
Status: Query finished. Received 3 results (3 unique) in 29 ms.

Select from queries that have recently been run against this database:
(hasADAssemblyComponentList ?MBOM ?ADlist)
Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(hasADAssemblyComponentList ?MBOM ?ADlist)
```

?MBOM	?ADlist
mboms1	(listof Bracket01 MainBody01 BodyBrassy01 Platelever01 Handle01 Ball01 HandleBallassy01 Blade01 TopCover01)
mbomi1	(listof Bracket01 MainBody01 BodyBrassy01 Platelever01 Handle01 Ball01 HandleBallassy01 Blade01 TopCover01)
mbomh1	(listof Bracket01 MainBody01 BodyBrassy01 Platelever01 Handle01 Ball01 HandleBallassy01 Blade01 TopCover01)

(b) Query for AD assembly component list


Assembly Reference Ontology - Query
File Edit Query Windows Help
Default User
Query
Proof
Run query
Stop query

Database: //localhost/Assembly Reference Ontology

Status: Query finished. Received 1 results (1 unique) in 24 ms.

Select from queries that have recently been run against this database:

(hasAuxiliaryMaterialList ?MBOM ?AMlist)

Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(hasAuxiliaryMaterialList ?MBOM ?AMlist)
```

?MBOM	?AMlist
mboms1	(listof PaintABC01 TapeXYZ01 OilShell01)

(c) Query for auxiliary material list

Figure 6.9: Queries made to find out AR assembly component list, AD assembly component list and auxiliary material list.

It is evident from the results of this experiment that the knowledgebase system is capable of understanding the semantics of MBOM concepts and therefore does not allow assertions which do not follow the formal definitions of the MBOM concepts. Furthermore, the results have also shown that formal definition of the concepts are inherited by the specialized classes and that the specialized concepts e.g. MBOMs, MBOMh, MBOMi cannot violate the definitions of their parent classes e.g. MBOM.

The results have also revealed that more constraints can be applied to the specialized concepts to exploit them for specific applications. This also suggests that the ARO concepts can be specialized into different application specific concepts by applying axioms to control their semantics.

6.3.2 Identifying the Assembly Design Related Information by Using the ARO Concept Assembly Feature and the Standard Tolerance Information Formalized in Chapter 5

The purpose of this experiment is to demonstrate that the ARO supports the capture of the design viewpoint of assembly feature which subsequently can be linked with the assembly process planning viewpoint of the assembly feature concept to share assembly knowledge. The experiment also validates the formal semantics of design information described in section 5.3.4 and the information deduced from a set of axioms.

This experiment uses the assembly feature related facts based on the journal bearing assembly shown in figure 5.5 in chapter 5. Once the KFL file has been loaded successfully into the IODE, the facts shown in table 6-2 were asserted to populate the assembly design related information.

Table 6-2: Assembly design related facts asserted for journal bearing assembly

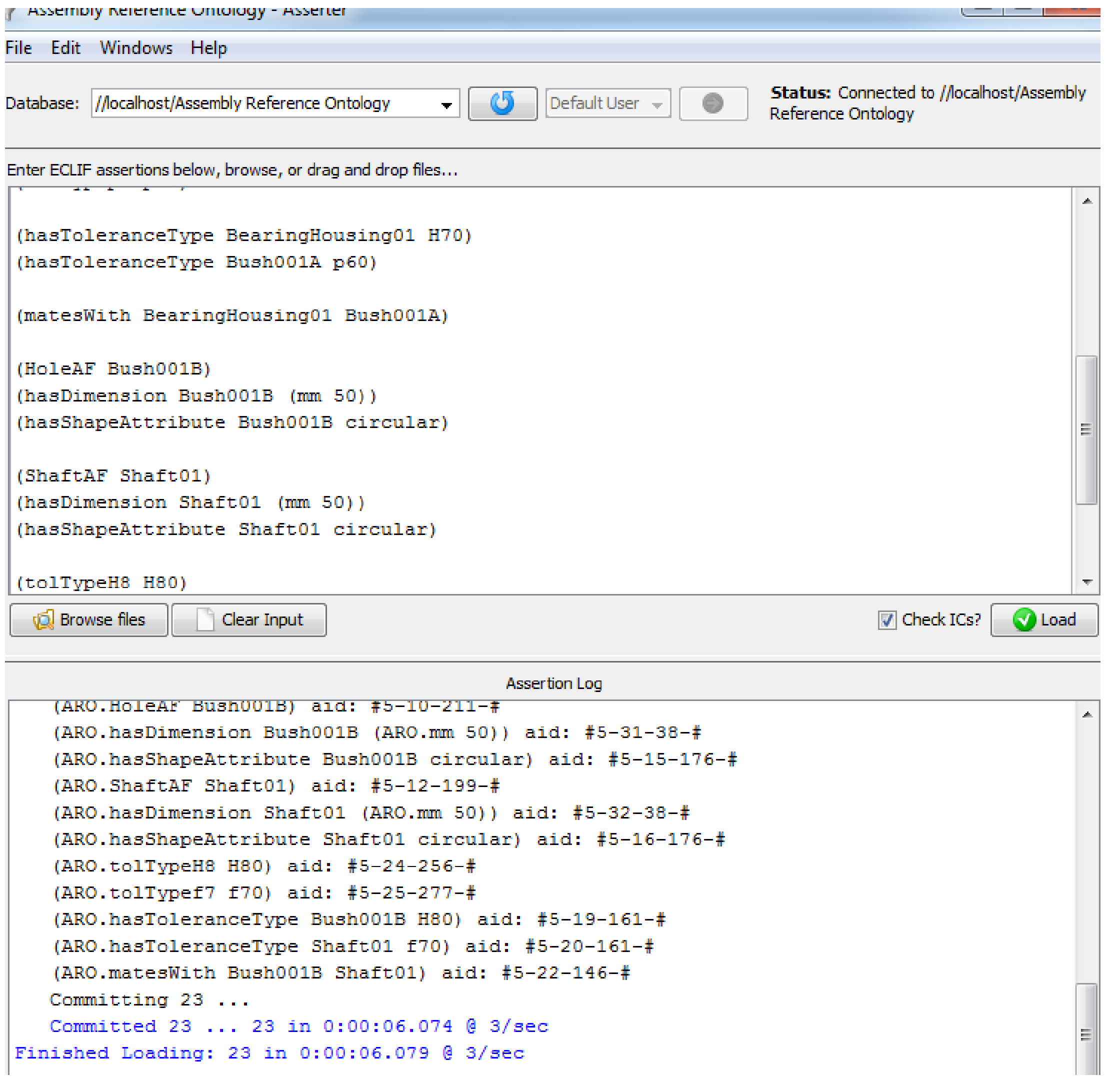
Classes and Relations	Instances
HoleAF	BearingHousing01 Bush001B
ShaftAF	Bush001A Shaft01
hasDimension	BearingHousing01 (mm 52) Bush001A (mm 52) Bush001B (mm 50) Shaft01 (mm 50)
hasShapeAttribute	BearingHousing01 circular Bush001A circular Bush001B circular Shaft01 circular
hasToleranceType	BearingHousing01 H70 Bush001A p60 Bush001B H80 Shaft01 f70
matesWith	BearingHousing01 Bush001A Bush001B Shaft01

BearingHousing01, Bush001A, Bush001B, and Shaft01 represent the instances of four assembly features of the journal bearing assembly shown in figure 5.5 of chapter 5. The basic dimension of these features has been populated with the help of the relation “hasDimension” where the first argument represents the assembly feature and the second argument specifies the linear dimension (expressed in mm measurement function). Four different measurement functions: micron, mm, cm, and m have been defined. These functions are inter-convertible and the ontology allows conversions as described in section 5.3.4.1.1 of chapter 5.

The shape attributes of all four assembly features described in table 6-2 have been populated as circular. This is because the standard tolerance information based on BS 4500 is only applicable to circular hole and shaft assembly features. The other input requires the tolerances for the assembly features be asserted. The ARO allows user defined tolerance as well as the standard tolerance (based on BS 4500). In this experiment it is intended to use the standard tolerance and for this reason it requires to specify the tolerance type.

The assembly features BearingHousing01 and Bush001A have been allocated H7 and p6 tolerance types respectively. Furthermore the assembly features Bush001B and Shaft01 have been allocated H8 and f7 tolerance types. It is important to mention that the tolerance types starting with the upper case e.g. H7, and H8 represent the hole assembly features while the ones staring with small case e.g. p6, and f7 represent shaft assembly features. The relation “matesWith” has been instantiated between the features BearingHousing01 and Bush001A, and between the features Bush001B and Shaft01 as shown in table 6-2. The facts summarized in table 6-2 have been asserted successfully in IODE as shown in figure 6.10.

Once the facts have been asserted successfully in the database, queries can be made to find out the assembly design related information. Information related to tolerance, maximum and minimum allowable dimensions, types of fits formed between assembly features, allowance, and allowable dimensional value or range of values can be determined using the knowledge base.


Assembly Reference Ontology - Asserter
File Edit Windows Help
Database: //localhost/Assembly Reference Ontology
Default User
Status: Connected to //localhost/Assembly Reference Ontology
Enter ECLIF assertions below, browse, or drag and drop files...
Browse files
Clear Input
Check ICs?
Load
Assertion Log

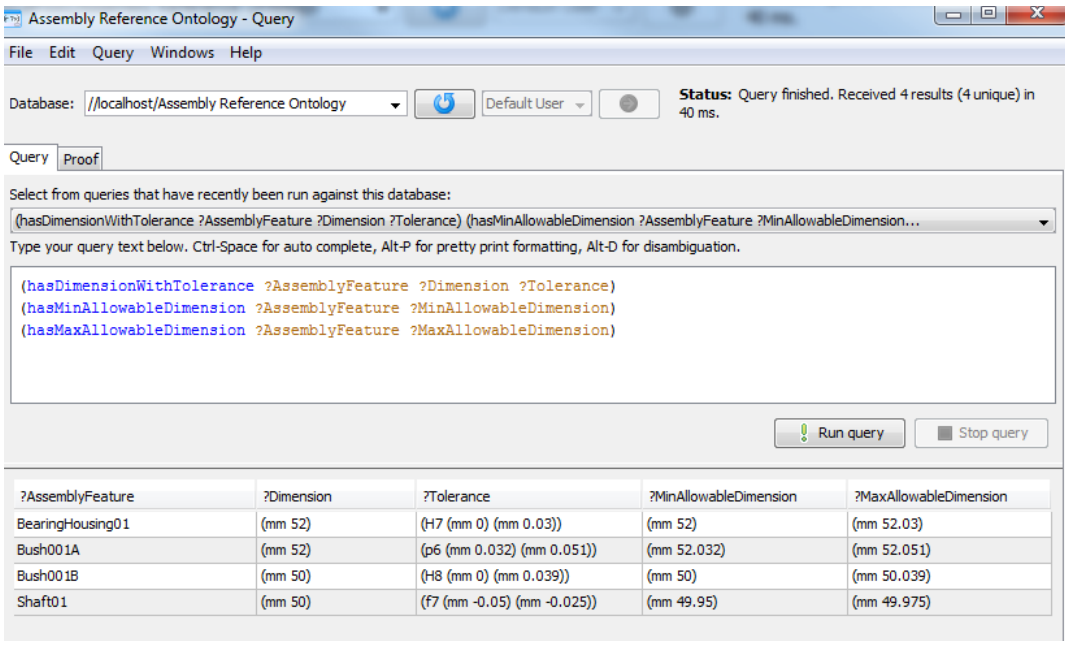
```
(hasToleranceType BearingHousing01 H70)
(hasToleranceType Bush001A p60)

(matesWith BearingHousing01 Bush001A)

(HoleAF Bush001B)
(hasDimension Bush001B (mm 50))
(hasShapeAttribute Bush001B circular)

(ShaftAF Shaft01)
(hasDimension Shaft01 (mm 50))
(hasShapeAttribute Shaft01 circular)

(tolTypeH8 H80)
```

```
    (ARO.HoleAF Bush001B) aid: #5-10-211-#
    (ARO.hasDimension Bush001B (ARO.mm 50)) aid: #5-31-38-#
    (ARO.hasShapeAttribute Bush001B circular) aid: #5-15-176-#
    (ARO.ShaftAF Shaft01) aid: #5-12-199-#
    (ARO.hasDimension Shaft01 (ARO.mm 50)) aid: #5-32-38-#
    (ARO.hasShapeAttribute Shaft01 circular) aid: #5-16-176-#
    (ARO.tolTypeH8 H80) aid: #5-24-256-#
    (ARO.tolTypef7 f70) aid: #5-25-277-#
    (ARO.hasToleranceType Bush001B H80) aid: #5-19-161-#
    (ARO.hasToleranceType Shaft01 f70) aid: #5-20-161-#
    (ARO.matesWith Bush001B Shaft01) aid: #5-22-146-#
   Committing 23 ...
   Committed 23 ... 23 in 0:00:06.074 @ 3/sec
Finished Loading: 23 in 0:00:06.079 @ 3/sec
```

Figure 6.10: Successful assertion of facts described in table 6-2

Figure 6.11 shows three different queries made in IODE to find out the information related to the tolerance, minimum and maximum allowable dimensions of assembly features. The first part of the query asks to find out the tolerance quantities for all the assembly features which exist in the database (currently only four assembly features related to journal bearing assembly are populated as shown in table 6-2). In response to this query, the system returns the tolerance quantities allocated to the assembly features as shown in the third column of figure 6.12. It is worth noticing that all the tolerances allocated to assembly features are standard tolerances e.g. H7, p6, H8, f7. This is because of the standard tolerance types which have been asserted (please see table 6-2) and consequently the system returns tolerance quantities (at the back end these tolerance quantities have been deduced using the inference axioms described in 5.3.4.1).

Similarly two other queries have been made to determine the minimum and maximum allowable dimensions of assembly features as shown in figure 6.12. These minimum and maximum allowable dimensions have been deduced by adding the lower and upper quantities in the basic dimensions of assembly features. It is worth mentioning that all the measurement units are represented in millimetres (mm) as all quantities in BS 4500 standard are specified in mm. However queries can also be made in other measurement functions as well.



Database: //localhost/Assembly Reference Ontology

Status: Query finished. Received 4 results (4 unique) in 40 ms.

Select from queries that have recently been run against this database:

(hasDimensionWithTolerance ?AssemblyFeature ?Dimension ?Tolerance) (hasMinAllowableDimension ?AssemblyFeature ?MinAllowableDimension...

Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

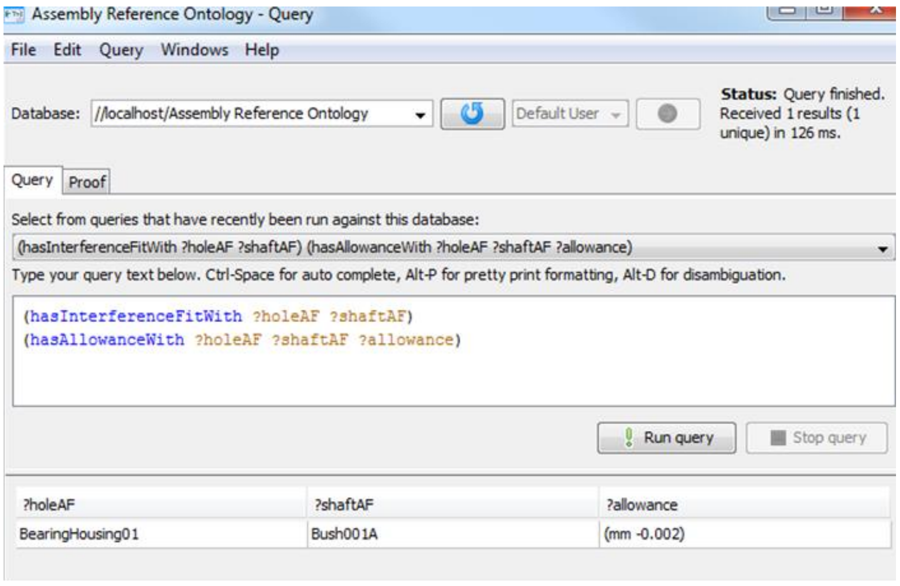
```
(hasDimensionWithTolerance ?AssemblyFeature ?Dimension ?Tolerance)
(hasMinAllowableDimension ?AssemblyFeature ?MinAllowableDimension)
(hasMaxAllowableDimension ?AssemblyFeature ?MaxAllowableDimension)
```

?AssemblyFeature	?Dimension	?Tolerance	?MinAllowableDimension	?MaxAllowableDimension
BearingHousing01	(mm 52)	(H7 (mm 0) (mm 0.03))	(mm 52)	(mm 52.03)
Bush001A	(mm 52)	(p6 (mm 0.032) (mm 0.051))	(mm 52.032)	(mm 52.051)
Bush001B	(mm 50)	(H8 (mm 0) (mm 0.039))	(mm 50)	(mm 50.039)
Shaft01	(mm 50)	(f7 (mm -0.05) (mm -0.025))	(mm 49.95)	(mm 49.975)

Figure 6.11: Queries made to find out the tolerance, minimum and maximum allowable dimensions for journal bearing assembly features

Other queries may include finding out the types of assembly fits between a hole and a shaft assembly feature, to determine the allowance between a hole and a shaft assembly feature or finding out the assembly features whose dimensional specifications are within a range of allowable dimensions.

The fits and allowance related queries displayed in figure 6.12 demonstrate that the hole assembly feature BearingHousing01 has interference fit with the shaft assembly feature Bush001A, and the hole assembly feature Bush001B has clearance fit with the shaft assembly feature Shaft01. The knowledgebase system has deduced these types of fits on the basis of inference rules described in section 5.3.4.1.


Assembly Reference Ontology - Query
File Edit Query Windows Help
Query
Proof
Run query
Stop query

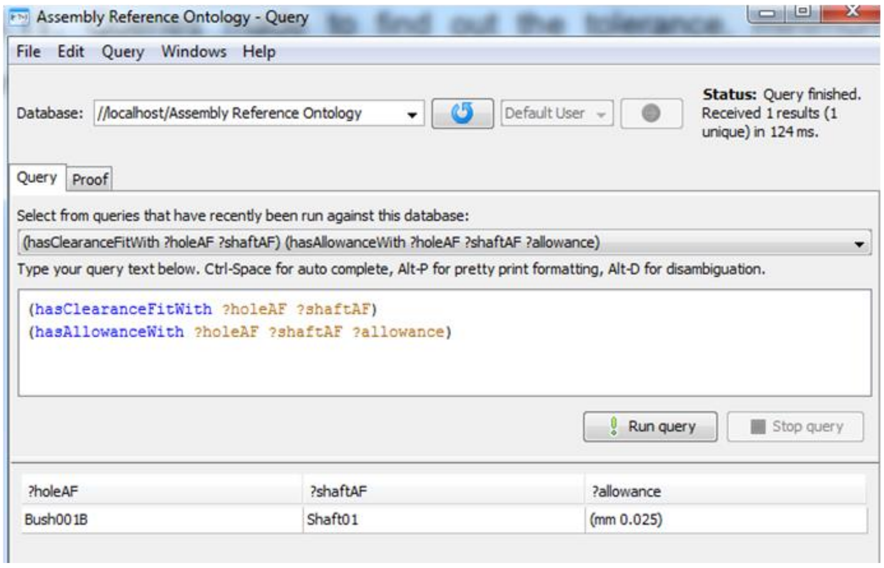
Database: //localhost/Assembly Reference Ontology Default User

Status: Query finished. Received 1 results (1 unique) in 126 ms.

Select from queries that have recently been run against this database:
(hasInterferenceFitWith ?holeAF ?shaftAF) (hasAllowanceWith ?holeAF ?shaftAF ?allowance)
Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(hasInterferenceFitWith ?holeAF ?shaftAF)
(hasAllowanceWith ?holeAF ?shaftAF ?allowance)
```

?holeAF	?shaftAF	?allowance
BearingHousing01	Bush001A	(mm -0.002)


Assembly Reference Ontology - Query
File Edit Query Windows Help
Query
Proof
Run query
Stop query

Database: //localhost/Assembly Reference Ontology Default User

Status: Query finished. Received 1 results (1 unique) in 124 ms.

Select from queries that have recently been run against this database:
(hasClearanceFitWith ?holeAF ?shaftAF) (hasAllowanceWith ?holeAF ?shaftAF ?allowance)
Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

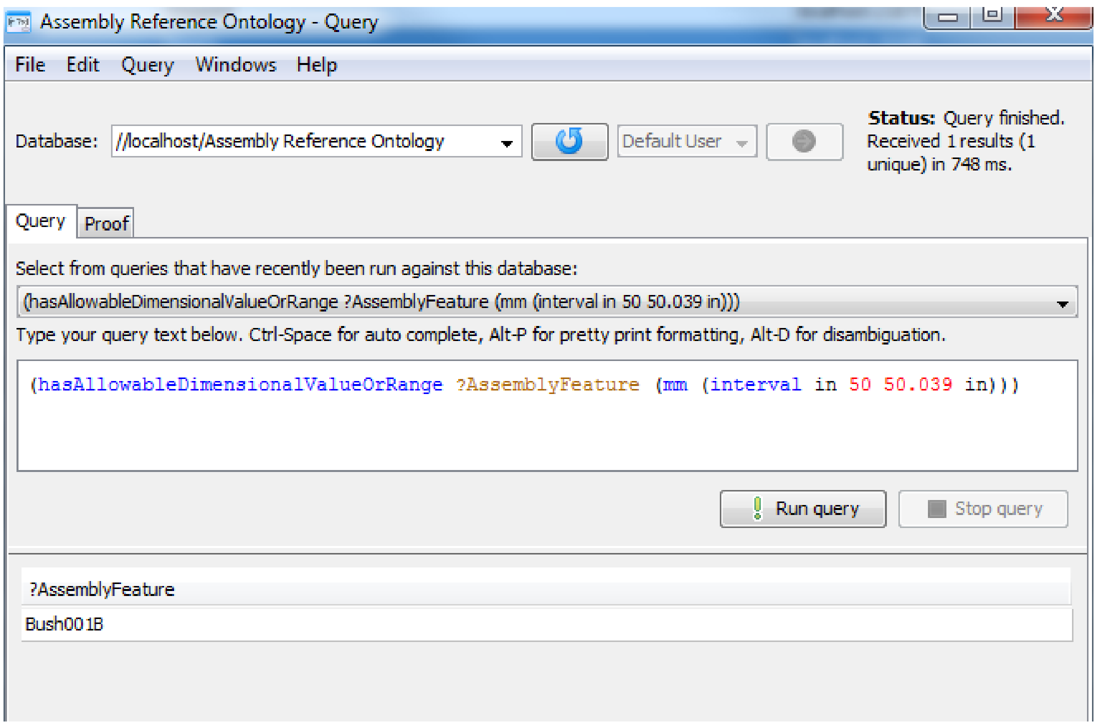
```
(hasClearanceFitWith ?holeAF ?shaftAF)
(hasAllowanceWith ?holeAF ?shaftAF ?allowance)
```

?holeAF	?shaftAF	?allowance
Bush001B	Shaft01	(mm 0.025)

Figure 6.12: Queries to find out fits and allowance related information

Figure 6.12 also shows queries to find out the allowance between the mating assembly features. The allowance displayed as negative represents the maximum interference between BearingHousing01 and Bush001A whereas the allowance displayed as positive is the minimum clearance between Bush001B and the Shaft01.

The query shown in figure 6.13 aims to find out the assembly feature whose dimensions are acceptable within a range of values. Specifically the query in figure 6.13 asks for the assembly features having dimension range in between 50 mm and 50.039 mm (both numbers included). This query displays bush feature B which satisfies the query as can be seen from figure 6.11 as well.



Database: //localhost/Assembly Reference Ontology

Status: Query finished. Received 1 results (1 unique) in 748 ms.

Select from queries that have recently been run against this database:

(hasAllowableDimensionalValueOrRange ?AssemblyFeature (mm (interval in 50 50.039 in)))

Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(hasAllowableDimensionalValueOrRange ?AssemblyFeature (mm (interval in 50 50.039 in)))
```

```
?AssemblyFeature
Bush001B
```

Figure 6.13: Query to find out assembly features for a range of acceptable dimensional values.

6.3.3 Evaluating the Assembly Process Planning Consequences of Product Assembly during the Assembly Design Stage and Investigating a Case of Inter-Domain Assembly Knowledge Sharing

The aims of this experiment are as follows:

- To validate that the ARO supports the capture of assembly process planning view point of assembly feature concept
- To establish that the ARO can relate the assembly design knowledge with the assembly process planning knowledge using the concept of assembly feature
- To demonstrate that the assembly process planning consequences can be determined during the assembly design stage.

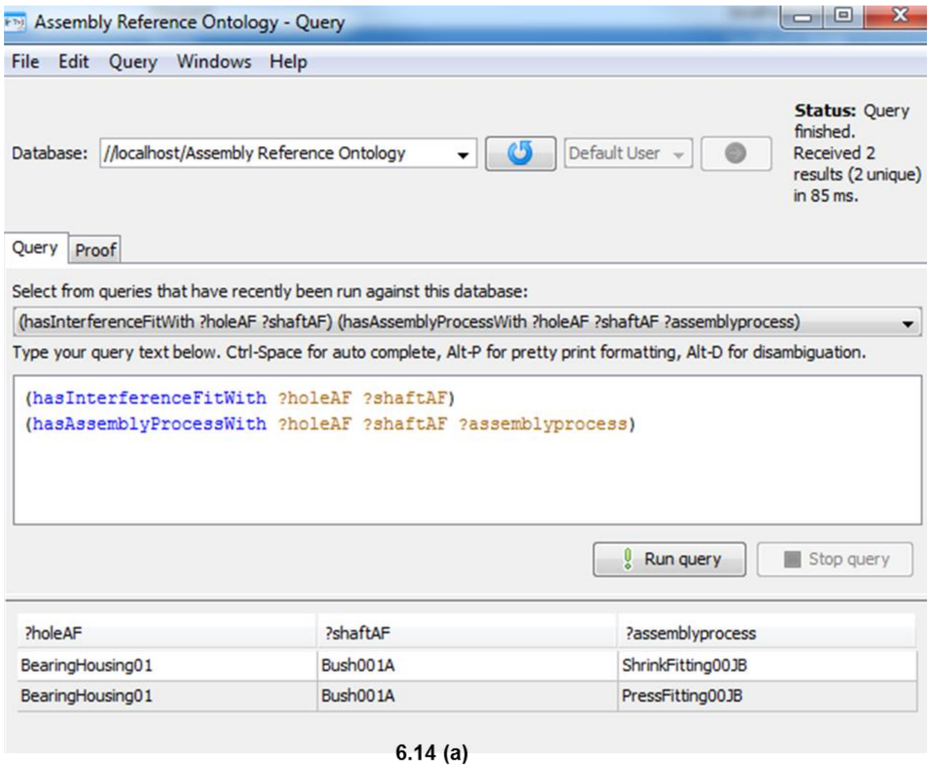
In order to validate that the ARO supports the capture of assembly process planning viewpoint of assembly features, it requires the population of instances of assembly process planning related concepts e.g. assembly processes, assembly resources and manufacturing facilities. During the formalization of assembly process planning related concepts (please refer back to section 5.3.4.2), various assembly processes were suggested depending upon the type of fit. They include press fitting, shrink fitting, manual insertion, machine assisted insertion etc. The instances of these assembly processes are shown in table 6-3. Similarly the instances of manufacturing facilities and the assembly resources are also shown in table 6-3. These facts have been successfully populated in the IODE knowledge base.

Based on the information provided in table 6-3, different queries can be made to retrieve and relate the assembly process planning knowledge with the assembly design knowledge. For example, queries can be made to find out the possible assembly processes for the mating assembly features when they have a specific type of assembly fit. Figure 6.14 shows potential assembly processes for mating assembly features which have clearance and interference fits.

Table 6-3: Assembly process planning related facts asserted for journal bearing assembly

Classes and Relations	Instances
PressFitting	PressFitting00JB
ShrinkFitting	ShrinkFitting00JB
ManualInsertion	ManualInsertion00JB
MachineAssistedInsertion	MachineAssistedInsertion00JB
ManufacturingFacility	ManufacturingFacility001 ManufacturingFacility002 ManufacturingFacility003
Furnace	Furnace001 Furnace002
Operator	Operator00A1 Operator00B1
Robot	Robot00A3 Robot00B3
hasAssemblyResource	ManufacturingFacility001 PressFitMachine001 ManufacturingFacility001 Torch001 ManufacturingFacility001 Furnace001 ManufacturingFacility001 Operator00A1 ManufacturingFacility001 Operator00B1 ManufacturingFacility002 PressFitMachine002 ManufacturingFacility002 Torch002 ManufacturingFacility002 Furnace002 ManufacturingFacility003 Robot00A3 ManufacturingFacility003 Robot00B3

As the hole assembly feature Bearinghousing01 has interference fit with the shaft assembly feature Bush001A, hence the recommended assembly processes for these mating features are shrink fitting and press fitting as shown in figure 6.14 (a). Similarly the hole assembly feature “Bush001B” has clearance fit with the shaft assembly feature “Shaft01” therefore the possible assembly processes for these assembly features could be manual insertion and machine assisted insertion as shown in figure 6.14 (b).


Assembly Reference Ontology - Query
File Edit Query Windows Help
Database: //localhost/Assembly Reference Ontology
Default User
Query
Proof
Run query
Stop query

Status: Query finished. Received 2 results (2 unique) in 85 ms.

Select from queries that have recently been run against this database:

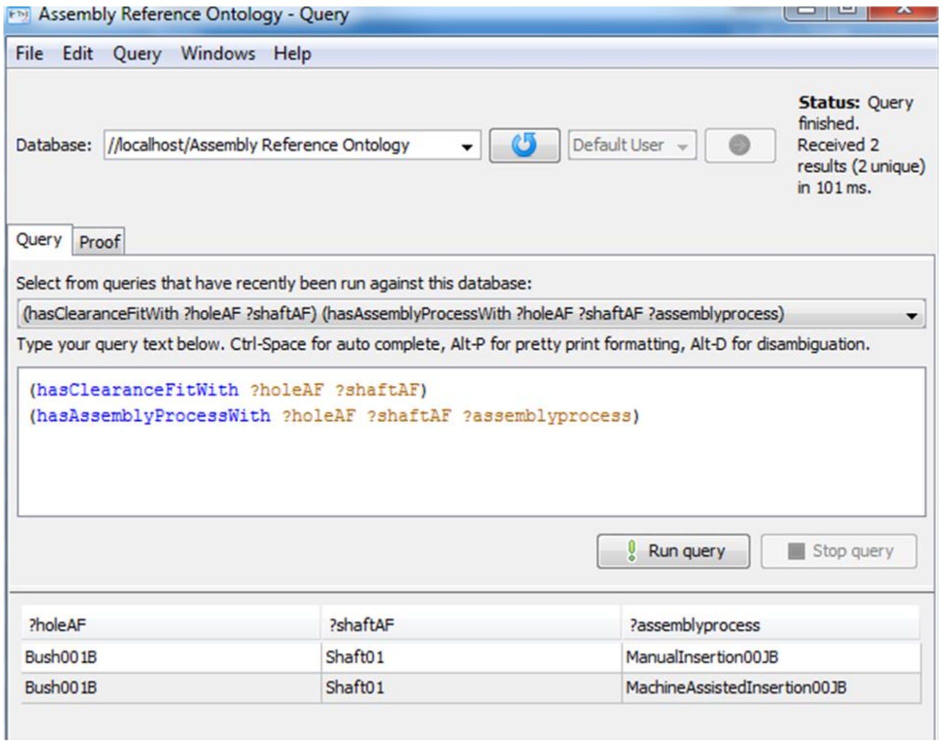
(hasInterferenceFitWith ?holeAF ?shaftAF) (hasAssemblyProcessWith ?holeAF ?shaftAF ?assemblyprocess)

Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(hasInterferenceFitWith ?holeAF ?shaftAF)
(hasAssemblyProcessWith ?holeAF ?shaftAF ?assemblyprocess)
```

?holeAF	?shaftAF	?assemblyprocess
BearingHousing01	Bush001A	ShrinkFitting00JB
BearingHousing01	Bush001A	PressFitting00JB

6.14 (a)


Assembly Reference Ontology - Query
File Edit Query Windows Help
Database: //localhost/Assembly Reference Ontology
Default User
Query
Proof
Run query
Stop query

Status: Query finished. Received 2 results (2 unique) in 101 ms.

Select from queries that have recently been run against this database:

(hasClearanceFitWith ?holeAF ?shaftAF) (hasAssemblyProcessWith ?holeAF ?shaftAF ?assemblyprocess)

Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(hasClearanceFitWith ?holeAF ?shaftAF)
(hasAssemblyProcessWith ?holeAF ?shaftAF ?assemblyprocess)
```

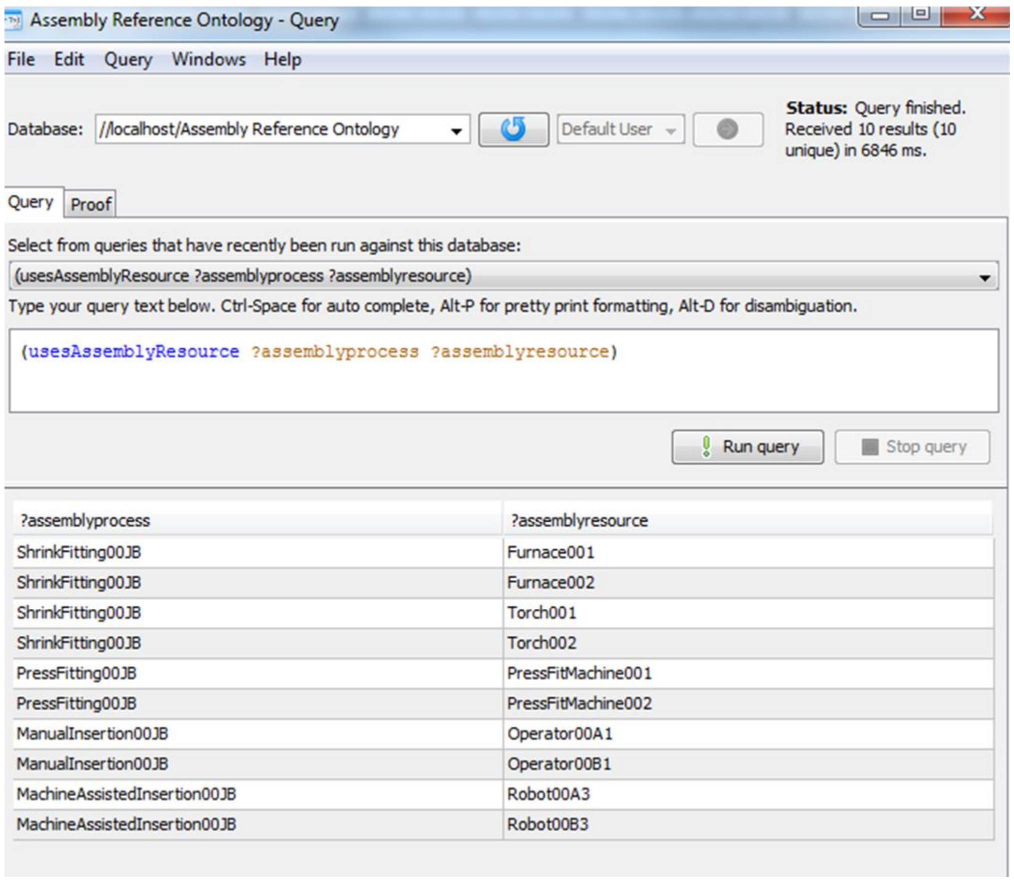
?holeAF	?shaftAF	?assemblyprocess
Bush001B	Shaft01	ManualInsertion00JB
Bush001B	Shaft01	MachineAssistedInsertion00JB

6.14 (b)

Figure 6.14: Queries to find out the assembly processes based on the type of assembly fits for the mating assembly features.

It is important to understand that the assembly design has an impact on assembly process planning domain. This is shown in the example presented above where the type of fit has an influence on the selection of assembly processes. This leads to the fact that the assemblability issues can be assessed during the design stage of the product and possible decisions can be taken to prevent the waste of time and money.

Another aspect of this experiment is the evaluation of assembly resources for the suggested assembly processes. A query can be made to find out the possible assembly resources against the suggested assembly processes. This is shown in figure 6.15. For example, the assembly resources Furnace001, Furnace002, Torch001, and Torch002 can be used for the assembly process ShrinkFitting00JB. Similarly PressFitMachine001 and PressFitMachine002 can be used for PressFitting00JB and so on.


Assembly Reference Ontology - Query
File Edit Query Windows Help
Default User
Query
Proof
Run query
Stop query

Database: //localhost/Assembly Reference Ontology

Status: Query finished. Received 10 results (10 unique) in 6846 ms.

Select from queries that have recently been run against this database:
(usesAssemblyResource ?assemblyprocess ?assemblyresource)

Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

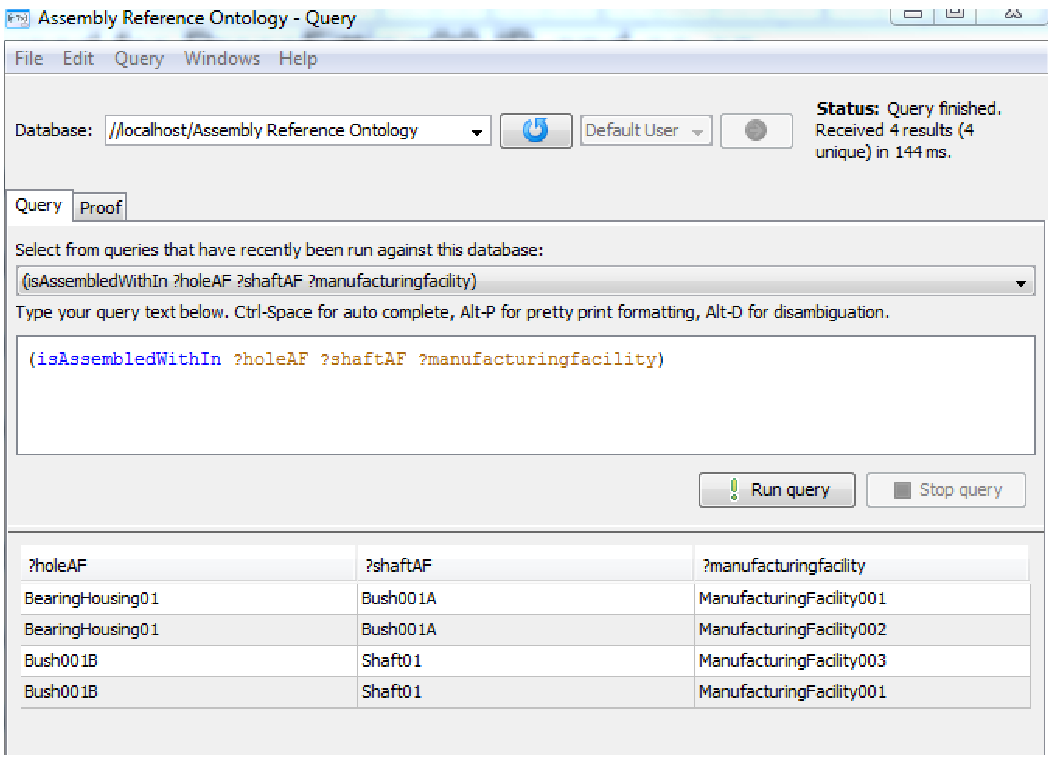
```
(usesAssemblyResource ?assemblyprocess ?assemblyresource)
```

?assemblyprocess	?assemblyresource
ShrinkFitting00JB	Furnace001
ShrinkFitting00JB	Furnace002
ShrinkFitting00JB	Torch001
ShrinkFitting00JB	Torch002
PressFitting00JB	PressFitMachine001
PressFitting00JB	PressFitMachine002
ManualInsertion00JB	Operator00A1
ManualInsertion00JB	Operator00B1
MachineAssistedInsertion00JB	Robot00A3
MachineAssistedInsertion00JB	Robot00B3

Figure 6.15: Query to find out the available assembly resources for the assembly processes

The query shown in figure 6.15 can also be linked with the queries shown in figure 6.14. These queries can be combined to find out the available assembly resources for a specific type of assembly fit or for specific hole and shaft assembly features.

It is also worth knowing that which manufacturing facility can carry out the assembly of hole and shaft assembly features. The query shown in figure 6.16 displays the requested information for the hole and shaft assembly features. This query can also be combined with the queries shown in figure 6.14 to find out specific information as required by the user.


Assembly Reference Ontology - Query
File Edit Query Windows Help
Default User
Query
Proof
Run query
Stop query

Database: //localhost/Assembly Reference Ontology

Status: Query finished. Received 4 results (4 unique) in 144 ms.

Select from queries that have recently been run against this database:
(isAssembledWithIn ?holeAF ?shaftAF ?manufacturingfacility)

Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(isAssembledWithIn ?holeAF ?shaftAF ?manufacturingfacility)
```

?holeAF	?shaftAF	?manufacturingfacility
BearingHousing01	Bush001A	ManufacturingFacility001
BearingHousing01	Bush001A	ManufacturingFacility002
Bush001B	Shaft01	ManufacturingFacility003
Bush001B	Shaft01	ManufacturingFacility001

Figure 6.16: Query to find out the available manufacturing facility for the assembly of hole and shaft assembly features

The query shown in figure 6.16 returns back the results which suggest that BearingHousing01 and Bush001A can be assembled in ManufacturingFacility001and ManufacturingFacility002. This is because

ManufacturingFacility003 does not have the assembly resources available to carry out the assembly processes based on interference fit e.g. shrink fitting and press fitting. In contrast to the ManufacturingFacility003, the instances of other two manufacturing facilities have got resources like press fit machine, torch, and furnace and therefore they can be used as manufacturing facilities for the assembly features: BearingHousing01 and Bush001A.

Similarly, Bush001B and Shaft01 (these two assembly features have clearance fit with each other) can be assembled in ManufacturingFacility001 and ManufacturingFacility003. This is because the ManufacturingFacility002 does not have assembly resources to perform assembly processes based on clearance fit e.g. manual insertion or machine assisted insertion.

This experiment has demonstrated that the assembly process planning viewpoint of assembly concept can be captured using the ARO. The results have also suggested that the assembly design knowledge can be related with assembly process planning knowledge using the concept of assembly feature. It is also important to mention that the decisions taken during the assembly design phase have an impact on the assembly process planning stage. It is argued that the ARO can be used to support the decision making during the early stages of assembly design.

6.4 Case Study

6.4.1 Introduction

This section explores a case study in the automotive sector to demonstrate that the proposed ARO framework can work for areas other than discussed in chapter 5. The case study considers engine assembly as a test case and explores the product design change effects onto the assembly process planning domain. The case study discusses two different engines to be assembled using the same set of resources and evaluates whether those existing assembly resources can be used for the new engine.

The results of the case study validate the applicability of the ARO in a wider scope within the assembly domain. A detailed description of the case study scenario is exposed in section 6.4.2. Section 6.4.3 describes the formal representation of the key concepts and relationships used. Section 6.4.5 explains the experimental evaluation of the assembly resources based on the formalization of concepts and relationships. Finally section 6.4.5 presents the concluding case study remarks.

6.4.2 Overview of The Case Study Scenario

6.4.2.1 The Engine Assembly Line

This case study takes into account a real engine assembly line scenario in an automotive industry. At present the assembly line is used for a three cylinder engine and has around 150 assembly stations. Various assembly operations are carried out on each station until the engine is unmounted from the assembly line. The assembly line has been originally designed to accommodate a range of products. However it is important to evaluate the capability of the assembly resources associated with the assembly line in order to assemble the new product or a modified product during the product design stage. The earlier assessment of relevant assembly resources is necessary to support the decisions which may affect the assembly cost and time. For instance, if the new or modified engine design is passed on to the assembly line without resource evaluation, it may require redesign of the assembly resources and/or the engine itself.

The evaluation of the assembly resources can be carried out by human experts working on the assembly line or by using the simulation based applications. However as the engine assembly comprises of multiple assembly operations and these assembly operations have multiple steps, therefore resource evaluation becomes a cumbersome job for a human expert. Even if the resource evaluation is carried out via simulation, it becomes a time consuming task.

This work employs the use of formal ontologies to evaluate the assembly resources for a new or modified product on the existing assembly line. The underlying structure of the ontology is based on the ARO proposed in chapter 3 and it uses most of the concepts and relationships introduced in chapter 4 and chapter 5. The resource evaluation has been carried out by analysing the mating resource and product features. The analysis exploits the ARO and some additional case study specific axioms which have been discussed in section 6.4.4.

As discussed above, the assembly line has around 150 work stations where different assembly operations can be performed however the scope of this case is limited to one operation. The assembly operation considered for this case study is called OP60 and is discussed in the next section. The author believes that resource evaluation on other operations can be carried out on similar lines as for OP60.

6.4.2.2 OP60 Station

OP60 also called the engine mounting operation is the very first assembly operation on the engine assembly line. OP60 is partially manual and partially automatic assembly operation. A visual representation of the main elements of OP60 is shown in figure 6.17. Various kinds of assembly resources (as shown in the figure) which have been used in the OP60 are: crane, operator, dowel gun, engine block rack, turn table, pallet, conveyor, and the engine block load machine.

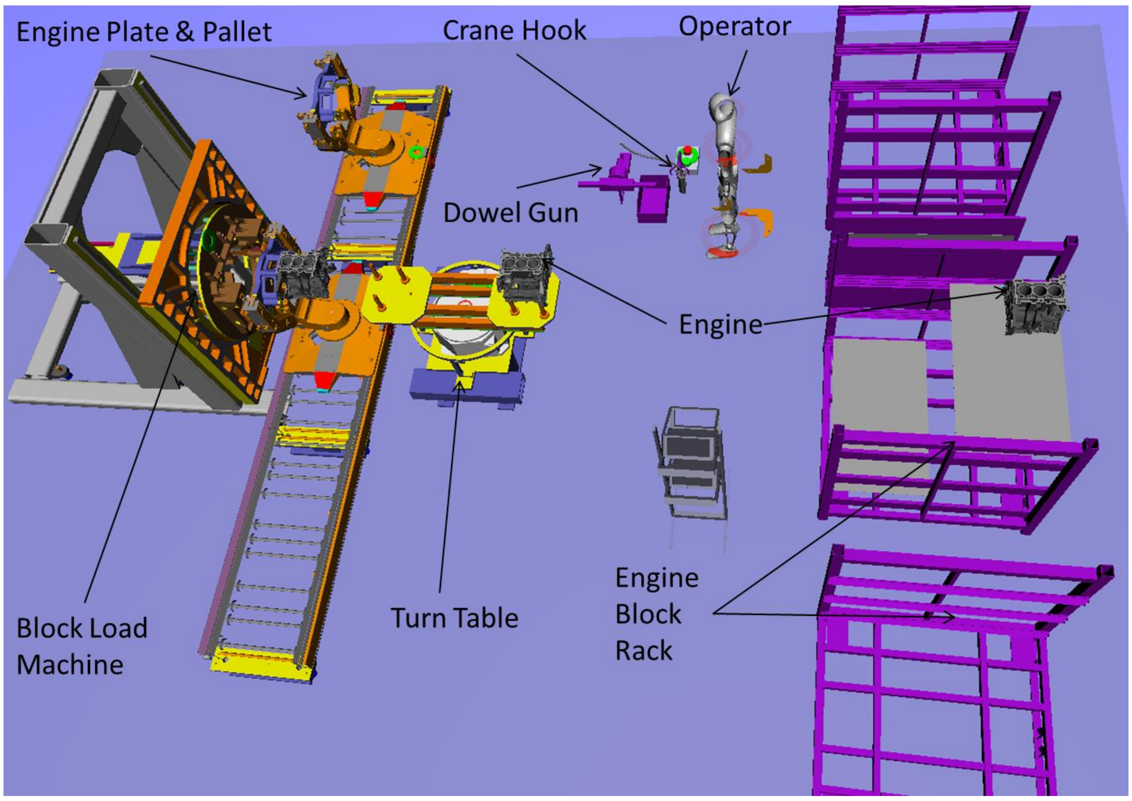


Figure 6.17: A visual overview of the main elements of OP60

The human operator performs all types of manual activities which include operating the crane, moving the engines from the rack, and installing the dowel pins in the engine. The racks are used to place engine blocks whereas the dowel gun is used to install dowel pins in the engine. Rotary table is used to transport the engine when the pallet comes in front of the block load machine. The conveyor continuously moves and has pallets installed on it as shown in figure 6.17. Engine blocks are loaded at OP60 on the pallets and a complete engine assembly is unmounted at the last station on the assembly line. The empty pallet at the last station appears again at OP60 for engine block mounting and the cycle continues. The engine block load machine clamps the engine side plate and aligns it with the engine block holes. The engine block machine has a set of four nut runners which are used to fasten the bolts in order to assemble side plate with the engine block.

Keeping in view the definition of the step in section 4.3.16, OP60 has been categorized into four steps which are as follows:

Step1: Move the crane to grasp the engine on the rack

Step2: Move and place the engine on turn table

Step3: Install the dowel pins into the engine

Step4: Move, align, and assemble the engine with the engine plate

The visual illustration of these steps is shown in figure 6.18. In step 1 of OP60, the operator moves the crane hook from its default position (shown in figure 6.17) and grasps the engine as shown in 6.18. In the second step, the operator lifts the engine with crane and moves it towards the turn table. The turn table has three locating pins where the engine is placed. The next step is the installation of dowel pins into the engine while it rests on the locating pins of the turn table as shown in the step 3 of figure 6.18. During step 3, the operator reaches the dowel gun (shown in figure 6.17), grasps it, and moves it towards the engine placed on turn table. At this point, the operator installs the dowel pins into the engine as shown in figure 6.18.

In the fourth and the last step, the engine is transported from turn table position shown at step 2 and 3 to engine position at step 4. Here at first the engine side plate is aligned with engine and then engine is fastened with side plate by the nut runners. Once they are assembled the pallet moves on and the next cycle starts.

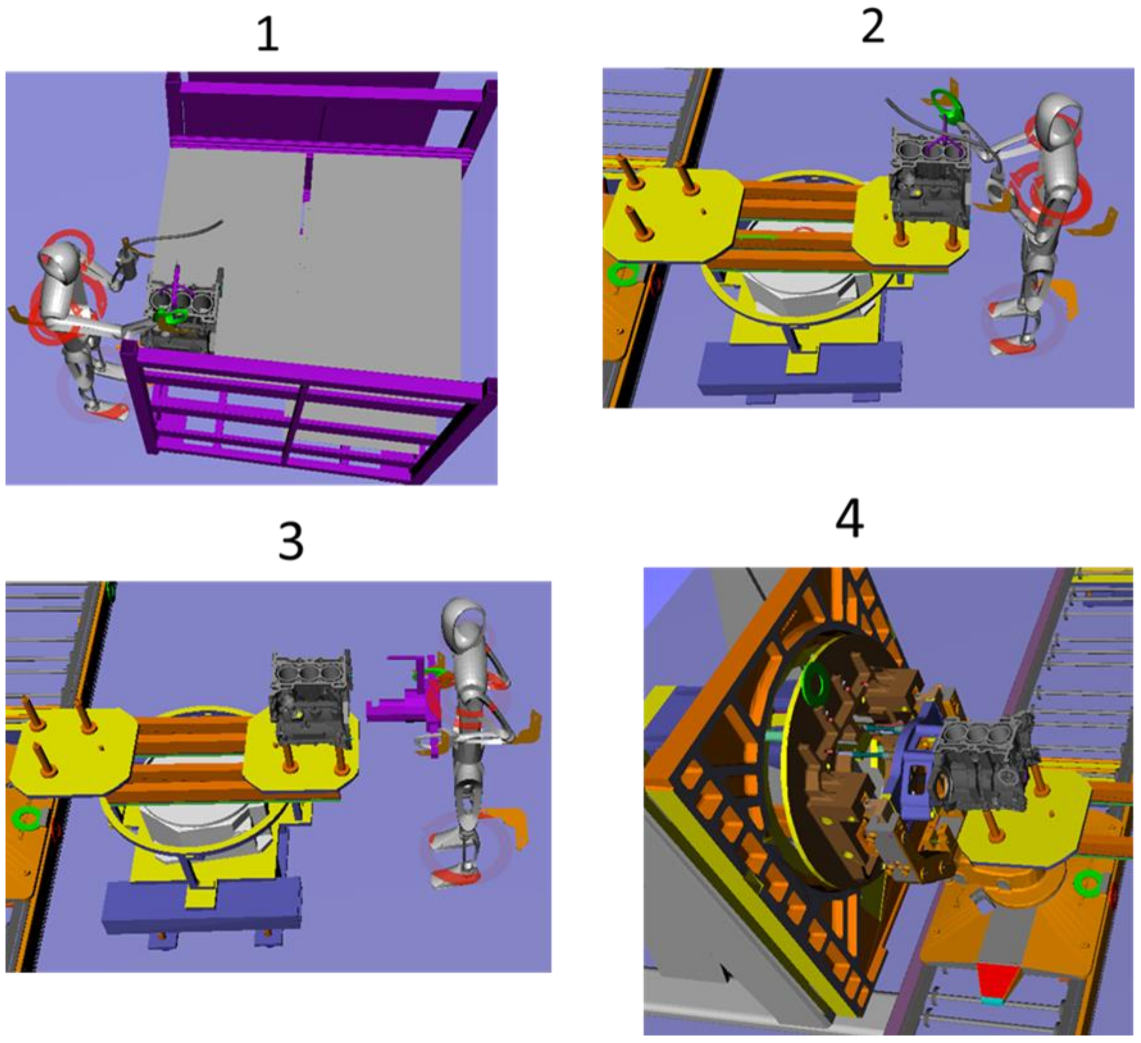


Figure 6.18: Steps involved in OP60 assembly operation

6.4.2.3 Products

The automotive industry under study is planning to use a mixed mode manufacturing strategy where two different types of products have been planned to run on the same assembly line simultaneously. The existing product on the assembly line is a three cylinder engine whereas the new engine planned to be assembled in the assembly line is a four cylinder engine. The two products are shown in the figure 6.19.

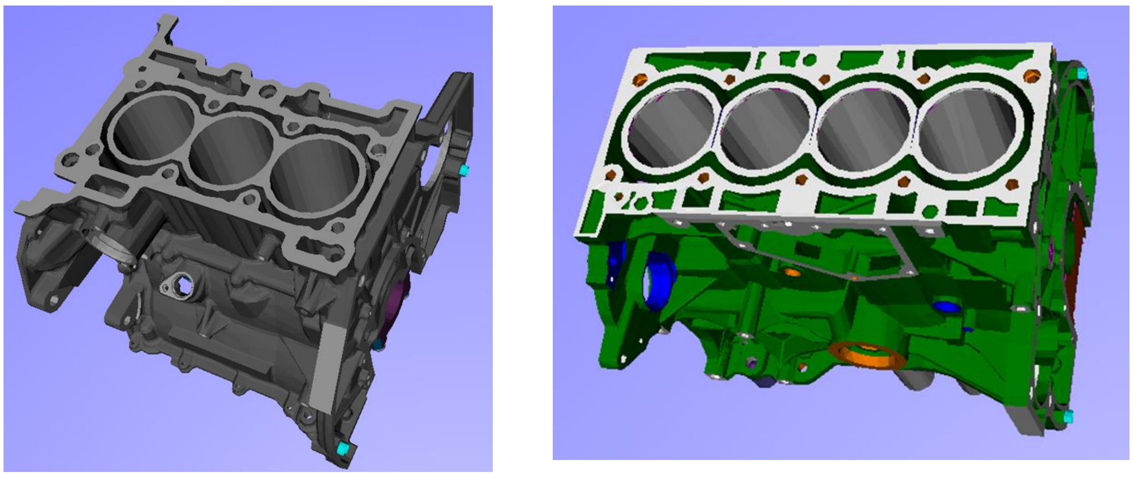


Figure 6.19: A view of the 3 cylinder engine (existing product) and 4 cylinder engine (new product)

As discussed earlier, the new product should be assessed against the existing assembly resources before it is actually brought in for assembly. This work employs the use of product features and resource features to evaluate the resource capability to handle multiple products. Product and resource features and their specializations have already been discussed in section 4.3.12 and section 4.3.19 in chapter 4.

The feature related concepts used in this case study are: assembly feature, and assembly resource feature. The specialized classes of assembly features used for this case study are: hole assembly feature, shaft assembly feature, and handling assembly feature. The specialized classes of assembly resource feature used for this case study are handling resource assembly feature, hole resource assembly feature and shaft resource assembly feature. The next section explains the product change effects onto the assembly process planning domains using the feature related concepts and other ARO concepts.

6.4.2.4 Product Change Effects

Design changes in the product can significantly affect the subsequent product lifecycle stages. It is important to understand that these effects should be taken into account during the early stages of product. In the context of this case study, the change of product can have potential issues during the assembly process planning domain. To evaluate these effects, the author has considered step 2 of OP60 for detail analysis.

As discussed in section 6.4.2.2, during the step 2 of OP60 the engine is placed on the locating pins of the turn table. As the turn table is used for handling of engines in step 2 therefore the assembly resource features used during step 2 are handling assembly resource features as shown in figure 6.20.

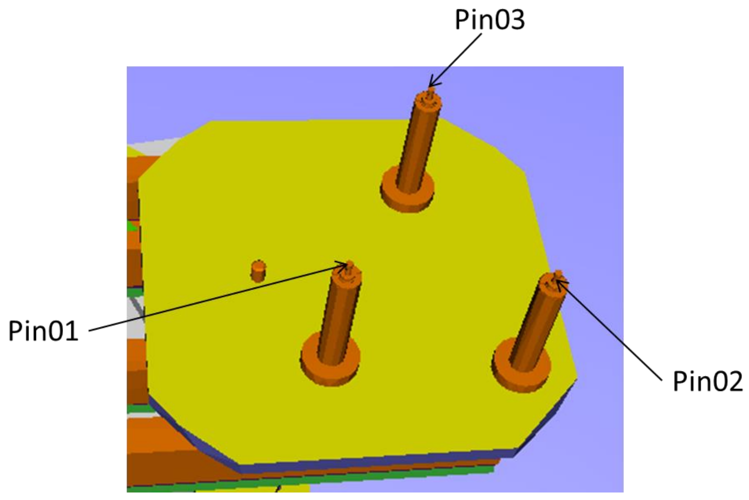
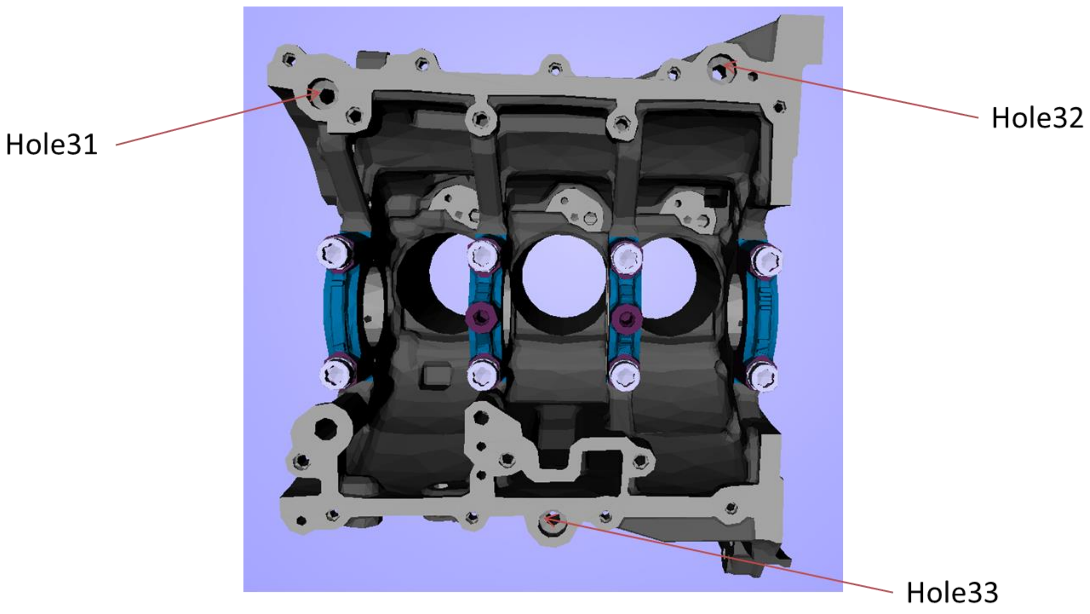
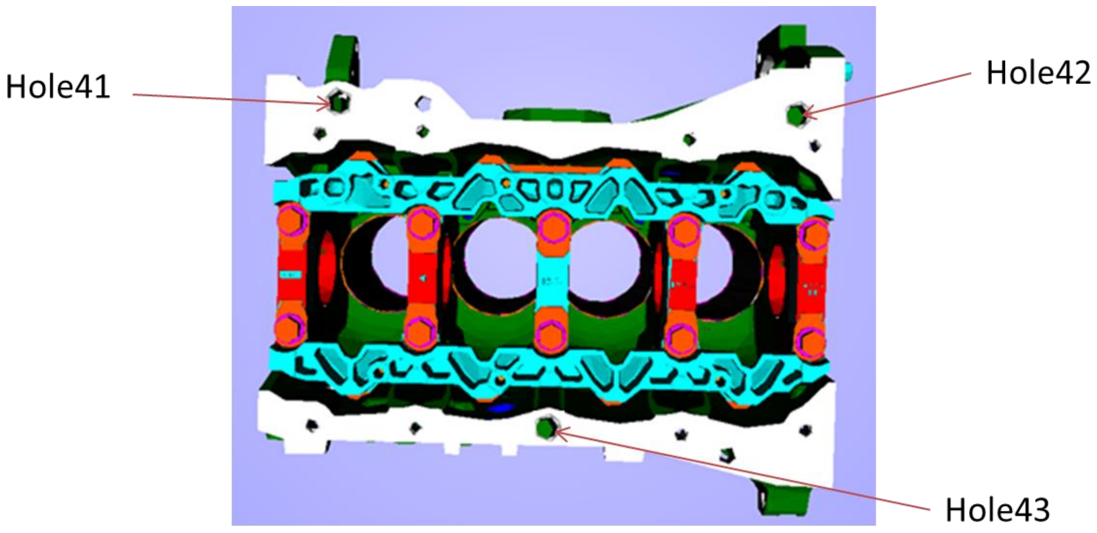


Figure 6.20: Turn table handling assembly resource features (locating pins)

The corresponding handling assembly features on three cylinder and four cylinder engine for step 2 of OP60 are shown in figure 6.21. These product assembly features mate with the assembly resource features shown in figure 6.20 during step 2 of OP60. In other words, Hole31 and Hole41 should mate Pin01, Hole32 and Hole42 should mate with Pin02, whereas Hole33 and Hole43 should mate Pin03.



a. Handling features on 3 cylinder engine



b. Handling features on 4 cylinder engine

Figure 6.21: Handling assembly features on 3 cylinder and 4 cylinder engines for step 2 of OP60

As the 3 cylinder engine is already running on the assembly line and is using the assembly resources at OP60 workstation, it is understood that the assembly resources at OP60 station are appropriate for 3 cylinder engine. However when a new product i.e. 4 cylinder engine is brought in for the same assembly line, it is necessary to evaluate the assembly resources for the new product.

For the scenario discussed above, any change in the design of handling assembly features on the product may directly affect the suitability of the assembly resource for that product. Hence it is important that the effect of such changes should be considered when they are required to be made. For the handling assembly features shown in figure 6.21, and handling resource assembly features shown in figure 6.20, there are three different type of checks which should be made in order to assess the suitability of an assembly resource i.e. turn table for the specific product i.e. 3 cylinder and 4 cylinder engines. These three checks are as follows:

1. Quantity of the mating assembly features on product and resource should be same
2. Size of the mating assembly features on product and resource should be appropriate in order to support their mating.
3. Spatial location of the mating assembly features on product and resource should be same at the point of mating.

Once the assembly resource has passed these three checks, it can be used for the required product. For example, the turn table shown in figure 6.20 has three mating assembly features i.e. Pin01, Pin02, Pin03. These assembly features mate with three assembly features on existing product (3 cylinder engine) which are: Hole31, Hole32, and Hole33. As the number of handling assembly features is equal to the number of handling resource assembly features for the step 2 of OP60, therefore it meets the first condition. The second check asks for size compatibility of the mating assembly features and assembly resource features.

As for hole and shaft specific temporary assembly cases, it is required that these features should have a clearance fit with each other. Once this condition is met, the third check is about the spatial location of assembly and assembly resource features. As discussed in section 4.3.5 of chapter 4, a spatial location can be specified as point location and angular location. The point and angular spatial location of the mating assembly features and assembly resource features should be same. In the present case (of the existing 3 cylinder engine), the point and angular locations of assembly features and assembly resource features are same.

6.4.3 Formal Representation of Case Study Scenario

The ARO concepts explained in chapter 4 are being explored further to support the formal representation of the case study scenario. However in addition to the ARO relationships and axioms described in chapter 4, and 5, other supplementary relationships and axioms have been introduced to capture the case study specific knowledge. As discussed in section 6.4.2.4, an assembly resource can be used for a product if it meets three conditions. These conditions can be used to deduce whether an assembly resource can be used or not and this can be specified in terms of the following axiom.

```
(<= (canBeUsedAsAssemblyResourceFor ?AssemblyResource ?Engine)

      (and (AssemblyResource ?AssemblyResource)

      (Product ?Engine)

      (hasEqualNumOfFeatureUsedWith ?AssemblyResource ?Engine)

      (hasFeatureLocationMateableWith ?AssemblyResource ?Engine)

      (hasFeatureSizeMateableWith ?AssemblyResource ?Engine)))


      :rem "An assembly resource can be used a product if it has same
      number of assembly features, mateable location and mateable
      feature size with that of a product."
```

All relationships specified in the above axiom have been defined specifically for this case study as they were not defined in the ARO. Three conditions specified in the last three directives of above axiom have been deducted from other axioms which are explained in the following sections.

6.4.3.1 Feature Quantity Condition

As the first condition states that assembly resource should have equal number of features with the product being assembled (engine in present case) so the following rule has been applied to deduce this condition.

```
(<= (hasEqualNumOfFeatureUsedWith ?AssemblyResource ?Engine)

      (and (AssemblyResource ?AssemblyResource)

      (Product ?Engine)

      (hasNumOfHandlingAFat ?Engine ?HAFQuantity ?step)

      (Step ?step)

      (hasNumOfHandlingARFat ?AssemblyResource ?HARFQuantity ?step)

      (= ?HAFQuantity ?HARFQuantity)

      (gtNum ?HAFQuantity 0)

      (gtNum ?HARFQuantity 0)))

:rem "An assembly resource has equal number of handling assembly
resource features at a particular step of an operation if at the same
step product has same number of handling assembly features except
zero."
```

The above axiom states that an assembly resource can have equal number of features with the product if that assembly resource has handling resource assembly feature quantity used at a particular step equal to the product handling assembly features used at the same step. The concept of step has been used because a product or assembly resource can have multiple assembly features that may be used in other steps or assembly operations.

The last two conditions in the above mentioned axiom state that the product or assembly resource handling feature quantity should always be greater than zero. These conditions have been added because in certain circumstances there may be a possibility that only tooling features are used at a particular step of an assembly operation. In that case, both product and assembly resource would have zero number of handling features. The above axiom is only applicable for product and resource handling features however similar axioms can be applied for tooling features.

Inference rules have been applied to count the asserted handling features. This enables the knowledge base to automatically count the features quantity where the latter has already been used in the previous axioms. Following two axioms have been used to count product handling assembly features.

```
 (<= (hasNumOfHandlingAF ?Engine ?HAFQuantity)

          (and (Product ?Engine)

            (countf (?HAF)

            (hasHandlingAF ?Engine ?HAF) ?HAFQuantity)))

:rem "A product has handling assembly feature quantity equal to the
number of asserted handling assembly features for that product."
```

```
(<= (hasNumOfHandlingAFat ?Engine ?HAFQuantity ?step)

       (and (Product ?Engine)

         (Step ?step)

         (countf (?HAF)

      (hasHandlingAFusedAt ?Engine ?HAF ?step) ?HAFQuantity)))

:rem "A product has handling assembly feature quantity at a particular
step of an operation equal to the number of asserted handling assembly
features used at that step."
```

The first axiom infers the total number of asserted handling assembly features for a specific product i.e. 3 cylinder engine, 4 cylinder engine. However the second axiom counts the total number of asserted handling assembly features at a specific step of an assembly operation e.g. step 2 of OP60. Similar axioms have been applied to count handling resource assembly features.

Now to identify the handling features used at a specific step (a condition specified in the above mentioned axiom), the following inference rule has been applied.

```
  (<= (hasHandlingAFusedAt ?Engine ?HAF ?step)

      (and (Product ?Engine)

             (HandlingAF ?HAF)

             (Step ?step)

             (hasHandlingAF ?Engine ?HAF)

             (usesHandlingAF ?step ?HAF)

             (hasAssemblyOperation ?Engine ?AssemblyOperation)

             (hasStep ?AssemblyOperation ?step)))

:rem "A product has handling assembly features used at a particular
step of an operation if that step is part of the assembly operation
and it uses those handling features."
```

The above axiom states that a product i.e. engine has handling assembly features used at a specific step if that product has that handling assembly feature. The other conditions are: the step should use the handling assembly feature, assembly operation should have the step, and product should have the assembly operation attached with it.

Similar rules have been applied to deduce this kind of information handling assembly resource features. Please refer back to appendix B.4 for more detail.

6.4.3.2 Feature Location Condition

The second condition for the evaluation of the assembly resource for the new product is the feature location. As discussed in section 6.4.2.4, the spatial location of the product and resource features should be same at the point of mating. The following axiom has been applied to deduce whether an assembly resource has a feature location mateable with that of a product.

```
(<= (hasFeatureLocationMateableWith ?AssemblyResource ?Engine)
      (and (AssemblyResource ?AssemblyResource)
      (Product ?Engine)
      (hasAssemblyOperation ?Engine ?AssemblyOperation)
      (AssemblyOperation ?AssemblyOperation)
      (usesAssemblyResource ?AssemblyOperation ?AssemblyResource)
      (AssemblyOperation ?AssemblyOperation)
      (not (exists (?HAF ?pointlocation1 ?angularlocation1)
      (and (hasHandlingAF ?Engine ?HAF)
      (HandlingAF ?HAF)
      (hasPointLocation  ?HAF ?pointlocation1)
      (hasAngularLocation  ?HAF ?angularlocation1)
      (not (exists (?HARF ?pointlocation2 ?angularlocation2)
      (and (hasHandlingARF ?AssemblyResource ?HARF)
      (HandlingARF ?HARF)
      (hasPointLocation  ?HARF ?pointlocation2)

      (hasAngularLocation  ?HARF ?angularlocation2)

      (= ?pointlocation1 ?pointlocation2)

      (= ?angularlocation1 ?angularlocation2)

      (matesWith ?HAF ?HARF)))))))))

 :rem "An assembly resource has features location mateable with a product if location of handling assembly features on that product are same as that of handling resource assembly features on that assembly resource."
```

The above axiom infers that an assembly resource and a product can have a feature location mateable with each other if the handling assembly resource feature and handling assembly feature have the same point and angular locations. The information regarding the point and angular locations could be obtained from respective geometric modelling software systems, if implemented.

6.4.3.3 Feature Size Condition

The mateable size condition may vary for different types of assembly features. However as the focus of this case study is on hole and shaft type of assembly features and, the product and resource need temporary mating, therefore they should have a clearance fit with each other. The following axiom specifically deduces the necessary information.

```
(<= (hasFeatureSizeMateableWith ?AssemblyResource ?Engine)

      (and (AssemblyResource ?AssemblyResource)

       (Product ?Engine)

       (hasAssemblyOperation ?Engine ?AssemblyOperation)
       (AssemblyOperation ?AssemblyOperation)

       (usesAssemblyResource ?AssemblyOperation ?AssemblyResource)

       (AssemblyOperation ?AssemblyOperation)

       (hasAssemblyFeature ?Engine ?holeAF)

       (HoleAF ?holeAF)

       (HandlingAF ?holeAF)

       (hasAssemblyResourceFeature ?AssemblyResource ?shaftARF)

       (ShaftARF ?shaftARF)

       (HandlingARF ?shaftARF)

       (hasShapeAttribute ?holeAF ?circular)

       (Circular ?circular)

       (hasShapeAttribute ?shaftARF ?circular)

       (hasClearanceFitWith ?holeAF ?shaftARF)))

:rem "An assembly resource has feature size mateable with a product if
the handling feature has clearance fit with handling resource feature
of an assembly resource."
```

The above rule dictates that an assembly resource and a product can have feature size compatibility if these features have clearance fit with each other. As discussed in chapter 5, the hasClearanceFitWith relation is held between a hole assembly feature and a shaft assembly feature (subsumptions of product features), whereas resource features were not accommodated. As this work considers the resource and product features for size compatibility therefore the axioms related to fits related information need to change.

For instance, the updated axiom for a clearance fit between a resource feature and a product feature, and vice versa is given as follows:

```
(<= (hasClearanceFitWith ?h ?s)
                  (and (or (HoleAF ?h)
                           (HoleARF ?h))
                        (hasShapeAttribute ?h ?circular)
                        (Circular ?circular)
                        (or (ShaftAF ?s)
                            (ShaftARF ?s))
                        (hasShapeAttribute ?s ?circular)
                        (or (matesWith ?h ?s)
                              (and (hasDimension ?h ?d1)
                                   (hasDimension ?s ?d2)
                                   (= ?d1 ?d2)
                                   (matesWith ?h ?s)
                                   (hasToleranceType ?h ?tolH8)
                                   (hasToleranceType ?s ?tolf7)
                                   (tolTypeH8 ?tolH8)
                                   (tolTypef7 ?tolf7)))
                        (hasMinAllowableDimension ?h ?holeMinDim)
                        (hasMaxAllowableDimension ?s ?shaftMaxDim)
                        (measureLT ?shaftMaxDim ?holeMinDim)))
```

```
:rem "A hole has clearnce Fit With a shaft if the minimum allowable
dimension of hole is larger than the maximum allowable dimension of
shaft."
```

The above axiom incorporates assembly resource hole and shaft features along with the product hole and shaft assembly features. The axiom add an operator “or” to allow the rule to have clearance fit with: a hole assembly feature and a shaft assembly feature, a hole assembly resource feature and a shaft assembly feature and vice versa. The other conditions for hasClearnceFitWith inference rule are the same.

6.4.4 Assembly Resource Evaluation

6.4.4.1 Populating the Product and Assembly Resource Information

Once the underlying formal ontological structure for the case study is in place, product and resource related information can be populated for the resource evaluation. Table 6-4 shows selected set of important facts asserted for turn table resource evaluation. The existing product 3CylinderEngine and the new product 4CylinderEngine are populated against the class Product. Turn table has been asserted as assembly resource for step “Step2” of assembly operation “Op60” as shown in the table 6-4. The hole features “Hole31”, “Hole32”, “Hole33” of 3CylinderEngine and “Hole41”, “Hole42”, “Hole43” of 4CylinderEngine are also instances of handling assembly features therefore they have been asserted under both the classes.

Similarly the shaft features “Pin01”, “Pin02”, “Pin03” of turn table are instances of both the shaft assembly resource class and handling assembly resource class as shown in the table.

Table 6-4: Product and assembly resource related facts asserted in the knowledge base

Classes and Relations	Instances
Product	3CylEngine 4CylEngine
AssemblyResource	TurnTable
AssemblyOperation	Op60
Step	Step2
HandlingAF, HoleAF	Hole31 Hole32 Hole33 Hole41 Hole42 Hole43
HandlingARF, ShaftARF	Pin01 Pin02 Pin03
hasHandlingAF	3CylEngine Hole31 3CylEngine Hole32 3CylEngine Hole33 4CylEngine Hole41 4CylEngine Hole42 4CylEngine Hole43
hasHandlingARF	TurnTable Pin01 TurnTable Pin02 TurnTable Pin03
matesWith	Hole31 Pin01 Hole32 Pin02 Hole33 Pin03 Hole41 Pin01 Hole42 Pin02 Hole43 Pin03
hasDimension	Hole31 (mm 20) Hole32 (mm 20) Hole33 (mm 20) Hole41 (mm 20) Hole42 (mm 20) Hole43 (mm 20) Pin01 (mm 20) Pin02 (mm 20) Pin03 (mm 20)
hasPointLocation	Hole31 (pointlocation (mm 0) (mm 0) (mm 0)) Hole32 (pointlocation (mm 210) (mm 0) (mm 0)) Hole33 (pointlocation (mm 105) (mm 238) (mm 0)) Hole41 (pointlocation (mm 0) (mm 0) (mm 0)) Hole42 (pointlocation (mm 300) (mm 0) (mm 0)) Hole43 (pointlocation (mm 141) (mm 218) (mm 0)) Pin01 (pointlocation (mm 0) (mm 0) (mm 0)) Pin02 (pointlocation (mm 210) (mm 0) (mm 0)) Pin03 (pointlocation (mm 105) (mm 238) (mm 0))
hasAngularLocation	Hole31 (angularlocation (degree 90) (degree 90) (degree 0)) Hole32 (angularlocation (degree 90) (degree 90) (degree 0)) Hole33 (angularlocation (degree 90) (degree 90) (degree 0)) Hole41 (angularlocation (degree 90) (degree 90) (degree 0)) Hole42 (angularlocation (degree 90) (degree 90) (degree 0)) Hole43 (angularlocation (degree 90) (degree 90) (degree 0)) Pin01 (angularlocation (degree 90) (degree 90) (degree 0)) Pin02 (angularlocation (degree 90) (degree 90) (degree 0)) Pin03 (angularlocation (degree 90) (degree 90) (degree 0))

As the Hole31, Hole32, Hole33 belong to 3CylEngine and Hole41, Hole42, Hole43 belong to 4CylEngine, therefore the relation hasHandlingAF has been instantiated to link the instances of handling features with those of product. Similarly the relation hasHandlingARF has been instantiated to link instances of handling resource features with those of assembly resource.

The relation matesWith has been instantiated to relate the mating product handling features with those of resource handling features. Hole31 and Hole41 mate with Pin01, Hole32 and Hole42 mate with Pin02, and, Hole33 and Hole43 mate with Pin03 as shown in table 6-4.

All the product and resource handling features shown in figure 6.20 and 6.21 have a nominal dimension of 20 mm therefore they have been linked with their dimensions with the relation hasDimension as shown in the table.

The spatial location of the product and resource handling features has been represented by the relations hasPointLocation and hasAngularLocation. A reference point has been assumed to be the same for product and resource features from which the point location of the mating features have been considered. The hasPointLocation relation links product and resource handling features with their distance from the reference point.

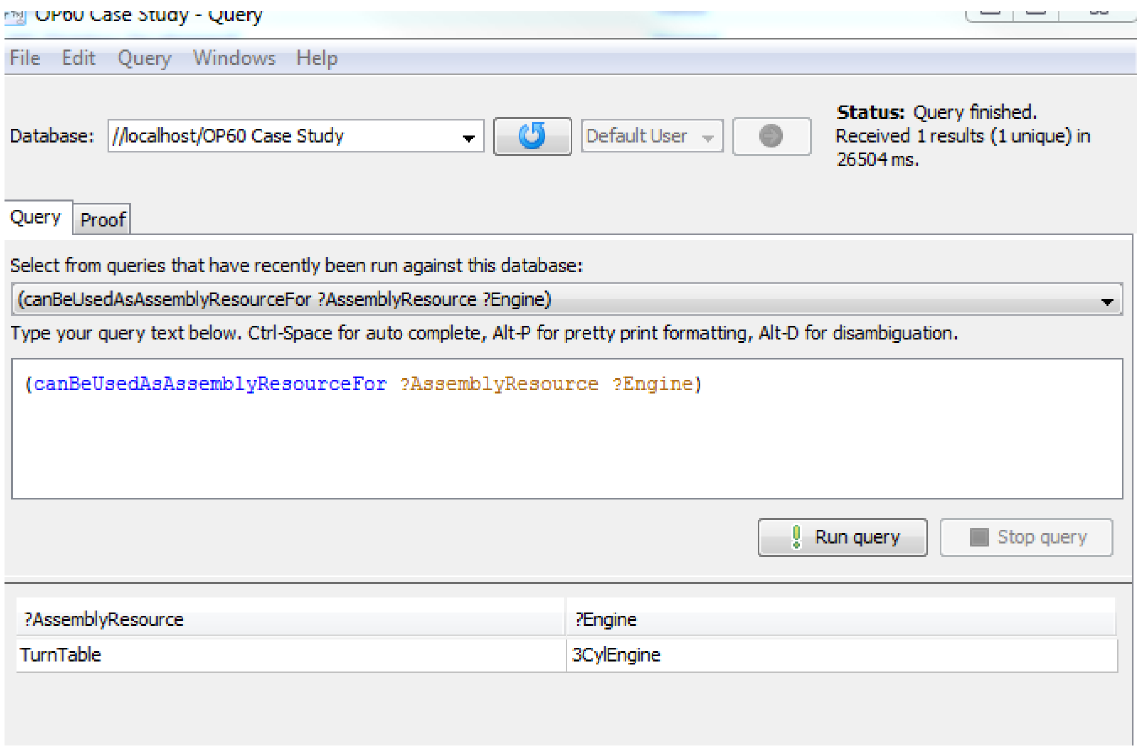
Similarly the angular location of these features has been considered from the same reference point whereas the angles measured are between the central axis of these features with those of the X, Y, and Z planes. The relation hasAngularLocation has been instantiated to relate the product and resource handling features with their angular dimensions as shown in the table 6-4.

6.4.4.2 Evaluating the Assembly Resource for Existing and New Products

Queries can be made to evaluate the assembly resource for a range of products. As this case study considers only two products: 3CylEngine and 4CylEngine, therefore the results are expected for these two products only. Figure 6.22 displays the result of the following query:

```
(canBeUsedAsAssemblyResourceFor ?AssemblyResource ?Engine)
```

The result actually shows that the assembly resource “TurnTable” (shown in figure 6.17) can be used for existing product “3CylEngine”. However as the facts have been asserted for both 3 and 4 cylinder engine, the system suggests that the turn table in its current condition cannot be used for 4 cylinder engine.



Status: Query finished. Received 1 results (1 unique) in 26504 ms.

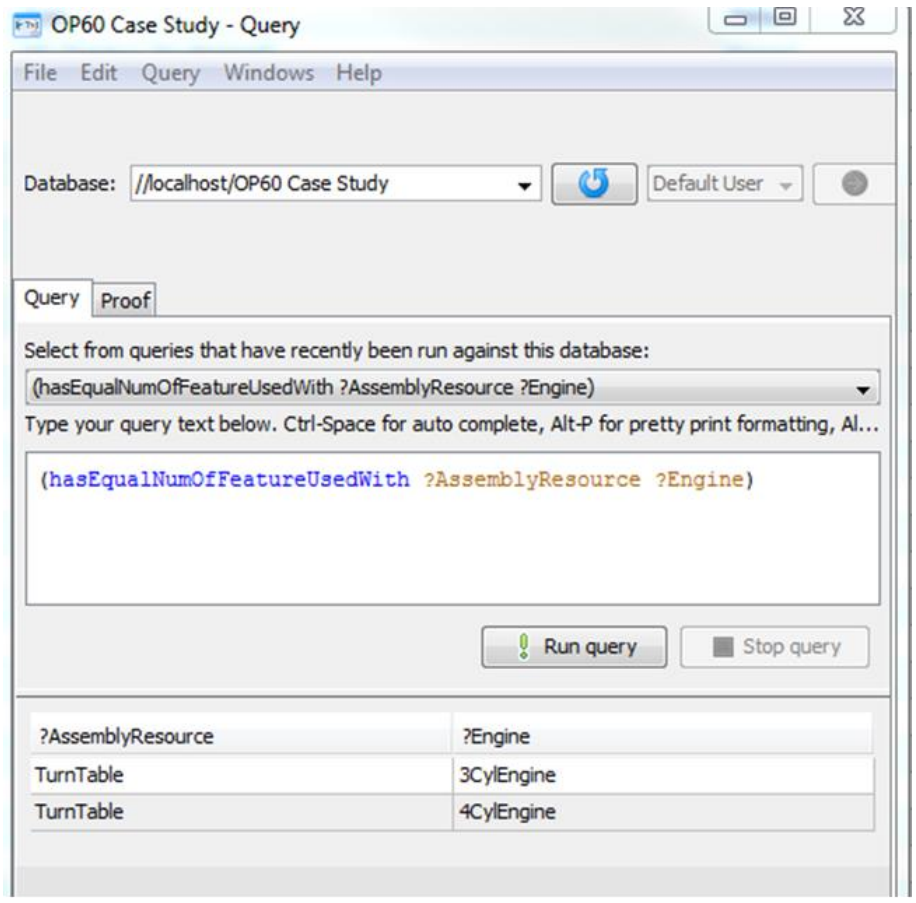
Select from queries that have recently been run against this database:
(canBeUsedAsAssemblyResourceFor ?AssemblyResource ?Engine)
Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(canBeUsedAsAssemblyResourceFor ?AssemblyResource ?Engine)
```

?AssemblyResource	?Engine
TurnTable	3CylEngine

Figure 6.22: Turn table suitability assessment query for 3 cylinder and 4 cylinder engines

Further queries can be made to explore the reasons why turn table cannot be used for 4 cylinder engine. The three conditions described in section 6.4.3, can be used as queries to find out whether the turn table can be used for both engines. These queries are shown in figure 6.23 where assembly resource has been assessed for the products by using the feature quantity, feature size and feature location conditions. The queries validate that the feature quantity and size conditions have been met for both engines however the feature location condition is only met for 3 cylinder engine as shown in figure 6.23 (b). Further queries for feature location have shown that the angular location of resource and product features are mateable (as shown in figure 6.23 (a) and 6.23 (c)) however the point location of these features are not mateable because they are located at different positions. This implies that the turn table in its current state cannot be used for 4 cylinder engine.



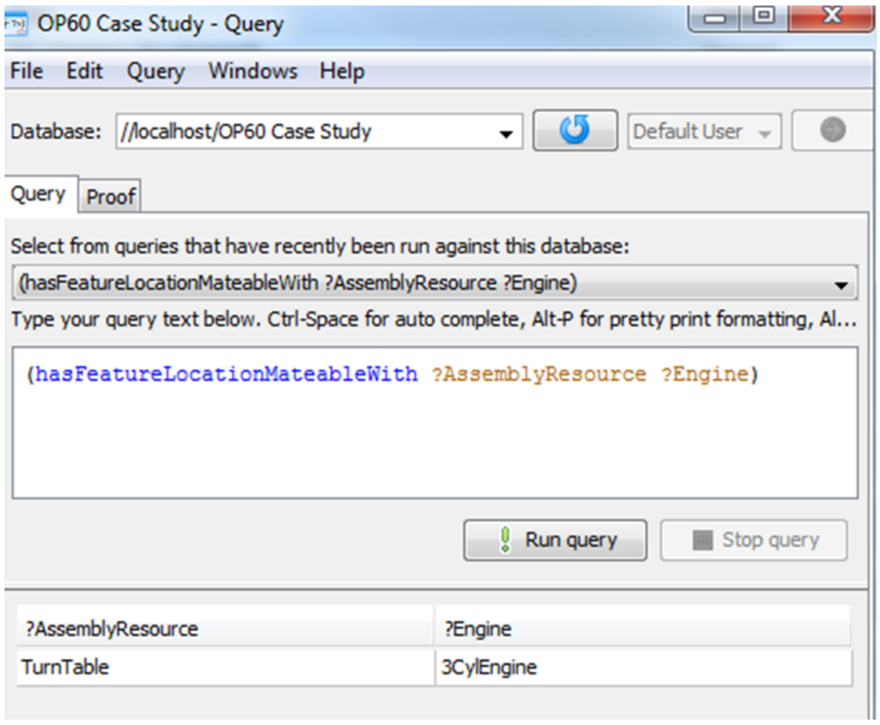
Database: //localhost/OP60 Case Study

Select from queries that have recently been run against this database:
(hasEqualNumOfFeatureUsedWith ?AssemblyResource ?Engine)
Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Al...

```
(hasEqualNumOfFeatureUsedWith ?AssemblyResource ?Engine)
```

?AssemblyResource	?Engine
TurnTable	3CylEngine
TurnTable	4CylEngine

6.23 (a)


OP60 Case Study - Query
File Edit Query Windows Help
Database: //localhost/OP60 Case Study
Default User
Query
Proof
Run query
Stop query

Select from queries that have recently been run against this database:

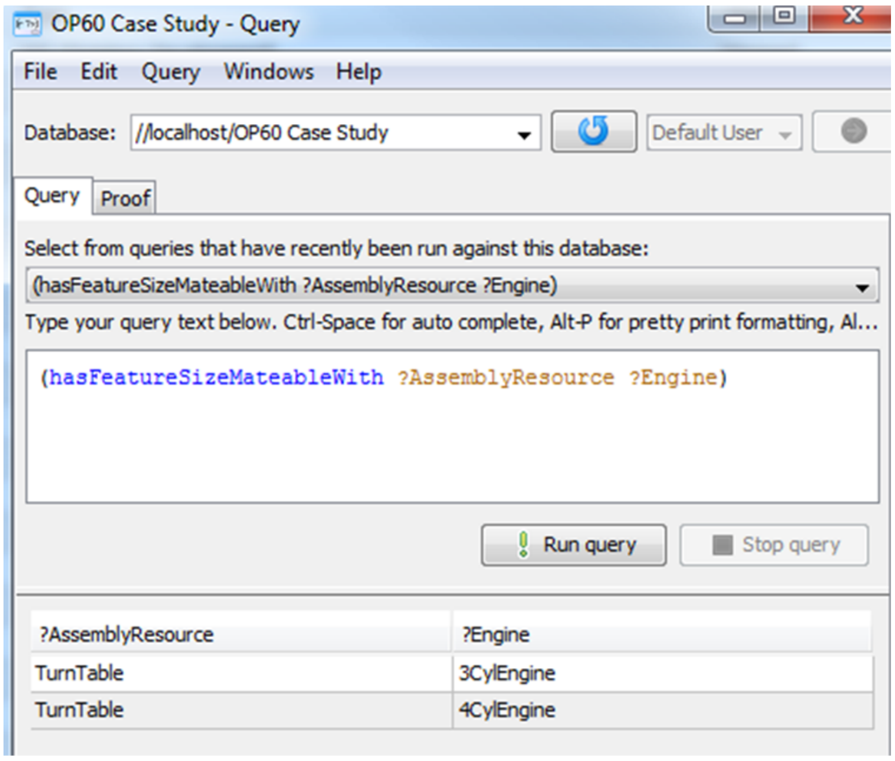
(hasFeatureLocationMateableWith ?AssemblyResource ?Engine)

Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Al...

```
(hasFeatureLocationMateableWith ?AssemblyResource ?Engine)
```

?AssemblyResource	?Engine
TurnTable	3CylEngine

6.23 (b)


OP60 Case Study - Query
File Edit Query Windows Help
Database: //localhost/OP60 Case Study
Default User
Query
Proof
Run query
Stop query

Select from queries that have recently been run against this database:

(hasFeatureSizeMateableWith ?AssemblyResource ?Engine)

Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Al...

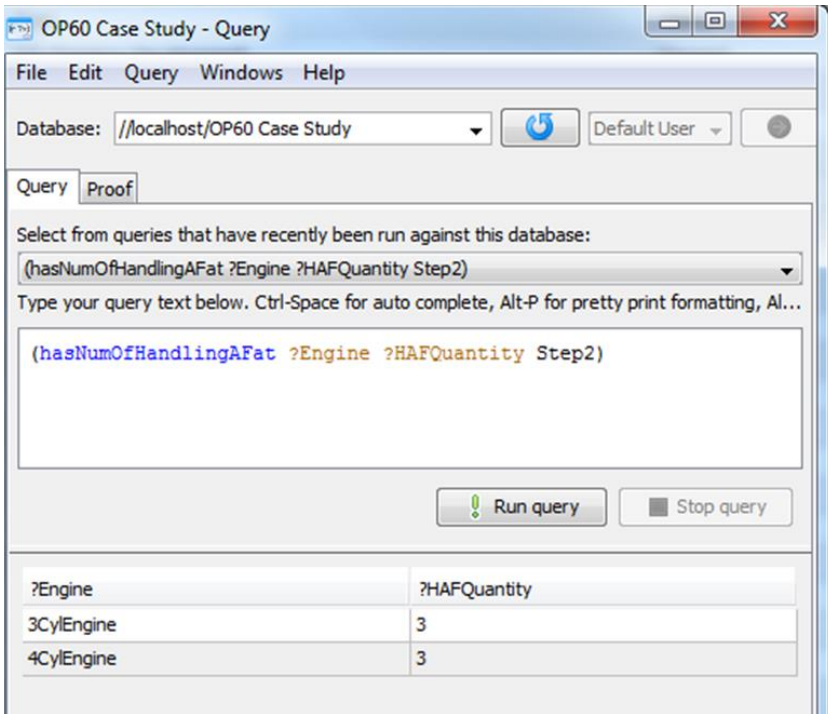
```
(hasFeatureSizeMateableWith ?AssemblyResource ?Engine)
```

?AssemblyResource	?Engine
TurnTable	3CylEngine
TurnTable	4CylEngine

6.23 (c)

Figure 6.23: Queries for assessment of feature quantity, feature size and feature location compatibility.

Other queries can also be made to verify the quantity and size of the product and resource features. For example, the query shown in figure 6.24 determines the number of handling assembly features used at step 2. The query result validates that the number of handling assembly features for both engines is 3 which can also be seen in figure 6.21.



Database: //localhost/OP60 Case Study

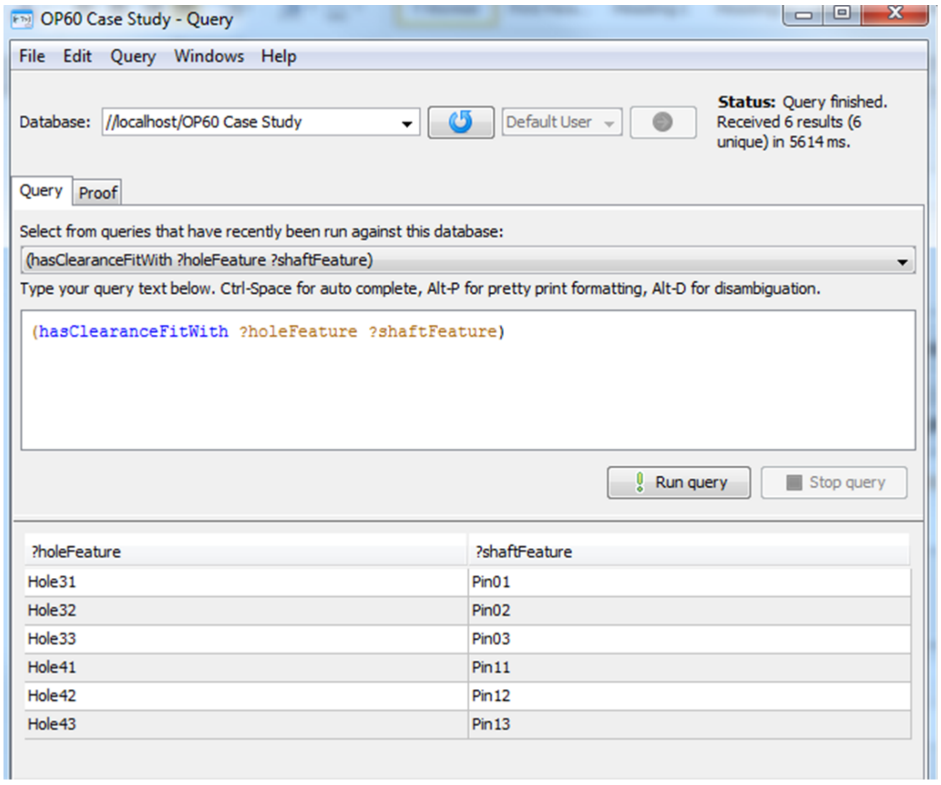
Select from queries that have recently been run against this database:
(hasNumOfHandlingAFat ?Engine ?HAFQuantity Step2)
Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Al...

```
(hasNumOfHandlingAFat ?Engine ?HAFQuantity Step2)
```

?Engine	?HAFQuantity
3CylEngine	3
4CylEngine	3

Figure 6.24: Query to find out number of handling assembly features used at step 2

As this case study also uses the standard tolerance information (see chapter 5 for more detail) to apply the tolerance grades therefore queries can also be made to validate the clearance fit condition for product and resource feature size compatibility. The tolerance type for product handling features also instantiated as hole assembly features is H8, and tolerance type for resource handling features also instantiated as shaft resource handling features is f8. Therefore the product and resource handling features should have a clearance fit with each other. A query has been made to find out the clearance fit between product and resource features which validates that all product handling features used at step 2 have clearance fit with resource handling features. This query is shown in figure 6.25.


OP60 Case Study - Query
File Edit Query Windows Help
Database: //localhost/OP60 Case Study
Default User
Query
Proof
Run query
Stop query

Status: Query finished. Received 6 results (6 unique) in 5614 ms.

Select from queries that have recently been run against this database:

(hasClearanceFitWith ?holeFeature ?shaftFeature)

Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(hasClearanceFitWith ?holeFeature ?shaftFeature)
```

?holeFeature	?shaftFeature
Hole31	Pin01
Hole32	Pin02
Hole33	Pin03
Hole41	Pin11
Hole42	Pin12
Hole43	Pin13

Figure 6.25: Clearance fit between product and resource handling features

6.4.4.3 Modification of Assembly Resource Design and Re-evaluation

As the turn table cannot be used for the 4 cylinder engine due to its feature location therefore the automotive company has modified the design of the existing turn table and have included three more handling resource features. The modified turn table is shown in figure 6.26. The pins shown in green colours are the same pins used for three cylinder engine. However the purple colour pins: Pin11, Pin12 and Pin13 have been added in the modified design to accommodate the 4 cylinder engine. This modified turn table can be simultaneously used for the 3 cylinder as well as 4 cylinder engines. The yellow plate containing the purple pins is elevated when the 4 cylinder engine is brought here. Similarly when the 3 cylinder engine is transported here, the yellow plate with purple pins is pushed down for the engine to be placed on the green pins.

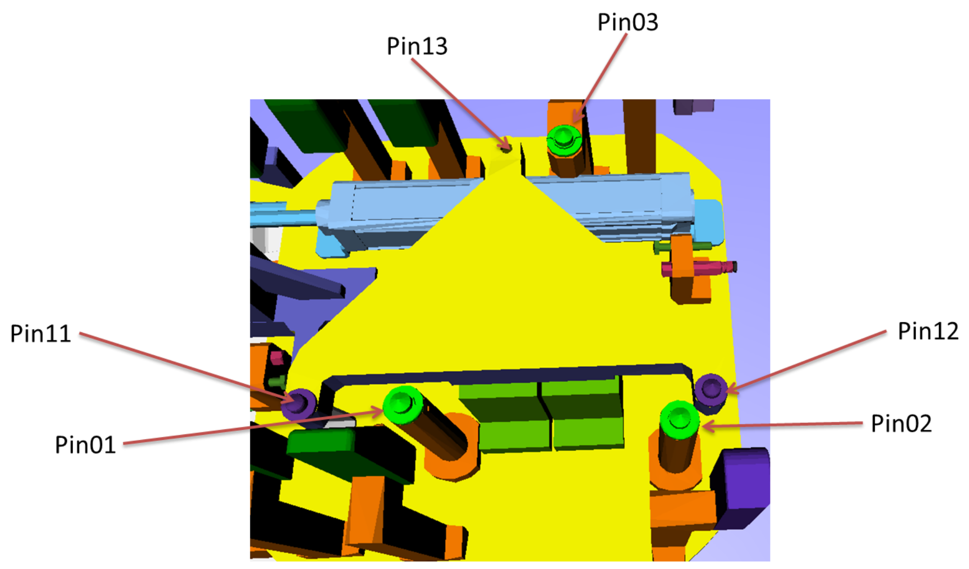


Figure 6.26: Modified design of turn table with the addition of three pins for four cylinder engine

As the turn table is modified therefore new facts have been populated to represent the modified turn table. Table 6-5 shows some of those newly asserted facts. The 3 cylinder engine will use handling resource features: P01, P02, and P03 at step 2, and 4 cylinder engine will use Pin11, Pin12, and Pin13 at step 2, however both the engines interact with different resource features at step 2. Therefore step 2 has been broken down to step 2a and step 2b. The 3 cylinder engine will use step 2a whereas the 4 cylinder engine will use step 2b. This also suggests that Hole31, Hole32, and Hole33 will mate with Pin01, Pin02, and Pin03 whereas Hole41, Hole42, and Hole43 will mate with Pin11, Pin12, and Pin13. These facts are shown in table 6-5.

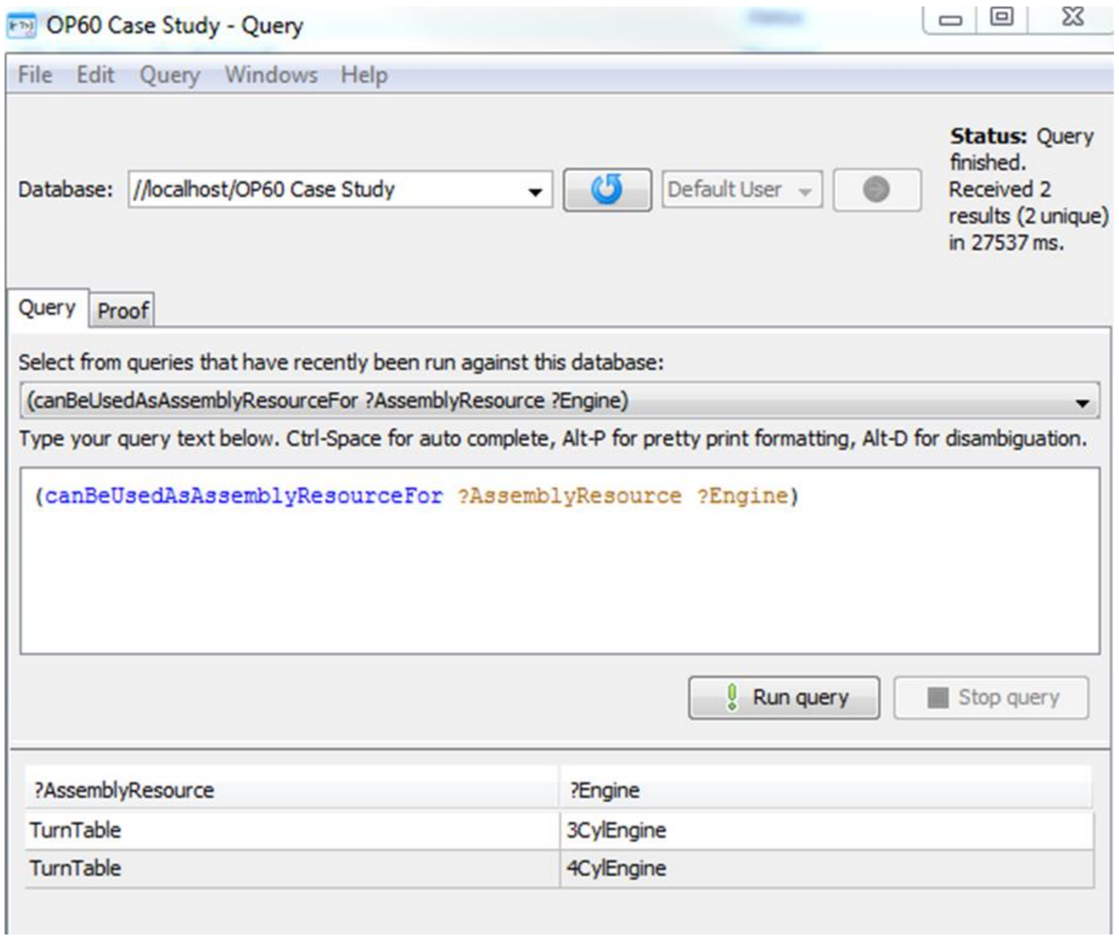
The newly added handling resource features Pin11, Pin12, and Pin13 have the same point location as that of Hole41, Hole42, and Hole43 as shown in table 6-5. The angular location of these features is also the same during their mating.

This implies that the feature location condition is now met for both the engines and therefore the turn table should be useable for both engines.

Table 6-5: Updated facts for modified turn table

Classes and Relations	Instances
Step	Step2a Step2b
HandlingARF, ShaftARF	Pin01 Pin02 Pin03 Pin11 Pin12 Pin13
hasHandlingARF	TurnTable Pin01 TurnTable Pin02 TurnTable Pin03 TurnTable Pin11 TurnTable Pin12 TurnTable Pin13
matesWith	Hole31 Pin01 Hole32 Pin02 Hole33 Pin03 Hole41 Pin11 Hole42 Pin12 Hole43 Pin13
hasPointLocation	Hole31 (pointlocation (mm 0) (mm 0) (mm 0)) Hole32 (pointlocation (mm 210) (mm 0) (mm 0)) Hole33 (pointlocation (mm 105) (mm 238) (mm 0)) Hole41 (pointlocation (mm -70) (mm -2) (mm 20)) Hole42 (pointlocation (mm 229) (mm 20) (mm 20)) Hole43 (pointlocation (mm 54) (mm 226) (mm 20)) Pin01 (pointlocation (mm 0) (mm 0) (mm 0)) Pin02 (pointlocation (mm 210) (mm 0) (mm 0)) Pin03 (pointlocation (mm 105) (mm 238) (mm 0)) Pin11 (pointlocation (mm -72) (mm -2) (mm 20)) Pin12 (pointlocation (mm 229) (mm 20) (mm 20)) Pin13 (pointlocation (mm 54) (mm 226) (mm 20))

Once all of the above facts are asserted in the database base (the underlying ontological structure is the same), similar queries can be made as they were made in section 6.4.4.2. A final resource assessment query has been made and its result is shown in figure 6.27. The result verifies that the turn table can be used as an assembly resource for both 3 cylinder and 4 cylinder engines.



Database: //localhost/OP60 Case Study

Status: Query finished. Received 2 results (2 unique) in 27537 ms.

Select from queries that have recently been run against this database:
(canBeUsedAsAssemblyResourceFor ?AssemblyResource ?Engine)
Type your query text below. Ctrl-Space for auto complete, Alt-P for pretty print formatting, Alt-D for disambiguation.

```
(canBeUsedAsAssemblyResourceFor ?AssemblyResource ?Engine)
```

?AssemblyResource	?Engine
TurnTable	3CylEngine
TurnTable	4CylEngine

Figure 6.27: Query made for assembly resource evaluation

6.4.5 Case Study Concluding Remarks

This case study has provided a practical example of product change effects onto the assembly process planning domain. In this case study, it has been demonstrated that the change of product design has consequences on the assembly process planning domain. An assembly resource was evaluated against a new product and it was found that the existing assembly resource required modifications in order to accommodate the new product.

The underlying ontological structure for this application specific scenario was provided by the ARO which successfully supported it. This implies that the ARO can be used as an underlying structure for different applications. However additional concepts, relationships, and axioms may be required for the application specific domains depending upon the application.

The developed knowledge base has been thoroughly explored and has been validated by making a range of queries. The case study has mainly focussed on hole and shaft type of handling features where an assembly resource has been evaluated using the feature based mating conditions. These conditions are feature quantity, feature size and feature spatial location. It is argued that the application ontology developed using the ARO as an underlying base, can be applicable to the assembly line operations and steps for the similar kind of product and resource features.

Finally it is concluded that the ARO can be linked with multiple domain specific scenarios. The knowledge associated with these scenarios can be shared across the assembly domains using the ARO as a base ontology.

6.5 Summary

This chapter has provided an experimental verification of key aspects of the research work reported in this thesis. Three different experiments have been performed to verify the capture and sharing of assembly knowledge. The results of the first experiment showed that the concepts defined using KFL based heavyweight ontological approach strictly adheres to their formal definitions. The results of the second experiment revealed that the ARO can support the capture of design perspective of assembly knowledge which can be subsequently linked with the assembly process planning knowledge.

The results of the third experiment exposed that the ARO is capable of capturing the assembly process planning viewpoint of assembly knowledge and that the assembly design knowledge can be related with the assembly process planning knowledge using the concept assembly feature.

It has also been recognized that the ARO can facilitate the development of application specific ontologies in the assembly domain by providing a set of formally defined reference concepts. This chapter has also presented a case study to further strengthen and validate the research claims. The results of the case study revealed that the ARO can be applied to the actual industrial scenarios. The results of the case study have also shown that the change in product design can have serious impact on the subsequent assembly process planning phase that such effects can be known during the early stages of design using the ARO as a reference.

CHAPTER 7

RESEARCH FINDINGS, CONCLUSIONS AND FUTURE WORK

7.1 Introduction

The research work reported in this thesis has explored the use of heavyweight reference ontology to support knowledge sharing within the assembly related domain. The ontology was built using a set of key assembly reference concepts and was formally defined in Common Logic (CL) based Knowledge Frame Language (KFL) to support knowledge sharing across the assembly design and assembly process planning domains. The ontology was tested in Integrated Ontology Development Environment (IODE) to verify the proposed research framework.

This chapter summarises the findings, conclusions and future research directions obtained through the implementation of the proposed research framework. Section 7.2 presents the review of the research findings whereas section 7.3 describes the conclusions drawn. Finally section 7.4 indicates the possible future research directions.

7.2 Review of Research Findings

The research work reported in this thesis was undertaken towards finding improved methods for knowledge sharing across the assembly design and assembly process planning domains. In this regard, a comprehensive literature review was carried out to identify and understand potential research gaps and opportunities. It was found that there exists a requirement for potential ontology based methods to support knowledge sharing in the assembly domain.

More narrowly, there is a need to exploit Common Logic (CL) based formal ontological methods to define and relate assembly domain related concepts in order to support knowledge sharing across the assembly design and assembly process planning domains. This fulfilled the first objective of this research work which was to identify potential research gap.

Based on the findings obtained from potential research gaps, further developments were made to identify key issues and requirements related to assembly knowledge sharing. This led to a proposed novel framework that aimed towards providing methods for improved knowledge sharing in the assembly domain. The following sections provide a summary of the key research findings obtained during the investigation of the proposed framework for assembly knowledge sharing.

7.2.1 The ARO Framework as a Set of Reference Concepts

Chapter 2 in general and section 2.7 in particular, highlighted the need to address the knowledge sharing issues in the assembly domain. Further issues related to assembly knowledge sharing were discussed in section 3.2 and the requirements to support assembly knowledge sharing were enumerated in section 3.3. These issues and requirements led towards the proposed Assembly Reference Ontology (ARO) framework described in section 3.4. The idea was to propose a framework in the form of a reference ontology for the assembly domain that should provide an intermediate reusable set of reference concepts to support assembly knowledge sharing. The ARO built on multiple layers of reference concepts ranging from generic to more specialized assembly domain related concepts (see section 3.4.1) which acted as a bridge between the foundation ontology concepts and domain specific concepts. The idea proposed in the form of the ARO framework achieved research objective 2 which was to propose a method for improved assembly knowledge sharing.

The proposed ARO framework described in section 3.4.1 triggered questions such as (1) what are the potential set of ARO concepts? (2) Can these concepts be formally defined using the CL based formal ontological approach? These questions are in line with the requirements and novelty aspects highlighted in sections 3.3 and 3.4.2 respectively. The answer to these questions helped to achieve objectives 3 and 4 of this research (see section 1.2).

Section 4.6 and chapter 5 provided a detailed overview of the formalization of ARO concepts and how they can be linked with assembly design and assembly process planning related concepts. The set of formally defined ARO concepts provided a semantic base for knowledge sharing within the assembly domain. This was experimentally demonstrated in chapter 6 where the ARO knowledge base was built by instantiation of the properties, relationships and functions. The knowledge base was then evaluated by making queries whereas the system returned the expected results thus verifying that the system understood the semantics of the concepts.

The research work carried out in this thesis aimed at exploring the ARO framework as a potential approach to support knowledge sharing across the assembly design and assembly process planning domains. However from commercial point of view, the ARO needs to be validated by assembly domain experts and the end users. This may require inclusion of other reference concepts related to the assembly domain. Furthermore, this thesis work focused on detailed exploration of tolerance and fits related assembly design knowledge and their resulting assembly process planning knowledge, and MBOM related assembly process planning knowledge. The detailed exploration of concepts related to these areas has provided the proof of concept for this research however further exploration may lead towards the identification of more ARO concepts for other areas related to the assembly domain.

For instance, the concepts “product family” and “Bill of Materials (BOM)” require further exploration in order to capture their assembly design and assembly process planning viewpoints to support knowledge sharing across the assembly domain. Similarly, further investigation is required to link the ARO with other domains such as single piece part manufacturing, inspection, operation, maintenance, and disposal.

With the increased public awareness and more strict laws related to the End-of-Life (EOL) product treatment (Ilgin et al., 2011) has resulted in growing trend of product repair, remanufacturing and recycling. In relation to the assembly domain, the repair and remanufacture require disassembly and re-assembly of products whereas recycling may require disassembly only. While this research work focused on potential impacts of assembly onto the design and planning stages, it did not cover the possible influence of disassembly onto the design and planning stages. Thus a further investigation is required to extend the ARO to include the reference concepts related to the product disassembly.

7.2.2 Capturing the Semantics of Concepts at Various Levels of Specializations

This research work has also contributed towards the need to capture the semantics of concepts at various levels of their specialisations, as mentioned in section 3.4.2.2, in order to support capture and sharing of assembly domain related knowledge. Evidence of this approach was shown with the detailed exploration of manufacturing bill of materials (MBOM) concept and its three different specialized concepts (see section 5.2).

The experimental investigation of these concepts was carried out in section 6.3.1 and it was found that the system was able to understand the semantics of MBOM and its specialized concepts. Furthermore, it was shown that the definition of the parent concept was inherited by all of its child classes. Moreover, different restrictions could be placed on concepts at the same specialization level in order to differentiate their semantics.

Semantics captured at more generic level provide a route to knowledge sharing by providing a common semantic base and therefore can support knowledge sharing across the specialized domains. For instance, in section 6.3.1, it was demonstrated that the common concept As Designed (AD) assembly component list provided a route to assembly knowledge sharing across three different interpretations of MBOM. This proof of concept supported the fact that the ARO can behave like a semantic base for domain specific concepts and can support knowledge sharing across multiple application specific domains.

7.2.3 Relating Assembly Design and Assembly Process Planning Knowledge

One of the requirements for assembly knowledge sharing was to relate multiple viewpoints of assembly knowledge as mentioned in section 3.3. The issue of relating multiple viewpoints of assembly knowledge was also identified as a potential research gap described in section 2.7. This research aspect was explored in detail in section 5.3. The concept assembly feature was investigated in detail because it carries both assembly design and assembly process planning implications as mentioned in section 4.3.12.

This thesis considered the tolerancing and fits related information from an assembly design viewpoint and assembly processes, assembly resources, and manufacturing facility related information from an assembly process planning viewpoint. This concept was formally captured in section 5.3.4, and was experimentally investigated in sections 6.3.2 and 6.3.3. The results of experimental investigation showed that assembly design related knowledge can be linked to the assembly process planning knowledge using the concept of assembly feature. The investigation further showed that decisions taken during the design stage can have potential implications during the assembly process planning stage.

For instance, in section 6.3.3 it was established that the assembly design knowledge can be related with assembly process planning knowledge and that the assembly process planning related consequences can be determined during the assembly design stage. More specifically, the design system is able to determine what assembly processes and resources are required against a specific type of assembly fit, and which manufacturing facility is able to carry out the assembly of mating assembly features. In this manner, the designer can assess the potential assembly process planning related issues during the assembly design stage.

The assembly design and assembly process planning related concepts explored in section 5.3 and subsequently investigated in sections 6.3.2, and 6.3.3, have shown the proof of concept. However other similar aspects of assembly design and assembly process planning can be further explored. For instance the design and planning knowledge related to hole and shaft assembly features having shape attributes other than circular can be investigated in future.

The research can also be extended to explore other ARO concepts which have multiple implications in assembly design and assembly process planning. Examples of two such concepts are product family and BOM. Similarly, assembly domain experts can be consulted to find out more concepts which might have different viewpoints for assembly design and assembly process planning domain.

7.2.4 Developing Application Ontology for the Case Study Scenario

In section 3.4.2, it was claimed that the ARO framework can support the development of application ontologies by providing a semantic base for their development. For this purpose, a case study was carried out in the automotive sector as explained in section 6.4. The case study gave an opportunity to test and extend the ARO, and it used the ARO database as a foundation to evaluate product design change effects onto the assembly process planning domain.

The ARO extended for the case study application was successfully implemented and evaluated by asserting the relevant facts and by making queries. The implementation of a case study allowed more realistic proof of concept and demonstrated that the ARO can be used as a base ontology for a range of industrial case scenarios.

Initial investigation was focussed on assembly resource evaluation for multiple products which consequently led towards the feature based mating conditions. The concepts of hole and shaft type handling features linked with assembly design and assembly process planning related knowledge supported the development of an application ontology for the case study scenario. It was shown that the ARO was capable to provide a semantic base to the application ontology.

The application ontology considered product and resource mating conditions based on feature quantity, feature size, and feature location (see section 6.4.3). However these conditions were identified after the author's discussion with the domain experts and the understanding developed from the case study. Therefore further investigation may be required to identify other mating conditions to evaluate the ability of ARO to support the development of application ontologies.

The application ontology for the case study used feature based evaluation for the automotive industry. However for broader applications, the ARO can be explored for, example scenarios, from other industrial sectors e.g. aero industries.

7.2.5 Small Scale Industrial Applications

Apart from the large scale applications of Common Logic (CL) based ontologies in resolving the knowledge sharing issues, they can be used to evaluate and monitor various day to day assembly line activities. Recently, the potential of a CL based formal ontological approach has been discussed with experts working for an automotive industry who were interested in exploration of its possible applications in automotive assembly line. Their particular focus was the assessment of the suitability of assembly resources for a range of products and evaluation of reach and positioning of human operator.

However, even if the CL based database is provided to the companies, they are still required to input facts and make queries. Chungoora (2010) noted that nonetheless, the fact assertion and query building are supported by the use of Application Programming Interfaces (APIs), appropriate training of users of such systems would be needed.

Amongst other applications, ARO can be extended to support the intelligent energy systems. For example, Ghani et al. (2011) proposed an integrated energy monitoring system in an automotive assembly line which suggests switching-off non-utilised machine (assembly resource) modules during the product assembly operations. These types of energy monitoring systems require application of rules to control the switch-on and switch-off activities of the machine modules. This is where formal ontologies can be applied and the extended ARO could have great potential to deal with these kinds of activities.

ARO can also be used to build libraries of assembly knowledge for shop floor activities. For instance, Chakrbarty et al. (2009) used a lightweight ontology based method to support the search and retrieval of information related to the shop floor assembly problems in an automotive industry. Similalrly the libraries of assembly knowledge can be built using the CL based computationally powerful approach which can effectively provide support to industries to access the relevant information with ease and within minimum possible time.

While there would be a wide range of industrial applications, however the ARO needs to be extended, implemented and validated to further evaluate its ability in the broader spectrum.

7.2.6 Evaluation of CL based Ontological Approach

In section 2.4.2.5 it was reported that CL based ontological approaches are more expressive and can capture more complex semantics as compared to other approaches particularly OWL. Literature has also revealed that most of the existing heavyweight ontologies are based on OWL and/or rule based language: Semantic Web Rule Language (SWRL) (See sections 2.5 and 2.6.5). The research in this thesis also verifies that CL is more capable in expressing and reasoning the complex semantics in the assembly domain.

For instance, it was revealed in sections 2.4.2.2 and 2.4.2.5 that in comparison with the CL based approach, OWL does not directly support ternary and higher order relations, and binary and higher order functions. However the research in this thesis has shown that CL based KFL allows the declaration of complex relations and functions having arity equal to or more than 3 and 2 respectively. In KFL, the arity is declared using the signature declaration and this could have any number of arguments. For instance, relations “hasAllowanceWith”, “hasAssemblyProcessWith”, and “isAssembledWithIn” are examples of ternary relations which have been used in section 5.3.4 and section 6.3. Similarly binary functions: tolerance, H7, H8, f7, and p6 (see section 5.3.4) and, ternary functions: pointlocation and angularlocation (see section 6.4.4.1) have been captured using KFL.

It was also mentioned in section 2.4.2.5 that KFL has distinct advantage over OWL that the former is built on closed world assumption which means that the things which are currently not known to be true, are considered false (Palmer et al. 2012). This leads to the fact that all KFL based query statements would only be true if and only if these statements are declared true in the asserted ARO database. In addition, the accuracy of KFL based queries has also been verified by checking that the system returns expected results from sets of known conditions. For example, consider the axiom in section 5.3.4.1.1 which infers the H7 tolerance quantities based on the basic dimension of assembly feature. The query for minimum and maximum allowable dimensions of the assembly feature (BearingHousing01) having basic dimension of 52 mm returns 52 and 52.03 mm respectively as shown in the query in figure 6.11 in section 6.3.2 as well.

In sections 2.4.2.2 and 2.4.2.5, it was also noted that in contrast to CL based approach, OWL does not support conjunction, disjunction, and negation operators. The conjunction operator “and” and the disjunction operators “or” have been widely used in the formalization of ARO and related concepts as can be seen in axioms in section 5.3.4. Furthermore KFL can effectively support the negation operator “not” which is used to negate a directive and can be potentially helpful where it is required to infer or retrieve information minus “not” directive. One such example can be found in section 5.2.3 where negation operator was used to define the semantics of the concept assembly component list and its subsumptions.

Palmer et al. (2012) found that SWRL does not support Integrity Constraints (ICs). However in contrast, CL based KFL supports the application of ICs to define the domain semantics. ICs prevent faulty assertions and only allow true assertions thus improving the accuracy and reliability of the model. This research work has successfully exploited the use of ICs in defining the ARO concepts. Examples of such ICs can be found in sections 4.6.4.1, 5.2.3, 5.3.4.1 and they were experimentally evaluated in chapter 6.

7.2.7 Novel Aspects of Research Work

This research work has made various contributions to resolve interoperability issues in the assembly domain and provided a step towards the formal assembly reference ontology for assembly knowledge sharing. The following points highlight the key novel aspects of this research work.

- A novel framework in the form of ARO has been proposed to support knowledge sharing within the assembly domain. Multiple levels of ARO concepts have been identified starting from the most generic level to most specialized assembly design and assembly process planning related levels.

- A key set of ARO concepts have been identified by reviewing existing literature and exploring different assembly design and assembly process planning related software systems. These ARO concepts represent key aspects of assembly domain.

- The understanding gained from the informal description of these ARO concepts led towards the development of UML based lightweight representation. The ARO concepts were specialized and generalised, and other relationships were identified to relate different ARO concepts.

- The UML based lightweight model provided visual support to formally define the ARO concepts. The CL based ontological formalism has been used to formally represent the ARO.

- Multiple interpretations of MBOM were identified and formally captured using the ARO concepts. This helped to understand that the varying meanings of concepts at different specialization levels can be captured and subsequently a route to enable knowledge sharing can be identified.

- It has been demonstrated that design and planning knowledge can be linked with ARO concepts and subsequently can be shared across the assembly design and assembly process planning domains.

- The ARO has used extensive axiomatization to capture the tolerance and assembly fits related design knowledge, and the resulting assembly process planning knowledge. The ARO database has been successfully evaluated by facts assertion and queries, and can be used to support decision making in the assembly domain.

- This research identified a feature based method to evaluate the assembly resource against a range of products for an automotive assembly line. The research identified three mating conditions based on quantity, location and size of the product and resource assembly features. This feature based assembly resource evaluation method can be exploited for a range of assembly line scenarios.

7.3 Conclusions

The research work reported in this thesis has demonstrated the potential of heavyweight ontologies to support knowledge sharing across the assembly design and assembly process planning domains. It is evident from this research that heavyweight ontologies can play a significant role in establishing the semantics of concepts related to the assembly domain which can consequently provide a base for knowledge sharing. The following paragraphs provide key conclusions drawn from this research work.

- The comprehensive literature review has shown that there is a need to exploit ontology based methods to support knowledge sharing in the assembly domain and the use of formal reference ontologies are emerging as promising candidates for the assembly domain.
- A knowledge sharing framework in the form of ARO was proposed and was experimentally evaluated using a selected set of reference concepts. The detailed investigation of these selected sets of ARO concepts provided a proof of concept for this research and supported the argument that the ARO can support knowledge sharing across the assembly design and assembly process planning domains.
- The formal definition of MBOM and its subsumptions proved that the meanings of concepts can be captured at various levels of concept specializations and that the system understands these meanings. It was also shown that the route for knowledge sharing can be established by identifying the common concepts.
- It was found that the multiple viewpoints of concepts can be captured and these concepts (when formally defined) can be used to relate

assembly design knowledge with the assembly process planning knowledge. This was shown by capturing the tolerance and fits related design knowledge associated with the assembly feature concept and the resulting assembly process planning knowledge. It was established that the assembly knowledge can be shared across the assembly design and assembly process planning domains using the ARO.

- Findings of this research also conclude that the ARO can be used as a semantic base to develop application ontologies. This was shown by the implementation of the ARO in the automotive assembly line scenario and by the development of application specific design and planning concepts from the ARO concepts.
- The CL based KFL was used as ontological formalism for the ARO which enabled the capture and sharing of assembly domain knowledge. This research has shown that the expressive power and inference capabilities of KFL have proven to be capable of representing the semantic complexities of the multiple inter-relationships involved in assembly.

7.4 Future Work

The findings of the research work carried out in this thesis also suggest recommendations for the future research. The following paragraphs highlight key future research directions.

The research work explored in this thesis demonstrated the proof of concept by detailed investigation of selected set of ARO concepts. However the research can be extended to explore other ARO concepts which have the potential to relate assembly design and assembly process planning domains. Potential examples of such concepts are Product family and BOM.

The hole and shaft assembly features having circular shape attributes were explored in detail in this thesis to relate assembly design and assembly process planning knowledge. Future research can be extended to explore hole and shaft assembly features having other shape attributes. Furthermore other types of assembly features e.g. plane mate assembly features and alignment assembly features can be explored to support capture and sharing of assembly knowledge.

In this thesis, the selection of assembly processes and resources was based on the type of fit. However other factors can also be explored which affect the selection of assembly processes and resources. For example, weight, size, quantity, and material type can also affect the selection of assembly processes and resources. Therefore a further investigation is required to explore the ARO to accommodate these factors to support the selection of assembly processes and resources.

The ARO exploits KFL based computationally powerful approach which can be potentially used to support complex scenarios in the assembly domain. For instance, the ARO can be explored further to capture the assembly sequence planning (ASP) related knowledge which potentially requires the application of constraints and rules. Furthermore, Demoly et al. (2011) found that considering ASP related knowledge during the early phases of design is an emerging and novel research area which promotes the assembly oriented design. Hence the ARO can be further investigated to support capture and sharing of ASP related knowledge.

In this thesis, the tolerance related knowledge was formally captured and linked with the assembly process planning related knowledge. However the tolerance knowledge is also related to the single piece part manufacturing domain where it depends upon the capabilities of manufacturing processes. The ARO can be extended to link the tolerance issues back to the single piece part manufacturing.

The ARO concepts considered in this thesis cover wide range of assembly domain aspects. However the ARO need to be explored further to find out other relevant aspects. This can be done by exploiting the use of ARO in a range of scenarios within the assembly domain to support the capture and sharing assembly knowledge.

As discussed in the research findings, ARO can be potentially exploited for assembly shop floor monitoring and control activities. One such area is energy monitoring of machine (assembly resource) components on assembly line where the ARO can be explored to intelligently control the machine components for a range of products.

The ARO concepts can be exploited for other related domains where products are disassembled and re-assembled as part of the domain activity. Two such domains are: (1) repair, and (2) remanufacturing. These domains require disassembly and re-assembly of products; therefore the ARO can be investigated to explore these domains to support the capture and sharing of assembly knowledge.

There are many potential avenues for further research in relation to reference ontologies. For example, the current approach can be explored for knowledge capture and sharing in product lifecycle domains like operations, and disposal as well as other domains such as business, and finance related domains.

PUBLICATIONS

Journal Publication

- **Imran**, M., & Young, B. (2013). The application of common logic based formal ontologies to assembly knowledge sharing. Journal of intelligent manufacturing. doi:10.1007/s10845-013-0768-4

Conference Publications

- Young, R., Hastilow, N., **Imran**, M., Chungoora, N., Usman, Z., & Cutting-Decelle, A.-F. (2013). Reference ontologies for manufacturing. International IFIP Working Conference on Enterprise Interoperability Information, Services and Processes for the Interoperable Economy and Society. Enschede.

- **Imran**, M., Young, B., & Usman, Z. (2012). Formal assembly reference ontology for assembly systems compatibility assessment. Proceedings of the 10th International Conference on Manufacturing Research ICMR 2012. Birmingham.

REFERENCES

Anderson, J. A. (2007). The Origo Handbook of Mathematics Education. Origo Education.

Angele, J., Kifer, M., & Lausen, G. (2009). Ontologies in F-Logic. In S. Staab, & R. Studer (Eds.), Handbook on Ontologies (pp. 46-69). Berlin: Springer-Verlag.

Anjum, N. A. (2011). Verification of Knowledge Shared Across Design and Manufacture Using a Foundation Ontology. PhD Thesis, Loughborough University, Wolfson School of Mechanical and Manufacturing Engineering.

Antoniou, G., & Harmelen, F. v. (2009). Web Ontology Language: OWL. In S. Staab, & R. Studder (Eds.), Handbook on Ontologies (pp. 91-110). Berlin: Springer-Verlag.

Arpírez, J. C., Corcho, O., Fernández-López, M., & Gómez-Pérez, A. (2001). WebODE: a scalable workbench for ontological engineering. K-CAP '01 Proceedings of the 1st international conference on Knowledge capture . New York.

Barton, J. A., Love, D. M., & Taylor, G. D. (2001). Design determines 70% of cost? A review of implications for design evaluation. Journal of Engineering Design, 12(1), 47-58.

Baudin, M. (2002). Lean assembly: the nuts and bolts of making assembly operations flow. Productivity Press.

Bauer, A., Bowden, R., Browne, J., Duggan, J., & Lyons, G. J. (1991). Shoop Floor Control Systems: From Design to Implementation. Ipswich: Chapman & Hall.

Bechky, B. A. (2003). Sharing Meaning Across Occupational Communities: The Transformation of Understandog on a Production Floor. ORGANIZATION SCIENCE, 14(3), 312-330.

Bennett, A. E., & Siy, L. J. (2009). Blueprint Reading For Welders. NewYork: Delamr, Cengage Learning.

Bley, H., & Franke, C. (2004). Integration of Product Design and Assembly Planning in the Digital Factory. CIRP Annals - Manufacturing Technology, 53(1), 25-30.

Borgo , S., & Leitao, P. (2004). The Role of Foundational Ontologies in Manufacturing Domain Applications. In R. Meersman, & Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE (Vol. 14, pp. 670-688). Springer.

Borgo , S., & Leitao, P. (2007). Foundation for a Core Ontology for Manufacturing. In Ontologies Integrated Series in Information Systems (Vol. 14, pp. 751-775). Springer.

Borgo, S., & Lesmo, L. (2008). The Attractiveness of Foundational Ontologies in Industry. In S. Borgo, & L. Lesmo, Formal Ontologies Meet Industry (pp. 1-9). IOS Press.

Borst, W. N. (1997). Construction of Engineering Ontologies for Knowledge Sharing and Reuse. Enschede, The Netherlands: W.N. Borst.

Brickley, D., & Guha, R. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. Retrieved 07 26, 2013, from http://www.w3.org/TR/rdf-schema/

Brinkley, J. F., Suciu, D., Detwiler, L. T., Gennari, J. H., & Rosse, C. (2006). A framework for using reference ontologies as a foundation for the semantic web. AMIA 2006 Symposium Proceedings Page.

Bruijn, J. d. (2007). Logics for the Semantic Web. In J. Cardoso (Ed.), Semantic Web Services: Theory, Tools, and Applications (pp. 24-43). IGI Global.

BS 4500. (1969). ISO Limits and Fits Standard.

BS 5191. (1975). British Standard: Glossary of Production Planning and Control Terms.

Burgun, A. (2006). Desiderata for domain reference ontologies in biomedicine. Journal of Biomedical Informatics, 39, 307-313.

Byrnes, D. (2011). AutoCAD For Dummies. Indiana: Wiley Publishing Inc.

Cardoso, J. (2007). The Semantic Web Vision: Where are We? IEEE Intelligent Systems, 22-26.

Case, K., & Harun, W. W. (2000). Feature-based representation for manufacturing planning. International Journal of Production Research, 38(17), 4285-4300.

Castano, S., & Antonellis, V. D. (1997). Semantic Dictionary Design for Database Interoperability. 13th International Conference on Data Engineering.

Chakrbarty, S., Chougule, R., & Lespe, R. M. (2009). Ontology-guided knowledge retrieval in an automobile assembly environment. International Journal of Advanced Manufacturing Technology, 44, 1237-1249.

Chandrasekaran, B., Josephson, J. R., & Benjamins, V. R. (1999). What are ontologies, and why do we need them? Intelligent Systems and their Applications, IEEE, 14(1), 20-26.

Chang, S.-H., Lee, W.-L., & Li, R.-K. (1997). Manufacturing bill-of-material planning. PRODUCTION PLANNING & CONTROL, 8(5), 437-450.

Chang, X., Rai, R., & Terpenny, J. (2010). Development and Utilization of Ontologies in Design for Manufacturing. Journal of Mechanical Design, 132(2).

Chen, D. (2006). Framework for Enterprise Interoperability. Proc. of IFAC Workshop EI2N.

Chen, D., & Doumeingts, G. (2003). European initiatives to develop interoperability of enterprise applications—basic concepts, framework and roadmap. Annual Reviews in Control, 27(2), 153-162.

Chen, D., & Vernadat, F. (2004). Standards on enterprise integration and engineering - state of the art. INT. J. COMPUTER INTEGRATED MANUFACTURING, 178(3), 235-253.

Cho, S. H. (2005). Mechanical Assembly. In F. Kreith, The Mechanical Engineering Handbook Series. CRC Press LLC.

Chungoora, N. (2010). A Framework to Support Semantic Interoperability in Product Design and Manufacture. Loughborough: Nitishal Chungoora.

Chungoora, N., & Young, R. (2011a). The configuration of design and manufacture knowledge models from a heavyweight ontological foundation. International Journal for Production Research, 49(15), 4701–4725.

Chungoora, N., & Young, R. (2011b). A Framework to Support Semantic Interoperability in Product Design and Manufacture. In A. Bernard, Global Product Development, Proceedings of 20th CIRP Design Conference (pp. 435-443). Springer-Verlag.

Chungoora, N., Cutting-Decelle, A., Young, R., Gunendran, G., Usman, Z., Harding, J. A., & Case, K. (2013). Towards the ontology-based consolidation of production-centric standards. International Journal of Production Research, 51(2), 327-345.

Chungoora, N., Gunendran, G. A., Young, R. I., Usman, Z., Anjum, N. A., Palmer, C., . . . Cutting-Decelle, A.-F. (2012). Extending product lifecycle management for manufacturing knowledge sharing. J Engineering Manufacture, 226(12), 2047-2063.

Ciocoiu, M., Nau, S. D., & Gruninger, M. (2001). Ontologies for Integrating Engineering Applications. Transactions of the ASME, 1, 12-22.

Cochrane, S. D., Case, K., Young, R. I., Harding, J. A., & Dani, S. (2005). knowledge sharing between design and manufacture. In R. J. Rajiv Khosla, Knowledge-based intelligent information and engineering systems: 9th International Conference KES 2005 (pp. 221-227). Berlin: Springer-Verlag.

Comaa, O., Mascle, C., & Veron, P. (2003). Geometric and form feature recognition tools applied to a design for assembly methodology. Computer-Aided Design, 35, 1193–1210.

Corcho, O., Fernandez-Lopez, M., & Gomez-Perez, A. (2002). Methodologies, tools and languages for building ontologies. Where is their meeting point? Data & Knowledge Engineering, 46, 41-64.

Danloy, J., Leroy, A., Lit, D. P., & Rekiek, B. (1999). A Pragmatic Approach For Precedence Graph Generation. IEEE International Symposium on Assembly and Task Planning, (pp. 387-392). Porto, Portugal.

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

Date, C. J. (2007). Logic and Databases The Roots of Relational Theory. Trafford Publishing.

Decker, S., Melnik, S., Harmelen, F. V., Fensel, D., Klein, M., Broekstra, J., . . . Horrocks, I. (2000). The Semantic Web: The Roles of XML and RDF. IEEE Internet Computing, 4(5), 63-74.

Delamer, I. M., & Lastra, J. L. (2006). Ontology Modeling of Assembly Processes and Systems using Semantic Web Services. International Conference on Industrial Informatics (pp. 611-617). IEEE.

Delchambre, A. (1992). Computer-aided Assembly Planning. Chapman & Hall.

Delugach, H. S. (2008). Towards conceptual structures interoperability using common logic. Proc. of the Third Conceptual Structures Tool Interoperability Workshop, held at the 16th International Conference on Conceptual Structures (ICCS 2008), July 7, 2008, UTM, (pp. 13-21). Toulouse, France.

Delugach, H. S. (2009). Representing metadata constraints in Common Logic. Int. J. Metadata, Semantics and Ontologies, 4(4), 277-286.

Demoly, F., Matsokis, A., & Kiritsis, D. (2012). A mereotopologicalproductrelationshipdescriptionapproach for assemblyorienteddesign. Robotics andComputer-IntegratedManufacturing, 28, 681–693.

Demoly, F., Yan, X.-T., Eynard, B., Rivest, L., & Gomes, S. (2011). An assembly oriented design framework for product structure engineering and assembly sequence planning. Robotics and Computer-Integrated Manufacturing, 27(1), 33-46.

Deneux, D. (1999). Introduction to Assembly Features: An Illustrated Synthesis Methodology. Journal of Intelligent Manufacturing, 10, 29-39.

Dictionary of Engineering. (2003). Dictionary if Engineering (2nd ed.). McGraw-Hill.

Dorador, J. (2001). Product and Process Informtation Interaction in Assembly Decision Support Systems. Loughborough University, Wolfson School. Loughborough: J.M. Dorador.

Dorador, J. M., & Young, R. (1999). Information Models to Support the Interaction Between Design For Assembly And Assembly Process PLanning. Proceedings of 1999 IEEE International Symposium on Assembly and Task Planning (pp. 39-44). Porto, Portugal: IEEE.

Elanchezhian, C., Selwyn, T. S., & Sundar, G. S. (2005). Computer Aided Manufacturing (First ed.). New Delhi: Laxmi Publications.

Falkenauer, E., & Delchambre, A. (1993). Resource Planning in The SCOPES Project. Proceedings of the 26th International Symposium on Automotive Technology and Automation, (pp. 295-302). Aachen, Germany.

Fan, I. S., & Liu, C. K. (1999). Product Family and Variants: Definitions and Models. In J. Ashayeri, W. G. Sullivan, & M. M. Ahmad (Eds.), Flexible Automation and Intelligent Manufacturing (pp. 213-224). New York.

Farquhar, A., Fikes, R., & Rice, J. (1997). The Ontolingua Server: a tool for collaborative ontology construction. International Journal of Human-Computer Studies, 46, 707-727.

Fathi, M. N., Eze, U. C., & Goh, G. G. (2011). Key determinants of knowledge sharing in an electronics manufacturing firm in Malaysia. Library Review, 60(1), 53-67.

Ferndndez, M., Gomez-Perez, A., & Juristo, N. (1997). METHONTOLOGY: from Ontological Art towards Ontological Engineering. AAAI.

FinES-Cluster. (2011). 2012 Fines Standardization TF. Retrieved 08 26, 2013, from http://www.fines-cluster.eu/fines/wp/standardizationtf/2012/03/27/7-4-1-foundation-and-domain-ontologies/#7

Fiorentini, X., Gambino, I., Liang, V.-C., Rachuri, S., Mani, M., & Bock, C. (2007). An Ontology for Assembly Representation. National Institute of Standards and Technology.

Fischer, U., & Stokic, D. (2002). Organisational Knowledge Management in Manufacturing Enterprises Solutions and Open Issues. In B. Stanford-Smith, E. Chiozza, & M. Edin (Eds.), Challenges and Achievements in E-business and E-work (pp. 819-826). IOS Press.

Fürst, F. (2005). Axiom-Based Ontology Matching. Proceedings of the 3rd international conference on Knowledge capture (pp. 195-196). New York, NY, USA: K-CAP '05 .

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., & Schneider, L. (2002). Sweetening Ontologies with DOLCE. In A. Gomez-Perez, & V. R. Benjamins (Ed.), Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web (pp. 166-181). Siguenza, Spain: Springer-Verlag.

Gasevic, D., Djuric, D., & Devedzic, V. (2006). Model Driven Architecture and Ontology Development. Berlin: Springer-Verlag.

Genesereth, M. R., Fikes, F. E., Bobrow, D., Brachman, R., Gruber, T., Hayes, P., . . . Schubert, L. (1992). Knowledge Interchange Format Version 3.0 Reference Manual. Stanford, California: Stanford University.

Ghani, U., Monfared, R., & Harrison, R. (2011). Energy Based Efficient Resources for Real Time Manufacturing Systems. Proceedings of the World Congress on Engineering, I. London.

Ginsberg, M. L. (1991). Knowledge Interchange Format: The KIF of Death. AI Magazine, 12(3), 57-63.

Gomez-Perez, A., Fernandez-Lopez, M., & Corcho, O. (2004). Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. London: Springer.

Grenon, P. (2003). [DL] CFP: Reference Ontology and Applications Ontology: Workshop in Hamburg, 15-18 September 2003. Retrieved 08 26, 2013, from http://dl.kr.org/pipermail/dl/2003/001594.html

Gruber, T. R. (1993). A translation approach to portable ontology specification. Knowledge Acquisition, 5(2), 199-220.

Gruninger, M., & Delaval, A. (2009). A First-Order Cutting Process Ontology for Sheet Metal Parts. Proceedings of the 2009 conference on Formal Ontologies Meet Industry (pp. 22-33). IOS Press Amsterdam.

Gruninger, M., & Fox, M. S. (1995). Methodology for the Design and Evaluation of Ontologies. International Joint Conference on Artificial Inteligence (IJCAI95), Workshop on Basic Ontological Issues in Knowledge Sharing.

Gruninger, M., Atefi, K., & Fox, M. S. (2001). Ontologies to Support Process Integration in Enterprise Engineering. Computational & Mathematical Organization Theory, 6, 381-394.

Guarino, N. (1997). Understanding, Building and Using Ontologies. International Journal of Human Computer Studies, 46(2-3).

Guarino, N., & Giaretta, P. (1995). Ontologies and Knowledge Bases Towards a Terminological Clarification. In N. Mars, Towards Very Large Knowledge Bases (pp. 25-32). Amsterdam: IOS Press.

Guha, R. V., & Lenat, D. B. (1990). Cyc: A Midterm Report. AI Magazine, 11(3), pp. 32-59. doi:0738-4602/90

Haasis, S., Frank, D., Rommel, B., & Weyrich, M. (2003). Feature Based Integration of Product, Process and Resources. In G. J. Rene Soenen, Feature Based Lifecycle Modelling (pp. 93-108). International Federation For Information Processing.

Hall, W. P., & Jones, M. (2002). Document-based Knowledge Management in Global Engineering and Manufacturing Projects. To be published .

Hardwick, M., Spooner, D. L., Rando, T., & Morris, K. C. (1996). Sharing Manufacturing Information in Virtual Enterprises. COMMUNICATIONS OF THE ACM, 39(2), 46-54.

Heijst, G. v., Schreiber, A. T., & Wielinga, B. J. (1996). Using explicit ontologies in KBS development. International Journal of Human-Computer Studies, 46(2-3), 183-292.

Hendriks, P. (1999). Why Share Knowledge? The Influence of ICT on the Motivation for Knowledge Sharing. Knowledge and Process Management, 6(2), 91-100.

Hill, A. V. (2012). Encyclopedia of Operations Management, A Field Manual and Glossary of Operations Management Terms and Concepts. (B. Render, Ed.) Pearson Education Inc.

Hirata, T. T. (2009). Customer Satisfaction Planning. Taylor & Francis Group.

Ho, D. (2011). Notepad++. Retrieved 08 22, 2011, from Notepad++: http://notepad-plus-plus.org/

Holland, W. v. (1997). Assembly Features in Modelling and Planning. Winfried van Holland.

Holland, W. v., & Bronsvoort, W. F. (2000). Assembly features in modeling and planning. Robotics and Computer Integrated Manufacturing, 16, 277-294.

Hong, P., Doll, W. J., Nahm, A. Y., & Li, X. (2004). Knowledge sharing in integrated product development. European Journal of Innovation Management, 7(2), 102-112.

Hooff, B. v.-d., & Ridder, J. A. (2004). Knowledge sharing in context: the influence of organizational commitment, communication climate and CMC use on knowledge sharing. Journal of Knowledge Management, 8(6), 117-130.

Hounsell, M. d. (1998). Feature-Based Validation Reasoning For Intent Driven Engineering Design. Loughborough University, Wolfson School of Mechanical & Manufacturing Engineering. Loughborough: Marcelo da Silva Hounsell.

Huang, T.-T., Chen, L., & Stewart, R. A. (2010). The moderating effect of knowledge sharing on the relationship between manufacturing activities and business performance. Knowledge Management Research & Practice, 8, 285-306.

Hui, W., Dong, X., Guanghong, D., & Linxuan, Z. (2006). Assembly planning based on semantic modelling approach. Computers in Industry, 58, 227-239.

IEEE-Std-Computer-Dictionary. (1991). IEEE Standard Computer Dictionary A Compilation of IEEE Standard Computer Glossaries. In IEEE Std 610.

Ilgin, M. A., Gupta, S. M., & Nakashima, K. (2011). Coping with disassembly yield uncertainty in remanufacturing using sensor embedded products. Journal of Remanufacturing.

IODE. (2013). Highfleet Tools User Manual. Highfleet.

ISO 18629-43. (2006). Online Browsing Platform (OBP). Retrieved from http://www.iso.org/obp/ui/#search

ISO 19115. (2003). Online Browsing Platform (OBP). Retrieved 08 29, 2012, from http://www.iso.org/obp/ui/#search

ISO/IEC-24707. (2007). Information technology — Common Logic (CL): a framework for a family of logic based. Retrieved 08 14, 2011, from Common Logic Standard: http://standards.iso.org/ittf/licence.html

ISO/TC 10303-224. (2003). ISO/DIS 10303-224:2003(E) Document. ISO.

ISO/TS-10303-1022. (2004). Industrial automation systems and integration -- Product data representation and exchange -- Part 1022: Application module: Part and Version Identification.

ISO-18629. (2005). Industrial automation systems and integration - Process Specification Language (PSL).

Jean, S., Pierra, G., & Ait-Ameur, Y. (2006). DOMAIN ONTOLOGIES : A DATABASE-ORIENTED ANALYSIS. Proc. of Web Information Systems and Technologies. Setubal, Portugal.

Jensen, C. H., Helsel, J. D., & Espin, E. (2001). Interpreting Engineering Drawings (6th ed.). Delmar Cengage Learning.

Jiao, J., Tseng, M. M., Ma, Q., & Zou, Y. (2000). Generic Bill-of-Materials-and-Operations for High-Variety Production Management. Concurrent Engineering: Research and Applications, 8(4), 297-321.

Jones, A., Yih, Y., & Wallace, E. (2001). Monitoring and Controlling Operations. In G. Salvendy (Ed.), Handbook of Industrial Engineering (pp. 1768-1790). John Wiley & Sons, Inc.

KFL Reference. (2012). Ontologies in KFL. HIGHFLEET®.

Khondoker, R. M., & Mueller, P. (2010). Comparing Ontology Development Tools Based on an Online Survey. Proceedings of the World Congress on Engineering. London: WCE.

Kifer, M., Lausen, G., & Wu, J. (1995). Logical Foundation of Object Oriented and Frame Based Languages. Journal of Association for Computing Machinery, 42(4), 741-843.

Kim, K.-Y., Manley, D. G., & Yang, H. (2006). Ontology-based assembly design and information sharing for collaborative product development. Computer-Aided Design, 38, 1233–1250.

Kleppe, A., Warmer, J., & Bast, W. (2003). MDA EXPLAINED The Model Driven Architecture: Practice and Promise. PAPERBACK.

Komatsoulis, G. A., Warzel, D. B., Hartel, F. W., Shanbhag, K., Chilukuri, R., Fragoso, G., . . . Covitz, P. A. (2008). caCORE version 3: Implementation of a model driven, service-oriented architecture for semantic interoperability. Journal of Biomedical Informatics, 41, 106-123.

Koren, Y. (2010). The Global Manufacuring Revolution: Product-Process-Business Integration and Reconfigurable Systems. John Wiley & Sons.

Krause, F. L., Kimura, F., Kjellberg, T., & Lu, S. C. (1993). Product Modelling. Annals of the CIRP, 42(2), 695-706.

Krulikowski, A. (1998). Fundamentals of Geometric Dimensioning and Toleraning. NewYork: Delmar, adivision of Thomson Learning Inc.

Kryssanov, V. V., Abramov, V. A., Fukuda, Y., & Konishi, K. (2006). The meaning of manufacturing know-how. CoRR.

Kuffner, T. A., & Ullman, D. G. (1991). The information requests of mechanical design engineers. DESIGN STUDIES, 12(1), 42-50.

Lam, T. H., Lee, R. S., & Liu, J. N. (2008). An Ontology Based Intelligent Mobile System For Tourist Guidance. In J. Fulcher, & C. L. Jain (Eds.), Computational Intelligence: A Compendium (pp. 381-404). Berlin: Springer Verlag.

Lambert, D., Saulwick, A., Nowak, C., Oxenhan, M., & O'Dea, D. (2009). An Overview of Conceptual Frameworks. DSTO Defence Science and Technology Organisation, Command, Control, Communications and Intelligence Division, Department of Defence. Edinburgh: Commonwealth of Australia.

Lanz, M., Garcia , F., Kallela, T., & T, R. (2008). Product-Process Ontology For Managing Assembly Specific Knowledge Between Product Design and Assembly System Simulation. In S. K. Svetan Ratchev, Micro-Assembly Technologies and Applications (Vol. 260, pp. 99-108). Boston: Springer.

Lassila, O., & Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. Retrieved 07 26, 2013, from http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

Lee, C., Leem, C. S., & Hwang, I. (2011). PDM and ERP integration methodology using digital manufacturing to support global manufacturing. Int J Adv Manuf Technol, 399-409.

Lee, J.-N. (2001). The impact of knowledge sharing, organizational capability and partnership quality on IS outsourcing success. Information & Management, 38, 323-335.

Leila, Z.-G. (2009, November-December 29-04). Reference Ontology Presentation. Retrieved 08 26, 2013, from http://www.slideshare.net/ontoini/sitis-kare-09-reference-ontology-presentation

Lemaignan, S., Siadat, A., Dantan, J.-Y., & Semenenko, A. (2006). MASON: A Proposal for An Ontology of Manufacturing Dokmain. Proceedings of the IEEE Workshop on Distributed INtelligent Systems. IEEE.

Lenau, T., & Mu, L. (1993). Features in integrated modelling of products and their production. International Journal of Computer Integrated Manufacturing, 6(1), 65-73.

Lin, H. K., & Harding, J. K. (2007). A manufacturing system engineering ontology model on the semantic web for inter-enterprise collaboration. Computers in Industry, 58, 428-437.

Linn, R. J. (1997). Computer-aided Assembly Planning. In B. Wang, Integrated Product, Process and Enterprise Design (pp. 264-301). Chapman & Hall.

Lit, P. D., & Delchambre, A. (2003). Integrated design of a product family and its assembly system. Kluwer Academic Publishers.

Lohse, N. (2006). Towards an Ontology Framework For the Integrated Design of Modular Assembly Systems. University of Nottingham. Nottingham: University of Nottingham.

Lohse, N., Hirani , H., & Ratchev, S. (2006). Equipment Ontology For Modular Reconfigurable Assembly Systems. International Journal of Flexible Manufacturing System, 17, 301-314.

Lopez-Ortega, O., & Ramirez, M. (2005). A STEP-based manufacturing information system to share flexible manufacturing resources data. Journal of Intelligent Manufacturing, 16, 287-301.

LV, M.-Y., HOU, W.-J., & LI, X.-J. (2011). The Research of Hierarchy Assembly Semantic Model Based on Intelligent Assembly Process Planning. Key Engineering Materials, 467-469, 1933-1939.

Mantyla, M., Nau, D., & Shah, J. (1996, Feb). Challenges in Feature-Based Manufacturing Research. Communications of the ACM, 39(2), pp. 77-85.

Martin-Vega, L. A., Brown, H. K., Shaw, W. H., & Sanders,, T. J. (1995). Industrial Perspective on Research Needs and Opportunities in Manufacturing Assembly. Journal of Manufacturing Systems, 14(1), 45 - 58.

MDA-Guide-Document. (2003). MDA Guide Version 1.0. (J. Miller, & J. Mukerji, Eds.) OMG. Retrieved from http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf

Michel, J. J. (2005). Terminology extracted from some manufacturing and modelling related standards.

Mizoguchi, R. (2004). Tutorial on Ontological Engineering, Part 2: Ontology Development, Tools and Languages . New Generation Computing, 22, 61-96.

Mizoguchi, R., & Kozaki, K. (2009). Ontology Engineering Environments. In S. Staab, & R. Studder (Eds.), Handbook on Ontologies (pp. 315-335). Berlin: Springer-Verlag.

Molina, A., Ellis, T. I., Young, R. I., & Bell, R. (1995). Modelling Manufacturing Capability to Support Concurrent Engineering. Concurrent Engineering: Research and Applications, 3(1), 3-29.

Molloy, E., Yang, H., Browne, J., & Davies, B. J. (1991). Design for Assembly within Concurrent Engineering. CIRP Annals - Manufacturing Technology, 40(1), 107 - 110.

Molloy, O., Tilley, S., & Warman, E. A. (1998). Design For Manufacturing and Assembly. Chapman & Hall.

Mostefai, S., Bouras, A., & Batouche, M. (2006). Effective Collaboration in Product Development via a Common Sharable Ontology. International Journal of Computational Intelligence, 206-212.

Musen, M. A. (1992). Dimensions of Knowledge Sharing and Reuse. Computers and Biomedical Research, 25, 135-467.

Musen, M. A. (1998). Domain Ontologies in Software Engineering: Use of Protégé with the EON Architecture. Stanford, California.

Navigli, R., & Velardi, P. (2004). Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites. Computational Linguistics, 30(2), 152-179.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., & Swartout, W. R. (1991). Enabling Technology for KNowledge Sharing. AI Magzine, 12(3), 37-56.

Nemuraite, L., Ceponiene, L., & Vedricka, G. (2009). Representation of Business Rules in UML & OCL Models for Developing Information Models. In A.

P. Janis Stirna, The Practice of Enterprise Modeling: First IFIP WG 8.1 Working Conference, PoEM 2008. Laxenburg: IFIP International Federation For Information Processing.

Newman, S., Nassehi, A., Xu, X., Rosso, R., Wang, L., Yusof, Y., . . . Dhokia, V. (2008). Strategic Advantages of Interoperability for Global Manufacturing Using CNC Technology. Robotics and Computer Integrated Manufacturing, 24, 699-708.

Niles, I., & Pease, A. (2001). Towards a Standard Upper Ontology. Proceedings of the international conference on Formal Ontology in Information Systems (pp. 2-9). New York: ACM.

NIST. (1999). Interoperability Cost Analysis of the U.S. Automotive Supply Chain. North Carolina: NIST.

Nof, S. Y., Wilhelm, W. E., & Warnecke, H.-J. (1997). Industrial Assembly (First ed.). Chapman & Hall.

Noy, N. F., & McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology.

Oberle, D., Grimm, S., & Staab, S. (2009). An Ontology for Software. In Handbook on Ontologies, International Handbooks on Information Systems. Berlin: pringer-Verlag.

Oberle, D., Ankolekar, A., Hitzler, P., Cimiano, P., Sintek, M., Kiesel, M., . . . Zhou, J. (2007). DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). Journal of Web Semantics, 5, 156-174. doi:10.1016/j.websem.2007.06.002

Ouksel, A. M., & Sheth, S. (1999). Semantic interoperability in global information systems. SIGMOD Rec., 28(1), 5-12.

Oztemel, E., & Tekez, E. K. (2009). Integrating manufacturing systems through knowledge exchange protocols within an agent-based Knowledge Network. Robotics andComputer-IntegratedManufacturing, 25, 235-245.

Palmer, C., Chungoora, N., Young, R., Gunendran, A. G., Usman, Z., Case, K., & Harding, J. A. (2012). Exploiting unified modelling language (UML) as a preliminary design tool for Common Logic-based ontologies in manufacturing. International Journal of Computer Integrated Manufacturing, 1-17.

Pan, Z. J. (2009). Resource Description Framework. In S. Staab, & R. Studer (Eds.), Handbook on Ontologies (pp. 71-90). Berlin Heidelberg: Springer-Verlag.

Panetto, H. (2007). Towards a Classification Framework for Interoperability of Enterprise Applications. International Journal of Computer Integrated Manufacturing, 20(8), 727-740.

Park, J., & Simpson, T. W. (2008). Toward an activity-based costing system for product families and product platforms in the early stages of development. International Journal of Production Research, 46(1), 99-130.

Polovina, S., Cook, J., & Loke, J. (2009). A Practical Exploration of Ontology Interoperability. In F. D. Sebastian Rudolph, Conceptual Structures: Leveraging Semantic Technologies: 17th International Conference on Conceptual Structures, ICS 2009 (pp. 247-256). Moscow, Russia: Springer-Verlag.

Pratt, M. J. (2001). Introduction to ISO 10303 - the STEP Standard for Product Data Exchange. TECHNICAL NOTE. NIST.

Pratt, M. J., & Wilson, P. R. (1985). Requirements for support of form features in a solid modelling system. Texas, USA: Computer Aided Manufacturing-International.

Ray, S. R., & Jones, A. T. (2006). Manufacturing Interoperability. Journal of Intelligent Manufacturing, 17, 681-688.

Rekiek, B., Falkenauer, E., & Delchambre, A. (1997). Multi-Product Resource Planning. Proceedings of The 1997 IEEE International Symposium on Assembly and Task Planning, (pp. 115-121). California.

Riege, A. (2005). Three-dozen knowledge-sharing barriers managers must consider. JOURNAL OF KNOWLEDGE MANAGEMENT, 9(3), 18-35.

Roller, D. (1989). Design by Features: An Approach to High Level Shape Manipulation. Computers in Industry, 12(3), 185-191.

Rosen, D. W. (1993). Feature-Based Design: Four Hypotheses for Future CAD Systems. Research in Engineering Design, 5, 125-139.

Saaksvuori, A., & Immonen, A. (2008). Product Lifecycle Management (3rd ed.). Berlin Heidelberg: Springer-Verlag.

Sabou, M., Wroe, C., Goble, C., & Stuckenschmidt, H. (2005). Learning Domain Ontologies for Semantic Web Service Descriptions. Journal of Web Semantics, 3(4), 340-365.

Sackett, P. J., & Holbrook, A. E. (1988). DFA as a primary process decreases design deficiencies. Assembly Automation, 8(3), 137-140.

Sanchez-Alonso, S., & Garcia-Barriocanal, E. (2006). Making use of upper ontologies to foster interoperability between SKOS concept schemes. Online Information Review, 30(3), 253-277.

Sánchez-Ruíz, A., Umapathy, K., & Hayes, P. (2009). Toward Generic, Immersive, and Collaborative Solutions to the Data Interoperability Problem which Target End-Users. Journal of Computing Science and Engineering, 3(2), 127-141.

Schreiber, G., Wielinga, B., & Jansweijer, W. (1995). The Kactus View on the 'O' Word. In the proceeding of IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal: Academic Press.

SCRA. (2006). STEP APPLICATION HANDBOOK ISO 10303 VERSION 3. SCRA.

Sehgal, V. (2009). Enterprise Supply Chain Management: Integrating Best-in-Class Processes. John Wiley & Sons.

Semere , D. T., Dilshad , S., & Lindberg, B. (2007). Machining Ontology and Knolwedge Modelling. Royal Institute of Technology, Institute of Industrial Production.

Sengupta, K., & Hitzler, P. (2013). Web Ontology Language (OWL). Wright State University, Joshi Research Center, Dayton. Retrieved 07 30, 2013, from http://knoesis.wright.edu/pascal/pub/owl-encyc-13.pdf

Shah, J. J. (2001). Designing with Parametric CAD: Classification and Comparison of Construction Techniques. In F. Kimura (Ed.), Geometric Modelling: Theoretical and Computational Basis Towards Advanced CAD Applications (pp. 53-68). International Federation for Information Processing.

Stark, J. (2011). Product Lifecyle Management: 21st Century Paradigm for Product Realisation . Springer-Verlag.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge Engineering: Principles and Methods. Data & Knowledge Engineering, 25, 161-197.

Su, X., & Ilebrekke, L. (2002). A Comparative Study of Ontology Languages and Tools. CAiSE 14th International Conference in Advanced Information System Engineering . London: Springer Verlag.

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., & Wenke, D. (2002). OntoEdit: Collaborative Ontology Development for the Semantic Web. First International Semantic Web Conference, Lecture Notes in Computer Science. 2342, pp. 221-235. Berlin: Springer.

Taylor, D. (2005). Machine Trades Bluprint Reading (2nd Edition ed.). Thomson Delmar Learning.

True, M., & Izzi, C. (2002). Collaborative Product Commerce: Creating Value Across the Enterprise. White Paper.

Tsui, L., Chapman, S. A., Schnirer, L., & Stewart, S. (2006). A Handbook on Knowledge Sharing: Strategies and Recommendations for Researchers, Policymakers, and Service Providers.

Tursi, A., Panetto, H., Morel, G., & Dassisti, M. (2009). Ontological approach for products-centric information system interoperability in networked manufacturing enterprises. Annual Reviews in Control, 33(2), 238-245.

Uschold, M., & Gruninger, M. (2004). Ontologies and Semantics for Seamless Connectivity. SIGMOD Rec., 33(4), 58-64.

Uschold, M., & King, M. (1995). Towards e methodology for building ontologies. In Workshop on Basic Ontological Issues in Knowledge Sharing.

Usman, Z. (2012). A Manufacturing Core Concepts Ontology to Support Knowledge Sharing. PhD Thesis, Loughborough.

Usman, Z., Young, R., Chungoora, N., Palmer, C., Case, K., & Harding, J. (2013). Towards a formal manufacturing reference ontology. International Journal of Production Research. doi:http://dx.doi.org/10.1080/00207543.2013.801570

Vollmann, T. E. (1997). Manufacturing Planning and Control Systems (4th ed.).

W3C. (2004). OWL Web Ontology Language Overview. (D. L. McGuinness, & F. v. Harmelen, Editors) Retrieved 07 30, 2013, from http://www.w3.org/TR/owl-features/

W3C Website. (2006). Retrieved from W3C: http://www.w3.org/

Wache, H., Vogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hubner, S. (2001). Ontology-Based Integration of Information-A Survey of Existing Approaches. IJCAI--01 Workshop: Ontologies and Information Sharing . Retrieved from http://www.citeulike.org/user/Salma/article/593500

Walker, J. (2001). Manual and Automated Assembly. In K. Zandin, Maynard's industrial engineering handbook (pp. 14.61-14.89). McGraw-Hill.

Wang, K., & Tong, S. (2008). An Ontology of Manufacturing Knowledge for Design Decision Support. Wireless Communications, Networking and Mobile Computing.

Wei, S., Qin-Yi, M., & Tian-Yi, G. (2009). An Ontology-Based Manufacturing Design System. Information Technology Journal, 8(5), 643-656.

Whitney, D. E. (2004). Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development. New York, Oxford: OXFORD UNIVERSITY PRESS.

Woodley, M. S. (2005, 11 07). DCMI Gloassary. (The Dublin Core® Metadata Initiative) Retrieved 02 28, 2011, from http://dublincore.org/documents/usageguide/glossary.shtml

Xu, H., Xu, X., & He, T. (2008). Research on Transformation Engineering BOM into Manufacturing BOM Based on BOP. Applied Mechanics and Materials, 10-12, 99-103.

Xu, X. (2009). Integrating Advanced Computer-Aided Design, Manufacturing, and Numerical Control: Principles and Implementations. Hershey: IGI Global.

Young, B., Cutting-Decelle, A.-F., Guerra, D., Das, B., & Cochrane, S. (2005). Sharing Manufacturing Information and Knowledge in Design Decision and Support. In A. Bramley, D. Brissaud, D. Coutellier, & C. McMahon (Eds.), Advances in Integrated Design and Manufacturing in Mechanical Engineering (pp. 173-185). Springer.

Young, R., Chungoora, N., Usman, Z., Anjum, N., Gunendran, G., Palmer, C., . . . Cutting-Decelle, A. F. (2010). An Exploration of Foundation Ontologies and Verification Methods for Manufacturing Knowledge Sharing. International Conference on Interoperability of Enterprise Software and Applications: Proceedings of the Workshops and the Doctoral Symposium of the I-ESA International Conference (pp. 83-93). Wiley & Sons Inc.

Young, R., Gunendran, A. G., Cutting-Decelle, A. F., & Gruninger, M. (2007). Manufacturing knowledge sharing in PLM: a progression towards the use of heavy weight ontologies. International Journal of Production Research, 45(1), 1505-1519.

Young, R., Gunendran, G., Chungoora, N., Harding, J., & Case, K. (2009). Enabling interoperable manufacturing knowledge sharing in PLM. International conference on Product Lifecycle Management. Inderscience Enterprises Ltd.

Zeng, H., & Bin, H. (2004). A method calculation of materials requirements based on BOP in assembly production planning. In X. Y. Shao, & C. Deng (Ed.), International Conference on Manufacturing Automation. Professional Engineering Publishing Limited.

Zha, X. F., & Du, H. (2002). A PDES/STEP-based model and system for concurrent integrated design and assembly planning. Computer Aided Design, 34, 1087-1110.

Zhang, Y., Mo, R., Shi, Y., & Chang, Z. (2010). A Product Manufacturability Reverse Mapping Method Based on BOM Transformation. International Conference on Computer Application and System Modeling. IEEE.

Zhao, J., Cheung, W. M., & Young, R. I. (1999). A consistent manufacturing data model to support virtual enterprises. International Journal of Agile Management Systems, 1(3), 150-158.

Zhong, L.-w., & NI, J. (2010). A Study of BOM Automatic Creation Based on Products. International Conference on Mechanic Automation and Control Engineering (MACE). IEEE. doi:10.1109/MACE.2010.5535565

Zhou, J., & Dieng-Kuntz, R. (2004). Manufacturing Ontology Analysis and Design: Towards Excellent Manufacturing. 2nd IEEE International Conference on Industrial Informatics. IEEE.

Zhu, H., & Madnick, S. (2006). A Lightweight Ontology Approach to Scalable Interoperability. MIT Sloan Working Paper, 4621-06.

Zobolas, G. I., Tarantilis, C. D., & Ioannou, G. (2008). Extending capacity planning by positive lead times and optional overtime, earliness and tardiness for effective master production scheduling. International Journal of Production Research, 46(12), 3359-3356.

APPENDICES

RESEARCH TOOLS AND LANGUAGES

This chapter discusses the ontology modelling tools and languages used in this research work. The following tools and languages have been used for the development of ontologies.

- Enterprise Architect (EA) and UML
- IODE and Notepad++
- Knowledge Frame Language (KFL)

These tools and languages are discussed in detail in the following sections.

A.1 Enterprise Architect (EA) and UML

A.1.1 Introduction

Enterprise Architect (Sparx-Systems, 2011) is a UML (Unified Modelling Language) based modelling tool which can be used for design and construction of software systems. Designers and engineers can use Enterprise Architect (EA) for business process modelling or for general modelling purpose e.g. visualisation of existing processes and systems. EA includes all aspects of software development cycle which are: requirement gathering, analysis, model design, testing, change control, maintenance, implementation and traceability.

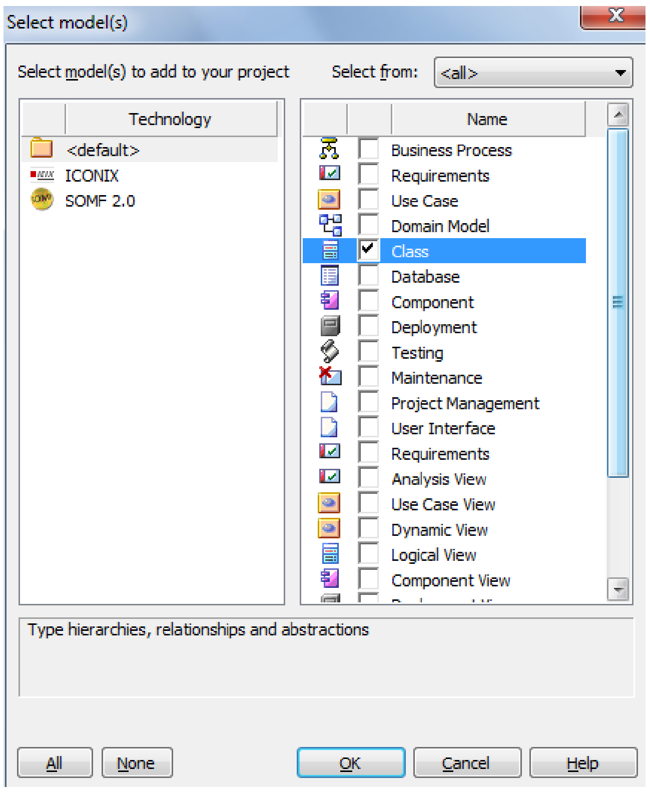
EA is widely used modelling tool as the company (Sparx Systems) claims that over 200,000 licenses have been sold and thousands of companies (ranging from multinational to small companies) are using EA for design and modelling of their systems. The company also claims that EA has become a default UML modelling tool for developers, analysts and consultants in more than 130 countries across the globe. Some of the key benefits and features of EA include: built in UML2.3 which supports all UML diagrams, modelling of class hierarchies, version control, low price, high speed, improved scalability and high usability.

Before we discuss the application of UML in the current research, it is good to know a little bit about UML. Unified Modelling Language or UML was developed by Rational Corp., originally with the contribution of Grady Booch, James Rambaugh and Ivars Jacobson in the early 1990s (Hunt, 2003) (Siau, 2001). UML was recognised as a standard by Object Management Group (OMG) in 1997 and now widely adopted as blueprints for various softwares (Booch, 1999). UML is precisely defined as “a family of graphical notations, backed by single metamodel, that help in describing and designing software systems, particularly software systems built using the Object-Oriented (OO) style” (Flower, 2003). UML has widespread application in various fields and it has been successfully applied to develop systems in the fields of e-commerce, computer games, command and control, banking, insurance, medical electronics, telephony, robotics and avionics (Booch, 1999).

As Flower (2003) supports the fact that UML has not been fully understood or used even by its developers so here a portion of UML related to this research has been discussed. The ARO has been developed using the class and class relationship functionality of EA which is discussed as follows.

A.1.2 Building Class Diagrams

A class diagram is used to represent objects and their inter-relationships relationships. Different ARO concepts e.g. product, component, and assembly feature can be represented by classes in UML. Hence the first step in EA to is to select a class model from the given options as shown in the figure A.1. A user can click on class icon and press ok to go ahead.



Select model(s)

Select model(s) to add to your project   Select from: <all>

Technology
<default>
ICONIX
SOMF 2.0

	Name
☐	Business Process
☐	Requirements
☐	Use Case
☐	Domain Model
☑	Class
☐	Database
☐	Component
☐	Deployment
☐	Testing
☐	Maintenance
☐	Project Management
☐	User Interface
☐	Requirements
☐	Analysis View
☐	Use Case View
☐	Dynamic View
☐	Logical View
☐	Component View

Type hierarchies, relationships and abstractions

Figure A.1: An overview of various modelling options available in Enterprise Architect

Once the class model is selected, the user will see a model bar (shown at the right side of the figure A.2). If we double click on the system icon on modelling bar, EA will take us to class diagram modelling environment. This is also shown in the figure A.2. A designer can drag the class icon (by holding the left mouse click) to the modelling area and can generate a new class. Similarly, we can also drag the class relationships (shown on left bottom side of the figure) to the modelling area to relate classes with each other. The class name can be renamed by simply double clicking the class icon in the modelling area. A window will pop up where we can change the name of the class.

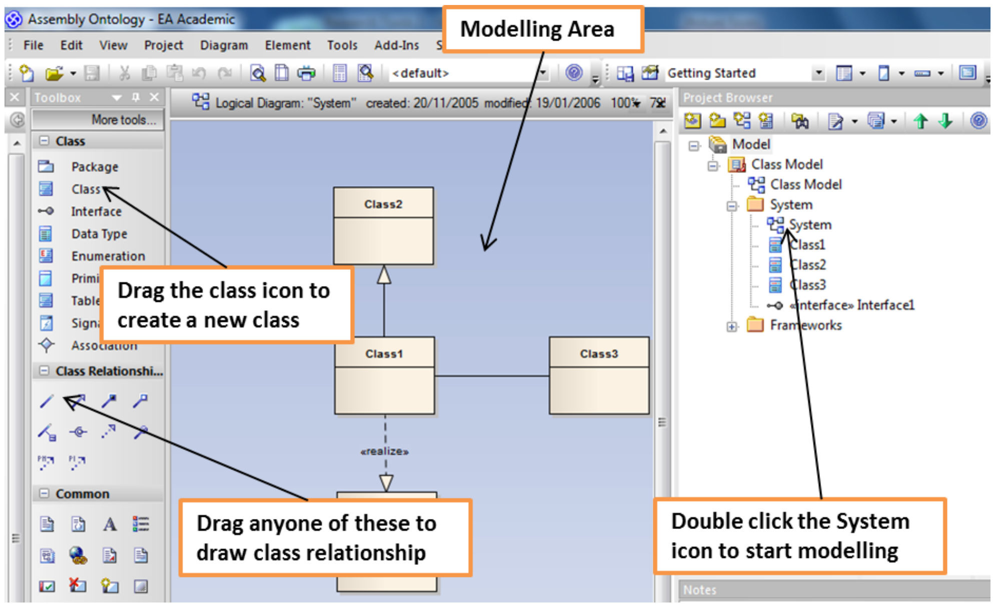
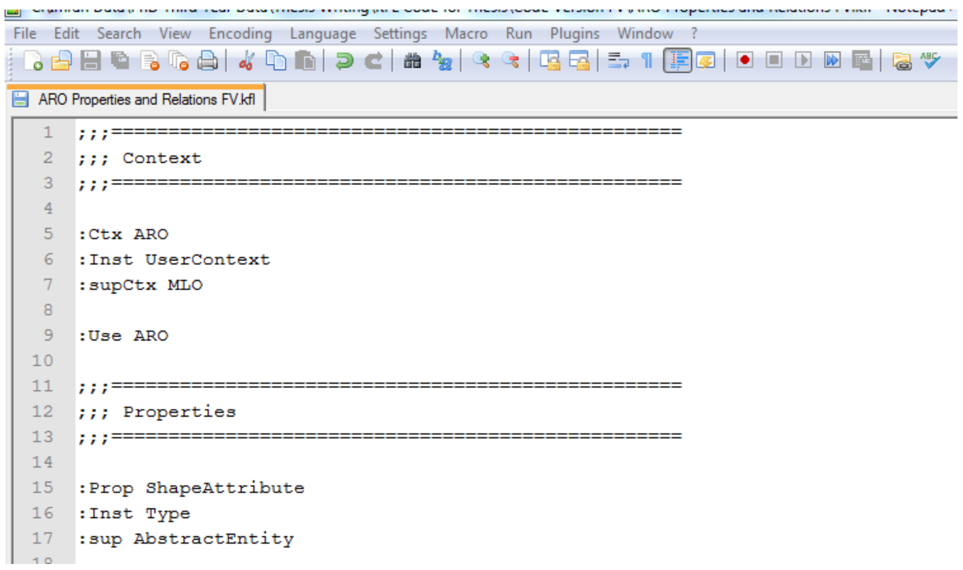

Assembly Ontology - EA Academic
File Edit View Project Diagram Element Tools Add-Ins
Modelling Area
Toolbox
Class
Package
Class
Interface
Data Type
Enumeration
Class Relationships
Common
Logical Diagram: "System" created: 20/11/2005 modified: 19/01/2006 100%
Class2
Class1
Class3
«realize»
Project Browser
Model
Class Model
System
Class1
Class2
Class3
«interface» Interface1
Frameworks
Drag the class icon to create a new class
Drag anyone of these to draw class relationship
Double click the System icon to start modelling

Figure A.2: Enterprise Architect class diagram environment

A.2 IODE and Notepad++

IODE; an acronym for Integrated Ontology Development Environment, is an ontology development tool developed by Highfleet. IODE provides a powerful expressive logic, model validation, a library for ontological content, and tools to support visualisation and sample data testing (HIGHFLEET, 2012). IODE uses Knowledge Framework Language (KFL) for writing code for ontologies. IODE has the capability to develop heavyweight Common-Logic based ontologies and Knowledge Bases (KBs). However unlike other ontological tools e.g. Protégé, it allows to write ontology codes outside the ontological environment (Chungoora, 2010).

Notepad++ a free source code editor (Ho, 2011), has been used to write KFL code. KFL files contain all the ontology code which is then implemented in IODE. A view of the Notepad++ is shown in figure A.3.



```
;;;==================================================

;;; Context
;;;==================================================

:Ctx ARO
:Inst UserContext
:supCtx MLO

:Use ARO

;;;==================================================
;;; Properties
;;;==================================================

:Prop ShapeAttribute
:Inst Type
:sup AbstractEntity
```

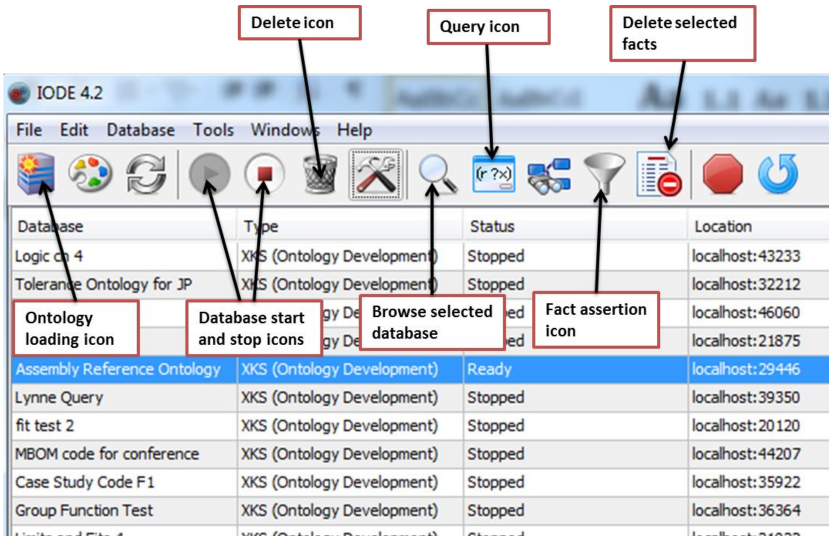
Figure A.3: A view of Notepad++ for writing KFL code

After writing coding in Notepad++, it is saved as file with .kfl extension. Then it is loaded into the IODE. The main components of IODE are shown in figure A.4. The icon shown on the top left corner of figure A.4 is used to load ontology files.

We can easily create, start, stop, delete, and configure various databases using the options shown in the figure. By hitting the browse database option one can explore all the internal structure of his ontology. Query tools help to show the already created knowledge and last two icons can be used for asserting facts or instances in system. The start and stop icons are used to start and stop the loaded databases. When a new database is added in IODE, it is by default in the start mode. However a database can be stopped by using the stop icon.

The other icons shown on the right side of the start icon will only be active if the database is in the “start” mode as shown in the figure. The delete icon is used to delete the loaded databases whereas the browse icon is used to explore the loaded ontology. For instance, by using the browse icon we can see ontology classes, relationships and axioms etc.

The query icon shown in figure A.4 is used to build queries whereas the fact assertion tool is used to populate facts into the database. The asserted facts can be deleted by stopping the database however IODE deletes all the asserted facts if the database is stopped by a user. Sometimes it is required to delete selected facts and this can be done by using the delete facts icon as shown in figure A.4.



Database	Type	Status	Location
Logic ch 4	XKS (Ontology Development)	Stopped	localhost:43233
Tolerance Ontology for JP	XKS (Ontology Development)	Stopped	localhost:32212
[illegible]	gy De	ed	localhost:46060
[illegible]	gy De	ed	localhost:21875
Assembly Reference Ontology	XKS (Ontology Development)	Ready	localhost:29446
Lynne Query	XKS (Ontology Development)	Stopped	localhost:39350
fit test 2	XKS (Ontology Development)	Stopped	localhost:20120
MBOM code for conference	XKS (Ontology Development)	Stopped	localhost:44207
Case Study Code F1	XKS (Ontology Development)	Stopped	localhost:35922
Group Function Test	XKS (Ontology Development)	Stopped	localhost:36364

Figure A.4: A view of IODE showing key options available in the tool

A.3 Knowledge Frame Language (KFL)

Knowledge Frame Language (KFL) is a Common Logic based ontology development language which has been used for this research work. KFL uses directives to specify the ontology code whereas each directive starts with colons followed by a keyword and certain arguments. In order to understand KFL, following concepts need to be understood.

- Contexts
- Properties
- Relations

- Functions
- Logic, Rules and Integrity

These are explained in the following sections.

A.3.1 Contexts

Contexts are very important in creating ontologies as they define specific point of view. Although we can create ontologies in IODE with a predefined context called Middle Level Ontology (MLO) however it is more convenient to define one's own context. We can create new context by writing a simple code in KFL as shown below. The first three fields (Ctx, Inst, supCtx) are mandatory for defining a new context in ontology while the last two are optional.

```
:Ctx ARO

:Inst UserContext

:supCtx MLO

:name "Assembly Reference Ontology"

:rem "The ARO context is used for the assembly domain"
```

Ctx (stands for context) defines the name of new concept (ARO in current case). We use UserContext as an instance of new context rather than SystemContext because we are defining our own context. The third field "supCtx" stands for super context of and we place MLO as super context of the ARO because it was default context. In the last two fields: name and rem (remarks), we can write anything within the inverted commas to elaborate the context code.

Once the context is defined, then we can use this context by writing the use directive as shown below.

```
:Use ARO
```

A.3.2 Properties

The term property refers to any taxonomic component while building an ontology in IODE. The term property is sometimes called class or category in other ontology development environments. Any concept/term is first represented as property in IODE and then relations, functions and logics are applied. When writing properties in KFL format, a user needs to write the following directives.

```
:Prop HandlingAF

:Inst Type

:sup AssemblyFeature

:name "Handling Assembly Feature"

:rem "Handling AF is used in resource evaluation"
```

The first three directives in the above code are mandatory while others are optional. Prop refers to property and represent the main concept e.g. HandlingAF in this case. Inst stands for “instance of” and represents property type. It has two kinds of properties in Upper Level Ontology (ULO) which are (1) Type and (2) MaterialRole. Types correspond to properties which do not change with the passage of time while MaterialRoles can change their status after sometime. In this work all the properties are instantiated under Type.

The third directive “sup” refers to super property relation and defines hierarchies of properties by this relation. For a property x to be super property of y, every instance of y should be an instance of x. For example an assembly feature is super property of handling AF as every instance of handling AF is an instance of the assembly feature.

The fourth and fifth directives are optional for properties where names and remarks (rem) define additional information related to the property as shown in the above example.

A.3.3 Relations

Properties are held together with the help of relationships. The sup property only defines hierarchy of properties and does not account for other relationships. A relation declaration consists of following directives.

```
:Rel hasTolerance

:Inst BinaryRel

:Sig Object Tolerance

Args “Assembly Feature” “Tolerance”
```

Like properties, the first three directives are compulsory for relations. The :Rel line describes the wording of relationship e.g. what relation a property has with the other one. :Inst directive defines the kind of relation depending upon the arity (number of arguments) of the relation. For example, BinaryRel (binary relation) in the above example connect two properties (Object and Tolerance) with each other. There are also other types of relations which are

- UnaryRel (one arguement)
- TernaryRel (three arguments)
- QuaternaryRel (four arguments)
- QuinaryRel (five arguments)
- Relation (Any number of arguments)

The :Sig directive should have a property for every argument position e.g. Object and Tolerance are two properties which have “hasTolerance” relationship.

A.3.4 Functions

Functions provide additional entities from one or more parameters and semantically differentiate between a description and what is described. Functions also allow parameters to be used for reasoning. Entities like fivePointTwoGrams or threePointTwoCentimetres would further complicate the model and make the parameters vague and unclear. KFL describes function in the following format to avoid the issues discussed above.

```
:Fun cm

:Inst UnaryFun

:Inst MeasureFun

:Sig RealNumber -> LengthDimension
```

This allows writing functions that describe length dimensions e.g. (cm 3.2). Similar to properties and relations, functions have three required directives which are Fun, Inst and Sig. It is also pertinent to note that except the first directive (Fun), the difference between a relation and function is the arrow in the :Sig directive. The text on the left side of arrow represents arguments to the function and text on the right side describes property instantiated by the function. Finally similar to relations, functions can also be classified by arity as follows.

- UnaryFunl (one arguement)
- BinaryFun (two arguments)
- TernaryFun (three arguments)
- QuaternaryFun (four arguments)
- QuinaryFun (five arguments)

A.3.5 Logics

Logics in KFL consist of constraints. These rules and constraints are the mandatory part of heavyweight ontologies and thus differentiate the latter from the lightweight ontologies. Rules help to infer the existing information and create new information while constraints pre-empt any data inconsistency. Constraints in turn enhance the data quality and speed up the query response times. Rules and constraints are extensively used in this research along with properties, relations, and functions to represent assembly knowledge.

A.3.5.1 Rules

As described above, rules infer new information from the existing statements and use implications. The implication operator is made up of an arrow and an equal size (as shown in the rule below). A rule consists of the antecedent and the consequent. The antecedent/s is/are the clause/clauses which help to infer new information. The consequent is the conclusive statement or the statement which is inferred using antecedents.

In KFL, the consequent will only be deduced if all the antecedent statements are true. For example in the following rule, the conclusive statement ((hasMinAllowableDimension ?p ?q+lower) will only be true if all the antecedent statements are ture.

```
(<= (hasMinAllowableDimension ?p ?q+lower)
        (and (Object ?p)
        (hasDimensionWithTolerance ?p ?q (?tol ?lower ?upper))
                 (measurePlus ?q ?lower ?q+lower)
                 (or (= ?tol tolerance)
                         (= ?tol f7)
                         (= ?tol H8)
                         (= ?tol k6)
```

```
                             (= ?tol H7)
                             (= ?tol p6))))
:rem "The Object ?p has a minimum allowable dimension which is
equivalent to its nominal dimension plus its lower deviation."
```

The above mentioned rule has also used conjunction and disjunction operators. The conjunction operator “and” combines two or more than two clauses to form the argument. The conjunction operator is true when all the clauses are true. The disjunction operator “or” combines two or more clauses and is true when at least one of the clauses is true.

A.3.5.2 Integrity Constraints (ICs)

Integrity Constraints (ICs) seem like rules apart from the fact that the IC directive starts with :IC. The IC directive represents the strength of the constraint and it shows the error messages in case of violation of constraints. The strength of IC can be categorized as

- Weak ICs
- Soft ICs
- Hard ICs
- Adamant ICs

A weak IC only shows an irregularity and does not indicate any problem. Soft IC is stronger than weak IC however it does not rollback a transaction. Hard IC is stronger than both of weak and soft ICs and could rollback a transaction. Adamant IC is strongest of all ICs and any violation of adamant IC could harm the integrity of logic engine. An example of hard integrity constrain is given below.

```
      (=> (hasTolerance ?x (tolerance ?q1 ?q2))
          (and (Object ?x)

                (Dimension ?q1)

                (Dimension ?q2)

                (measureLT ?q1 ?q2)))

    :IC hard "The lower deviation quantity of a Tolerance must
    always be less than its upper deviation quantity."
```

The above mentioned IC will prevent any attempt to assert first quantity of tolerance which is larger than the second quantity of tolerance.

A.4 References

Booch, G. (1999, October). UML in action. COMMUNICATIONS OF THE ACM, 54(9), pp. 26-28.

Chungoora, N. (2010). A Framework to Support Semantic Interoperability in Product Design and Manufacture. Loughborough: Nitishal Chungoora.

Flower, M. (2003). UML Distilled, Third Edition, A Brief Guide to the Standard Object Modelling Language. Addison-Wesley Professional.

HIGHFLEET. (2012). Integrated Ontology Development Environment - IODE™. (HIGHFLEET) Retrieved 08 22, 2012, from HIGHFLEET: http://www.highfleet.com/iode.html

Ho, D. (2011). Notepad++. Retrieved 08 22, 2011, from Notepad++: http://notepad-plus-plus.org/

Hunt, J. (2003). Guide to the unified process featuring UML, Java and design patterns. London: Springer-Verlag.

Siau, K. (2001). Rational Unified Process and Unified Modelling Language, A GOM Analysis. In T. A. Keng Siau, Unified modeling language: systems analysis, design and development issues (pp. 107-115). Idea Group Publishing.

Sparx-Systems. (2011). Enterprise Architect. (Sparx Systems Pty Ltd.) Retrieved 08 22, 2011, from Enterprise Architect: http://www.sparxsystems.com/

FORMALIZATION OF ARO AND RELATED CONCEPTS

B.1 Formalization of ARO

```
;;;==================================================
;;; Context
;;;==================================================

:Ctx ARO
:Inst UserContext
:supCtx MLO

:Use ARO

;;;==================================================
;;; Properties
;;;==================================================

:Prop ShapeAttribute
:Inst Type
:sup AbstractEntity

:Prop Polygon
:Inst Type
:sup ShapeAttribute

:Prop NonPolygon
:Inst Type
:sup ShapeAttribute

:Prop Triangular
:Inst Type
:sup Polygon

:Prop Square
:Inst Type
:sup Polygon
```

```
:Prop Pentagon
:Inst Type
:sup Polygon

:Prop Hexagon
:Inst Type
:sup Polygon

:Prop OtherPolygon
:Inst Type
:sup Polygon

:Prop Circular
:Inst Type
:sup NonPolygon

:Prop NonCircular
:Inst Type
:sup NonPolygon

:Prop DesignFunction
:Inst Type
:sup AbstractEntity

:Prop ToleranceType
:Inst Type
:sup AbstractEntity

:Prop Dimension
:Inst Type
:sup Quantity

:Prop LengthDimension
:Inst Type
:sup Dimension

:Prop AngularDimension
:Inst Type
:sup Dimension
```

```
:Prop Tolerance
:Inst Type
:sup Dimension

:Prop SpatialLocation
:Inst Type
:sup ConcreteEntity

:Prop PointLocation
:Inst Type
:sup SpatialLocation

:Prop AngularLocation
:Inst Type
:sup SpatialLocation

:Prop Step
:Inst Type
:sup Event

:Prop Process
:Inst Type
:sup Event

:Prop ManufacturingProcess
:Inst Type
:sup Process

:Prop AssemblyProcess
:Inst Type
:sup ManufacturingProcess

:Prop Operation
:Inst Type
:sup Event

:Prop ManufacturingOperation
:Inst Type
:sup Operation
```

```
:Prop AssemblyOperation
:Inst Type
:sup ManufacturingOperation

:Prop Product
:Inst Type
:sup Object

:Prop ProductVersion
:Inst Type
:sup Object

:Prop BOM
:Inst Type
:sup Object

:Prop EBOM
:Inst Type
:sup BOM

:Prop MBOM
:Inst Type
:sup BOM

:Prop BOP
:Inst Type
:sup Event

:Prop BOR
:Inst Type
:sup Object

:Prop Family
:Inst Type
:sup Object

:Prop ProductFamily
:Inst Type
:sup Family
```

```
:Prop DesignProductFamily
:Inst Type
:sup ProductFamily

:Prop ManufacturingProductFamily
:Inst Type
:sup ProductFamily

:Prop Component
:Inst Type
:sup Object

:Prop Part
:Inst Type
:sup Component

:Prop Subassembly
:Inst Type
:sup Component

:Prop AssemblyComponent
:Inst Type
:sup Component

:Prop ADAssemblyComponent
:Inst Type
:sup AssemblyComponent

:Prop ARAssemblyComponent
:Inst Type
:sup AssemblyComponent

:Prop Material
:Inst Type
:sup Object

:Prop AuxiliaryMaterial
:Inst Type
:sup Material
```

```
:Prop Feature
:Inst Type
:sup Object

:Prop FormFeature
:Inst Type
:sup Feature

:Prop ProductFeature
:Inst Type
:sup FormFeature

:Prop ResourceFeature
:Inst Type
:sup FormFeature

:Prop SinglePiecePartFeature
:Inst Type
:sup ProductFeature

:Prop AssemblyFeature
:Inst Type
:sup ProductFeature

:Prop HoleAF
:Inst Type
:sup AssemblyFeature

:Prop ShaftAF
:Inst Type
:sup AssemblyFeature

:Prop PlaneMateAF
:Inst Type
:sup AssemblyFeature

:Prop AlignmentAF
:Inst Type
:sup AssemblyFeature
```

```
:Prop HandlingAF
:Inst Type
:sup AssemblyFeature

:Prop ToolingAF
:Inst Type
:sup AssemblyFeature

:Prop ManufacturingResourceFeature
:Inst Type
:sup ResourceFeature

:Prop AssemblyResourceFeature
:Inst Type
:sup ManufacturingResourceFeature

:Prop HoleARF
:Inst Type
:sup AssemblyResourceFeature

:Prop ShaftARF
:Inst Type
:sup AssemblyResourceFeature

:Prop HandlingARF
:Inst Type
:sup AssemblyResourceFeature

:Prop ToolingARF
:Inst Type
:sup AssemblyResourceFeature

:Prop Resource
:Inst Type
:sup Object

:Prop ManufacturingResource
:Inst Type
:sup Resource
```

```
:Prop AssemblyResource
:Inst Type
:sup ManufacturingResource

:Prop ManufacturingFacility
:Inst Type
:sup Object

:Prop Enterprise
:Inst Type
:sup ManufacturingFacility

:Prop Factory
:Inst Type
:sup ManufacturingFacility

:Prop Shop
:Inst Type
:sup ManufacturingFacility

:Prop Cell
:Inst Type
:sup ManufacturingFacility

:Prop Station
:Inst Type
:sup ManufacturingFacility

:Prop ComponentList
:Inst Type
:sup List

:Prop AssemblyComponentList
:Inst Type
:sup ComponentList

:Prop ADAssemblyComponentList
:Inst Type
:sup AssemblyComponentList
```

```
:Prop ARAssemblyComponentList
:Inst Type
:sup AssemblyComponentList

:Prop AuxiliaryMaterialList
:Inst Type
:sup List

;;;==================================================
;;; Relations
;;;==================================================

:Rel hasDimension
:Inst BinaryRel
:Sig Object Dimension

:Rel hasTolerance
:Inst BinaryRel
:Sig Object Tolerance

:Rel hasToleranceType
:Inst BinaryRel
:Sig Object ToleranceType

:Rel hasDimensionWithTolerance
:Inst TernaryRel
:Inst IntensionalRel
:Sig Object Dimension Tolerance

:Rel matesWith
:Inst BinaryRel
:Inst IrreflexiveBR
:Sig Object Object

:Rel hasFitWith
:Inst BinaryRel
:Inst IrreflexiveBR
:Inst IntensionalRel
:Sig Object Object
```

```
:Rel hasMinAllowableDimension
:Inst BinaryRel
:Inst IntensionalRel
:Sig Object Dimension


:Rel hasMaxAllowableDimension
:Inst BinaryRel
:Inst IntensionalRel
:Sig Object Dimension

:Rel hasAllowableDimensionalValueOrRange
:Inst BinaryRel
:Inst IntensionalRel
:Sig Object Top

:Rel hasClearanceFitWith
:Inst BinaryRel
:Inst IrreflexiveBR
:Inst IntensionalRel
:Sig Object Object
:supRel hasFitWith

:Rel hasInterferenceFitWith
:Inst BinaryRel
:Inst IrreflexiveBR
:Inst IntensionalRel
:Sig Object Object
:supRel hasFitWith

:Rel hasTransitionFitWith
:Inst BinaryRel
:Inst IrreflexiveBR
:Inst IntensionalRel
:Sig Object Object
:supRel hasFitWith


:Rel hasAllowanceWith
```

```
:Inst TernaryRel
:Inst IntensionalRel
:Sig Object Object Dimension


:Rel hasShapeAttribute
:Inst BinaryRel
:Sig Object ShapeAttribute

:Rel hasLocation
:Inst BinaryRel
:Sig Object SpatialLocation

:Rel hasPointLocation
:Inst BinaryRel
:Sig Object PointLocation
:supRel hasLocation


:Rel hasAngularLocation
:Inst BinaryRel
:Sig Object AngularLocation
:supRel hasLocation

:Rel hasBOM
:Inst BinaryRel
:Sig Object BOM

:Rel hasBOP
:Inst BinaryRel
:Sig Object BOP

:Rel hasBOR
:Inst BinaryRel
:Sig Object BOR

:Rel hasComponent
:Inst BinaryRel
:Sig Object Component
```

```
:Rel hasProduct
:Inst BinaryRel
:Sig ProductFamily Product

:Rel hasProductVersion
:Inst BinaryRel
:Sig Product ProductVersion

:Rel hasAssemblyResource
:Inst BinaryRel
:Sig Object AssemblyResource

:Rel hasStep
:Inst BinaryRel
:Sig AssemblyOperation Step

:Rel hasAssemblyComponentList
:Inst BinaryRel
:Sig BOM AssemblyComponentList

:Rel hasADAssemblyComponentList
:Inst BinaryRel
:Sig BOM AssemblyComponentList
:supRel hasAssemblyComponentList

:Rel hasARAssemblyComponentList
:Inst BinaryRel
:Sig BOM AssemblyComponentList
:supRel hasAssemblyComponentList

:Rel hasAuxiliaryMaterialList
:Inst BinaryRel
:Sig BOM AuxiliaryMaterialList

:Rel hasAssemblyProcessWith
:Inst TernaryRel
:Inst IntensionalRel
:Sig Object Object AssemblyProcess

:Rel isAssembledWithIn
```

```
:Inst TernaryRel
:Inst IntensionalRel
:Sig Object Object ManufacturingFacility



:Rel usesAssemblyResource
:Inst BinaryRel
:Sig Event AssemblyResource

:Rel isPerformedIn
:Inst BinaryRel
:Inst IntensionalRel
:Sig AssemblyProcess ManufacturingFacility

:Rel hasAssemblyFeature
:Inst BinaryRel
:Inst RigidRel
:Sig Object AssemblyFeature

:Rel hasHandlingAF
:Inst BinaryRel
:Inst RigidRel
:Sig Object HandlingAF
:supRel hasAssemblyFeature

:Rel hasToolingAF
:Inst BinaryRel
:Inst RigidRel
:Sig Object ToolingAF
:supRel hasAssemblyFeature

:Rel hasAssemblyResourceFeature
:Inst BinaryRel
:Inst RigidRel
:Sig AssemblyResource AssemblyResourceFeature

:Rel hasHandlingARF
:Inst BinaryRel
:Inst RigidRel
:Sig AssemblyResource HandlingARF
```

```
:supRel hasAssemblyResourceFeature

:Rel hasToolingARF
:Inst BinaryRel
:Inst RigidRel
:Sig AssemblyResource ToolingARF
:supRel hasAssemblyResourceFeature

:Rel hasAssemblyOperation
:Inst BinaryRel
:Inst RigidRel
:Sig Product AssemblyOperation

;;;==================================================
;;; Functions
;;;==================================================

:Fun m
:Inst UnaryFun
:Inst MeasureFun
:Sig RealNumber -> LengthDimension

:Fun cm
:Inst UnaryFun
:Inst MeasureFun
:Sig RealNumber -> LengthDimension

:Fun mm
:Inst UnaryFun
:Inst MeasureFun
:Sig RealNumber -> LengthDimension

:Fun micron
:Inst UnaryFun
:Inst MeasureFun
:Sig RealNumber -> LengthDimension

:Fun degree
:Inst UnaryFun
:Inst MeasureFun
```

```
:Sig RealNumber -> AngularDimension

:Fun tolerance
:Inst BinaryFun
:Sig Dimension Dimension -> Tolerance

:Fun pointlocation
:Inst TernaryFun
:Sig LengthDimension LengthDimension LengthDimension -> PointLocation

:Fun angularlocation
:Inst TernaryFun
:Sig AngularDimension AngularDimension AngularDimension -> AngularLocation


;;;=========================================================
;;; Conversions between units of measurement of lengths
;;;=========================================================

(measureMultiple m 100 cm)
(measureMultiple cm 10 mm)
(measureMultiple mm 1000 micron)


;;;===================================================
;;; Logic
;;;===================================================


(=> (AssemblyComponentList ?l)
      (exists (?c)
            (and (AssemblyComponent ?c)
                     (item ?l ?c))))
:IC hard "Every assembly component list consists of at least a assembly component within the list."

(=> (AssemblyComponentList ?l)
       (not (exists (?other)
                    (and (item ?l ?other)
                    (not (AssemblyComponent ?other))))))
```

```
:IC hard "Every assembly component list should consist of exclusively assembly
components that make up the list."


(=> (ADAssemblyComponentList ?l)

       (not (exists (?x)

             (and (ARAssemblyComponent ?x)

               (item ?l ?x)))))

:IC hard "Every AD assembly component list should not consist of AR assembly
components."


(=> (ARAssemblyComponentList ?l)

       (not (exists (?x)

             (and (ADAssemblyComponent ?x)

               (item ?l ?x)))))

:IC hard "Every AR assembly component list should not consist of AD assembly
components."


(=> (AuxiliaryMaterialList ?l)

      (exists (?c)

      (and (AuxiliaryMaterial ?c)

            (item ?l ?c))))

:IC hard "Every Auxiliary Material List list consists of at least a Auxiliary Material
within the list."


(=> (AuxiliaryMaterialList ?l)

       (not (exists (?other)

            (and (item ?l ?other)

                (not (AuxiliaryMaterial ?other))))))

:IC hard "Every Auxiliary Material List should consist of exclusively Auxiliary Material
that make up the list."



(=> (MBOM ?mbom)

      (exists (?aclist)

            (and (AssemblyComponentList ?aclist)

       (hasAssemblyComponentList ?mbom ?aclist))))

:IC hard "Every MBOM should should have assembly component list."


(=> (hasTolerance ?x (tolerance ?q1 ?q2))

       (and (Object ?x)

              (Dimension ?q1)

              (Dimension ?q2)

              (measureLT ?q1 ?q2)))
```

```
:IC hard "The lower deviation quantity of a Tolerance must always be less than its upper deviation quantity."


(=> (and (hasTolerance ?x (tolerance (?mfunc1 ?num1) (?mfunc2 ?num2)))
                (Object ?x)
                (returnProp ?mfunc1 ?q1)
                (returnProp ?mfunc2 ?q2))
        (= ?mfunc1 ?mfunc2))
:IC hard "Only quantities of the same kind are allowed to participate in the tolerance function."


(<= (hasDimensionWithTolerance ?p ?q ?tol)
        (and (Object ?p)
                (Dimension ?q)
                (Tolerance ?tol)
                (hasDimension ?p ?q)
                (hasTolerance ?p ?tol)))
:rem "The Object ?p whose dimension and tolerance are ?q and ?tol respectively has ?q as its nominal dimension."


(<= (hasAllowableDimensionalValueOrRange ?p (cm ?valueOrInterval))
        (and (Object ?p)
                (hasMinAllowableDimension ?p (cm ?num1))
                (hasMaxAllowableDimension ?p (cm ?num2))
                (inInterval ?valueOrInterval (interval in ?num1 ?num2 in))))
:rem "The Object?p has an allowable dimension or an allowable range of dimensions in centimetres dictated by its minimum and maximum allowable dimensions."


(<= (hasAllowableDimensionalValueOrRange ?p (mm ?valueOrInterval))
        (and (Object ?p)
                (hasMinAllowableDimension ?p (mm ?num1))
                (hasMaxAllowableDimension ?p (mm ?num2))
                (inInterval ?valueOrInterval (interval in ?num1 ?num2 in))))
:rem "The Object ?p has an allowable dimension or an allowable range of dimensions in millimetres dictated by its minimum and maximum allowable dimensions."


(<= (hasAllowableDimensionalValueOrRange ?p (micron ?valueOrInterval))
        (and (Object ?p)
                (hasMinAllowableDimension ?p (micron ?num1))
                (hasMaxAllowableDimension ?p (micron ?num2))
                (inInterval ?valueOrInterval (interval in ?num1 ?num2 in))))
:rem "The Object ?p has an allowable dimension or an allowable range of dimensions in microns dictated by its minimum and maximum allowable dimensions."
```

B.2 Formalization of MBOM Domain Specific Concepts

```
;;;==================================================
;;; MBOMs
;;;==================================================


:Ctx APPMs
:Inst UserContext
:supCtx ARO

:Use APPMs


:Prop MBOMs
:Inst Type
:sup MBOM

(=> (MBOMs ?mboms)
      (exists (?amlist)
            (and (AuxiliaryMaterialList ?amlist)
               (hasAuxiliaryMaterialList ?mboms ?amlist))))
:IC hard "Every MBOMs should have auxiliary material list."



;;;===============================================
;;; MBOMh
;;;===============================================


:Ctx APPMh
:Inst UserContext
:supCtx ARO

:Use APPMh


:Prop MBOMh
:Inst Type
:sup MBOM
```

```
(=> (and (MBOMh ?mbh)
        (AuxiliaryMaterialList ?aml))
         (not (hasAuxiliaryMaterialList ?mbh ?aml)))
:IC hard "MBOMh should not have AuxiliaryMaterialList."


(=> (and (MBOMh ?mbh)
        (ARAssemblyComponentList ?aacl))
         (not (hasARAssemblyComponentList ?mbh ?aacl)))
:IC hard "MBOMh should not have ARAssemblyComponentList."

;;;===============================================
;;; MBOMi
;;;===============================================
:Ctx APPMi

:Inst UserContext

:supCtx ARO


:Use APPMi


:Prop MBOMi

:Inst Type

:sup MBOM


(=> (and (MBOMi ?mbi)

        (AuxiliaryMaterialList ?aml))

         (not (hasAuxiliaryMaterialList ?mbi ?aml)))

:IC hard "MBOMi should not have AuxiliaryMaterialList."
```

B.3 Formalization of Design and Planning Domain Concepts Related to Assembly Feature

B.3.1 Formalization of Assembly Design Domain Specific Concepts

```
:Ctx AyD
:Inst UserContext
:supCtx ARO

:Use AyD

;;;==================================================
;;; Properties
;;;==================================================

:Prop tolTypeH8
:Inst Type
:sup ToleranceType

:Prop tolTypeH7
:Inst Type
:sup ToleranceType

:Prop tolTypef7
:Inst Type
:sup ToleranceType

:Prop tolTypek6
:Inst Type
:sup ToleranceType

:Prop tolTypep6
:Inst Type
:sup ToleranceType

;;;==================================================
;;; Functions
```

```
;;;=====================================================

:Fun f7
:Inst BinaryFun
:Sig Dimension Dimension -> Tolerance


:Fun H8
:Inst BinaryFun
:Sig Dimension Dimension -> Tolerance

:Fun k6
:Inst BinaryFun
:Sig Dimension Dimension -> Tolerance

:Fun H7
:Inst BinaryFun
:Sig Dimension Dimension -> Tolerance

:Fun p6
:Inst BinaryFun
:Sig Dimension Dimension -> Tolerance


;;;=====================================================
;;; Logic
;;;=====================================================
(<= (hasMaxAllowableDimension ?p ?q+upper)
        (and (Object ?p)
                        (hasDimension ?p ?q)
                        (hasTolerance ?p (?tol ?lower ?upper))
                                (measurePlus ?q ?upper ?q+upper)
                 (or (= ?tol tolerance)
                         (= ?tol f7)
                         (= ?tol H8)
                         (= ?tol k6)
                         (= ?tol H7)
                         (= ?tol p6))
                        ))
```

```
;rem "The Object ?p has a maximum allowable dimension which is equivalent to its nominal
dimension plus its upper deviation."



(<= (hasMinAllowableDimension ?p ?q+lower)

        (and (Object ?p)

                 (hasDimensionWithTolerance ?p ?q (?tol ?lower ?upper))

                 (measurePlus ?q ?lower ?q+lower)

                 (or (= ?tol tolerance)

                        (= ?tol f7)

                        (= ?tol H8)

                        (= ?tol k6)

                        (= ?tol H7)

                        (= ?tol p6))))

;rem "The Object ?p has a minimum allowable dimension which is equivalent to its nominal
dimension plus its lower deviation."


(<= (hasDimensionWithTolerance ?c ?q ?tol)

        (and (Object ?c)

                 (Dimension ?q)

                 (Dimension ?tol)

                 (or (= ?tol (tolerance ?qt1 ?qt2))

                        (= ?tol (f7 ?qf1 ?qf2))

                        (= ?tol (H8 ?qH1 ?qH2))

                        (= ?tol (k6 ?qk1 ?qk2))

                        (= ?tol (H7 ?qH3 ?qH4))

                        (= ?tol (p6 ?qp1 ?qp2)))

                (hasTolerance ?c ?tol)

                (hasDimension ?c ?q)))

;rem "The object ?c hasDimensionWithTolerance is same as hasTolerance and hasDimension
for the given dimension and tolerance."


;;;==================================================

;;; f7 Tolerance Specifications

;;;==================================================




        (<= (hasTolerance ?s (f7 ?q1 ?q2))

                        (and (ShaftAF ?s)

                                (hasShapeAttribute ?s ?circular)

                                (Circular ?circular)

                                (= ?q1 (mm -0.016))
```

```
                              (= ?q2 (mm -0.006))
                              (hasDimension ?s ?q)
                              (measureLT (mm 0) ?q)
                              (measureLTE ?q (mm 3))
                              (hasToleranceType ?s ?t)
                              (tolTypef7 ?t)))
:rem  "A  shaft  should  have  tolerance  (f7  (-0.016  mm  to  -0.006  mm))  when  it  has  a
dimension range 0-3 mm."

               (<= (hasTolerance ?s (f7 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm -0.022))
                              (= ?q2 (mm -0.010))
                              (hasDimension ?s ?q)
                              (measureLT (mm 3) ?q)
                              (measureLTE ?q (mm 6))
                              (hasToleranceType ?s ?t)
                              (tolTypef7 ?t)))
:rem  "A  shaft  should  have  tolerance  (f7  (-0.022  mm  to  -0.010  mm))  when  it  has  a
dimension range 3-6 mm."

               (<= (hasTolerance ?s (f7 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm -0.028))
                              (= ?q2 (mm -0.013))
                              (hasDimension ?s ?q)
                              (measureLT (mm 6) ?q)
                              (measureLTE ?q (mm 10))
                              (hasToleranceType ?s ?t)
                              (tolTypef7 ?t)))
:rem  "A  shaft  should  have  tolerance  (f7  (-0.028  mm  to  -0.013  mm))  when  it  has  a
dimension range 6-10 mm."

               (<= (hasTolerance ?s (f7 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                               (= ?q1 (mm -0.034))
```

```
                                 (= ?q2 (mm -0.016))
                                (hasDimension ?s ?q)
                                (measureLT (mm 10) ?q)
                                (measureLTE ?q (mm 18))
                                (hasToleranceType ?s ?t)
                                (tolTypef7 ?t)))
:rem "A shaft should have tolerance (f7 (-0.034 mm to -0.016 mm)) when it has a
dimension range 10-18 mm."

                (<= (hasTolerance ?s (f7 ?q1 ?q2))
                        (and (ShaftAF ?s)
                                (hasShapeAttribute ?s ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm -0.041))
                                (= ?q2 (mm -0.020))
                                (hasDimension ?s ?q)
                                (measureLT (mm 18) ?q)
                                (measureLTE ?q (mm 30))
                                (hasToleranceType ?s ?t)
                                (tolTypef7 ?t)))
:rem "A shaft should have tolerance (f7 (-0.041 mm to -0.020 mm)) when it has a
dimension range 18-30 mm."

                (<= (hasTolerance ?s (f7 ?q1 ?q2))
                        (and (ShaftAF ?s)
                                (hasShapeAttribute ?s ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm -0.050))
                                (= ?q2 (mm -0.025))
                                (hasDimension ?s ?q)
                                (measureLT (mm 30) ?q)
                                (measureLTE ?q (mm 50))
                                (hasToleranceType ?s ?t)
                                (tolTypef7 ?t)))
:rem "A shaft should have tolerance (f7 (-0.050 mm to -0.025 mm)) when it has a
dimension range 30-50 mm."

                (<= (hasTolerance ?s (f7 ?q1 ?q2))
                        (and (ShaftAF ?s)
                                (hasShapeAttribute ?s ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm -0.060))
                                (= ?q2 (mm -0.030))
                                (hasDimension ?s ?q)
                                (measureLT (mm 50) ?q)
                                (measureLTE ?q (mm 80))
                                (hasToleranceType ?s ?t)
                                (tolTypef7 ?t)))
:rem "A shaft should have tolerance (f7 (-0.060 mm to -0.030 mm)) when it has a dimension range 50-80 mm."

                (<= (hasTolerance ?s (f7 ?q1 ?q2))
                        (and (ShaftAF ?s)
                                (hasShapeAttribute ?s ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm -0.071))
                                (= ?q2 (mm -0.036))
                                (hasDimension ?s ?q)
                                (measureLT (mm 80) ?q)
                                (measureLTE ?q (mm 120))
                                (hasToleranceType ?s ?t)
                                (tolTypef7 ?t)))
:rem "A shaft should have tolerance (f7 (-0.071 mm to -0.036 mm)) when it has a dimension range 80-120 mm."

                (<= (hasTolerance ?s (f7 ?q1 ?q2))
                        (and (ShaftAF ?s)
                                (hasShapeAttribute ?s ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm -0.083))
                                (= ?q2 (mm -0.043))
                                (hasDimension ?s ?q)
                                (measureLT (mm 120) ?q)
                                (measureLTE ?q (mm 180))
                                (hasToleranceType ?s ?t)
                                (tolTypef7 ?t)))
:rem "A shaft should have tolerance (f7 (-0.083 mm to -0.043 mm)) when it has a dimension range 120-180 mm."

                (<= (hasTolerance ?s (f7 ?q1 ?q2))
                        (and (ShaftAF ?s)
                                (hasShapeAttribute ?s ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm -0.096))
```

```
                                (= ?q2 (mm -0.050))
                                (hasDimension ?s ?q)
                                (measureLT (mm 180) ?q)
                                (measureLTE ?q (mm 250))
                                (hasToleranceType ?s ?t)
                                (tolTypef7 ?t)))
:rem "A shaft should have tolerance (f7 (-0.096 mm to -0.050 mm)) when it has a
dimension range 180-250 mm."

                (<= (hasTolerance ?s (f7 ?q1 ?q2))
                        (and (ShaftAF ?s)
                                (hasShapeAttribute ?s ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm -0.108))
                                (= ?q2 (mm -0.056))
                                (hasDimension ?s ?q)
                                (measureLT (mm 250) ?q)
                                (measureLTE ?q (mm 315))
                                (hasToleranceType ?s ?t)
                                (tolTypef7 ?t)))
:rem "A shaft should have tolerance (f7 (-0.108 mm to -0.056 mm)) when it has a
dimension range 250-315 mm."

                (<= (hasTolerance ?s (f7 ?q1 ?q2))
                        (and (ShaftAF ?s)
                                (hasShapeAttribute ?s ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm -0.119))
                                (= ?q2 (mm -0.062))
                                (hasDimension ?s ?q)
                                (measureLT (mm 315) ?q)
                                (measureLTE ?q (mm 400))
                                (hasToleranceType ?s ?t)
                                (tolTypef7 ?t)))
:rem "A shaft should have tolerance (f7 (-0.119 mm to -0.062 mm)) when it has a
dimension range 315-400 mm."

                (<= (hasTolerance ?s (f7 ?q1 ?q2))
                        (and (ShaftAF ?s)
                                (hasShapeAttribute ?s ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm -0.131))
```

```
                              (= ?q2 (mm -0.068))
                              (hasDimension ?s ?q)
                              (measureLT (mm 400) ?q)
                              (measureLTE ?q (mm 500))
                              (hasToleranceType ?s ?t)
                              (tolTypef7 ?t)))
:rem "A shaft should have tolerance (f7 (-0.131 mm to -0.068 mm)) when it has a dimension range 400-500 mm."


;;;==================================================
;;; H8 Tolerance Specifications
;;;==================================================
        (<= (hasTolerance ?h (H8 ?q1 ?q2))
                      (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.014))
                              (hasDimension ?h ?q)
                              (measureLT (mm 0) ?q)
                              (measureLTE ?q (mm 3))
                              (hasToleranceType ?h ?t)
                              (tolTypeH8 ?t)))
:rem "A hole should have tolerance (H8 (0 mm to 0.014 mm)) when it has a dimension range 0-3 mm."

               (<= (hasTolerance ?h (H8 ?q1 ?q2))
                      (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.018))
                              (hasDimension ?h ?q)
                              (measureLT (mm 3) ?q)
                              (measureLTE ?q (mm 6))
                              (hasToleranceType ?h ?t)
                              (tolTypeH8 ?t)))
:rem "A hole should have tolerance (H8 (0 mm to 0.018 mm)) when it has a dimension range 3-6 mm."

               (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.022))
                                (hasDimension ?h ?q)
                                (measureLT (mm 6) ?q)
                                (measureLTE ?q (mm 10))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
;rem "A hole should have tolerance (H8 (0 mm to 0.022 mm)) when it has a dimension range
6-10 mm."

                (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.027))
                                (hasDimension ?h ?q)
                                (measureLT (mm 10) ?q)
                                (measureLTE ?q (mm 18))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
;rem "A hole should have tolerance (H8 (0 mm to 0.027 mm)) when it has a dimension range
10-18 mm."

                (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.033))
                                (hasDimension ?h ?q)
                                (measureLT (mm 18) ?q)
                                (measureLTE ?q (mm 30))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
;rem "A hole should have tolerance (H8 (0 mm to 0.033 mm)) when it has a dimension range
18-30 mm."

                (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.039))
                                (hasDimension ?h ?q)
                                (measureLT (mm 30) ?q)
                                (measureLTE ?q (mm 50))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
:rem "A hole should have tolerance (H8 (0 mm to 0.039 mm)) when it has a dimension range
30-50 mm."

                (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.046))
                                (hasDimension ?h ?q)
                                (measureLT (mm 50) ?q)
                                (measureLTE ?q (mm 80))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
:rem "A hole should have tolerance (H8 (0 mm to 0.046 mm)) when it has a dimension range
50-80 mm."

                (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.054))
                                (hasDimension ?h ?q)
                                (measureLT (mm 80) ?q)
                                (measureLTE ?q (mm 120))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
:rem "A hole should have tolerance (H8 (0 mm to 0.054 mm)) when it has a dimension range
80-120 mm."

                (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.063))
                                (hasDimension ?h ?q)
                                (measureLT (mm 120) ?q)
                                (measureLTE ?q (mm 180))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
;rem "A hole should have tolerance (H8 (0 mm to 0.063 mm)) when it has a dimension range
120-180 mm."

                (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.072))
                                (hasDimension ?h ?q)
                                (measureLT (mm 180) ?q)
                                (measureLTE ?q (mm 250))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
;rem "A hole should have tolerance (H8 (0 mm to 0.072 mm)) when it has a dimension range
180-250 mm."

                (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.081))
                                (hasDimension ?h ?q)
                                (measureLT (mm 250) ?q)
                                (measureLTE ?q (mm 315))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
;rem "A hole should have tolerance (H8 (0 mm to 0.081 mm)) when it has a dimension range
250-315 mm."

                (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.089))
                                (hasDimension ?h ?q)
                                (measureLT (mm 315) ?q)
                                (measureLTE ?q (mm 400))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
;rem "A hole should have tolerance (H8 (0 mm to 0.089 mm)) when it has a dimension range
315-400 mm."

                (<= (hasTolerance ?h (H8 ?q1 ?q2))
                        (and (HoleAF ?h)
                                (hasShapeAttribute ?h ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.097))
                                (hasDimension ?h ?q)
                                (measureLT (mm 400) ?q)
                                (measureLTE ?q (mm 500))
                                (hasToleranceType ?h ?t)
                                (tolTypeH8 ?t)))
;rem "A hole should have tolerance (H8 (0 mm to 0.097 mm)) when it has a dimension range
400-500 mm."

;;;==================================================
;;; k6 Tolerance Specifications
;;;==================================================
        (<= (hasTolerance ?s (k6 ?q1 ?q2))
                        (and (ShaftAF ?s)
                                (hasShapeAttribute ?s ?circular)
                                (Circular ?circular)
                                (= ?q1 (mm 0))
                                (= ?q2 (mm 0.006))
                                (hasDimension ?s ?q)
                                (measureLT (mm 0) ?q)
                                (measureLTE ?q (mm 3))
                                (hasToleranceType ?s ?t)
                                (tolTypek6 ?t)))
```

```
:rem "A shaft should have tolerance (k6 (0 mm to 0.006 mm)) when it has a dimension
range 0-3 mm."


               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                      (and (ShaftAF ?s)
                             (hasShapeAttribute ?s ?circular)
                             (Circular ?circular)
                             (= ?q1 (mm 0.001))
                             (= ?q2 (mm 0.009))
                             (hasDimension ?s ?q)
                             (measureLT (mm 3) ?q)
                             (measureLTE ?q (mm 6))
                             (hasToleranceType ?s ?t)
                             (tolTypek6 ?t)))
:rem "A shaft should have tolerance (k6 (0.001 mm to 0.009 mm)) when it has a dimension
range 3-6 mm."

               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                      (and (ShaftAF ?s)
                             (hasShapeAttribute ?s ?circular)
                             (Circular ?circular)
                             (= ?q1 (mm 0.001))
                             (= ?q2 (mm 0.010))
                             (hasDimension ?s ?q)
                             (measureLT (mm 6) ?q)
                             (measureLTE ?q (mm 10))
                             (hasToleranceType ?s ?t)
                             (tolTypek6 ?t)))
:rem "A shaft should have tolerance (k6 (0.001 mm to 0.010 mm)) when it has a dimension
range 6-10 mm."

               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                      (and (ShaftAF ?s)
                             (hasShapeAttribute ?s ?circular)
                             (Circular ?circular)
                             (= ?q1 (mm 0.001))
                             (= ?q2 (mm 0.012))
                             (hasDimension ?s ?q)
                             (measureLT (mm 10) ?q)
                             (measureLTE ?q (mm 18))
                             (hasToleranceType ?s ?t)
```

```
                              (tolTypek6 ?t)))

:rem "A shaft should have tolerance (k6 (0.001 mm to 0.012 mm)) when it has a dimension range 10-18 mm."


               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.002))
                              (= ?q2 (mm 0.015))
                              (hasDimension ?s ?q)
                              (measureLT (mm 18) ?q)
                              (measureLTE ?q (mm 30))
                              (hasToleranceType ?s ?t)
                              (tolTypek6 ?t)))

:rem "A shaft should have tolerance (k6 (0.002 mm to 0.015 mm)) when it has a dimension range 18-30 mm."


               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.002))
                              (= ?q2 (mm 0.018))
                              (hasDimension ?s ?q)
                              (measureLT (mm 30) ?q)
                              (measureLTE ?q (mm 50))
                              (hasToleranceType ?s ?t)
                              (tolTypek6 ?t)))

:rem "A shaft should have tolerance (k6 (0.02 mm to 0.018 mm)) when it has a dimension range 30-50 mm."


               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.002))
                              (= ?q2 (mm 0.021))
                              (hasDimension ?s ?q)
                              (measureLT (mm 50) ?q)
                              (measureLTE ?q (mm 80))
                              (hasToleranceType ?s ?t)
                              (tolTypek6 ?t)))

:rem "A shaft should have tolerance (k6 (0.002 mm to 0.021 mm)) when it has a dimension range 50-80 mm."

               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.003))
                              (= ?q2 (mm 0.025))
                              (hasDimension ?s ?q)
                              (measureLT (mm 80) ?q)
                              (measureLTE ?q (mm 120))
                              (hasToleranceType ?s ?t)
                              (tolTypek6 ?t)))

:rem "A shaft should have tolerance (k6 (0.003 mm to 0.025 mm)) when it has a dimension range 80-120 mm."

               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.003))
                              (= ?q2 (mm 0.028))
                              (hasDimension ?s ?q)
                              (measureLT (mm 120) ?q)
                              (measureLTE ?q (mm 180))
                              (hasToleranceType ?s ?t)
                              (tolTypek6 ?t)))

:rem "A shaft should have tolerance (k6 (0.003 mm to 0.028 mm)) when it has a dimension range 120-180 mm."

               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.004))
                              (= ?q2 (mm 0.033))
                              (hasDimension ?s ?q)
                              (measureLT (mm 180) ?q)
                              (measureLTE ?q (mm 250))
                              (hasToleranceType ?s ?t)
                              (tolTypek6 ?t)))

:rem "A shaft should have tolerance (k6 (0.004 mm to 0.033 mm)) when it has a dimension range 180-250 mm."

               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                      (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.004))
                              (= ?q2 (mm 0.036))
                              (hasDimension ?s ?q)
                              (measureLT (mm 250) ?q)
                              (measureLTE ?q (mm 315))
                              (hasToleranceType ?s ?t)
                              (tolTypek6 ?t)))

:rem "A shaft should have tolerance (k6 (0.004 mm to 0.036 mm)) when it has a dimension range 250-315 mm."

               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                      (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.004))
                              (= ?q2 (mm 0.040))
                              (hasDimension ?s ?q)
                              (measureLT (mm 315) ?q)
                              (measureLTE ?q (mm 400))
                              (hasToleranceType ?s ?t)
                              (tolTypek6 ?t)))

:rem "A shaft should have tolerance (k6 (0.004 mm to 0.040 mm)) when it has a dimension range 315-400 mm."

               (<= (hasTolerance ?s (k6 ?q1 ?q2))
                      (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.005))
                              (= ?q2 (mm 0.045))
                              (hasDimension ?s ?q)
                              (measureLT (mm 400) ?q)
                              (measureLTE ?q (mm 500))
                              (hasToleranceType ?s ?t)
                              (tolTypek6 ?t)))
;rem "A shaft should have tolerance (k6 (0.005 mm to 0.045 mm)) when it has a dimension
range 400-500 mm."



;;;==================================================
;;; H7 Tolerance Specifications
;;;==================================================
        (<= (hasTolerance ?h (H7 ?q1 ?q2))
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.010))
                              (hasDimension ?h ?q)
                              (measureLT (mm 0) ?q)
                              (measureLTE ?q (mm 3))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
;rem "A hole should have tolerance (H7 (0 mm to 0.010 mm)) when it has a dimension range
0-3 mm."

               (<= (hasTolerance ?h (H7 ?q1 ?q2))
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.012))
                              (hasDimension ?h ?q)
                              (measureLT (mm 3) ?q)
                              (measureLTE ?q (mm 6))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
;rem "A hole should have tolerance (H7 (0 mm to 0.012 mm)) when it has a dimension range
3-6 mm."

               (<= (hasTolerance ?h (H7 ?q1 ?q2))
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.015))
                              (hasDimension ?h ?q)
                              (measureLT (mm 6) ?q)
                              (measureLTE ?q (mm 10))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
;rem "A hole should have tolerance (H7 (0 mm to 0.015 mm)) when it has a dimension range
6-10 mm."

               (<= (hasTolerance ?h (H7 ?q1 ?q2))
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.018))
                              (hasDimension ?h ?q)
                              (measureLT (mm 10) ?q)
                              (measureLTE ?q (mm 18))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
;rem "A hole should have tolerance (H7 (0 mm to 0.018 mm)) when it has a dimension range
10-18 mm."

               (<= (hasTolerance ?h (H7 ?q1 ?q2))
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.021))
                              (hasDimension ?h ?q)
                              (measureLT (mm 18) ?q)
                              (measureLTE ?q (mm 30))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
;rem "A hole should have tolerance (H7 (0 mm to 0.021 mm)) when it has a dimension range
18-30 mm."

               (<= (hasTolerance ?h (H7 ?q1 ?q2))
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.025))
                              (hasDimension ?h ?q)
                              (measureLT (mm 30) ?q)
                              (measureLTE ?q (mm 50))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
:rem "A hole should have tolerance (H7 (0 mm to 0.025 mm)) when it has a dimension range
30-50 mm."

               (<= (hasTolerance ?h (H7 ?q1 ?q2))
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.030))
                              (hasDimension ?h ?q)
                              (measureLT (mm 50) ?q)
                              (measureLTE ?q (mm 80))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
:rem "A hole should have tolerance (H7 (0 mm to 0.030 mm)) when it has a dimension range
50-80 mm."

               (<= (hasTolerance ?h (H7 ?q1 ?q2))
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.035))
                              (hasDimension ?h ?q)
                              (measureLT (mm 80) ?q)
                              (measureLTE ?q (mm 120))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
:rem "A hole should have tolerance (H7 (0 mm to 0.035 mm)) when it has a dimension range
80-120 mm."

               (<= (hasTolerance ?h (H7 ?q1 ?q2))
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.040))
                              (hasDimension ?h ?q)
                              (measureLT (mm 120) ?q)
                              (measureLTE ?q (mm 180))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
;rem "A hole should have tolerance (H7 (0 mm to 0.040 mm)) when it has a dimension range
120-180 mm."

              (<= (hasTolerance ?h (H7 ?q1 ?q2))
                      (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.046))
                              (hasDimension ?h ?q)
                              (measureLT (mm 180) ?q)
                              (measureLTE ?q (mm 250))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
;rem "A hole should have tolerance (H7 (0 mm to 0.046 mm)) when it has a dimension range
180-250 mm."

              (<= (hasTolerance ?h (H7 ?q1 ?q2))
                      (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.052))
                              (hasDimension ?h ?q)
                              (measureLT (mm 250) ?q)
                              (measureLTE ?q (mm 315))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
;rem "A hole should have tolerance (H7 (0 mm to 0.052 mm)) when it has a dimension range
250-315 mm."

              (<= (hasTolerance ?h (H7 ?q1 ?q2))
                      (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.057))
                              (hasDimension ?h ?q)
                              (measureLT (mm 315) ?q)
                              (measureLTE ?q (mm 400))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
:rem "A hole should have tolerance (H7 (0 mm to 0.057 mm)) when it has a dimension range
315-400 mm."

               (<= (hasTolerance ?h (H7 ?q1 ?q2))
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0))
                              (= ?q2 (mm 0.063))
                              (hasDimension ?h ?q)
                              (measureLT (mm 400) ?q)
                              (measureLTE ?q (mm 500))
                              (hasToleranceType ?h ?t)
                              (tolTypeH7 ?t)))
:rem "A hole should have tolerance (H7 (0 mm to 0.063 mm)) when it has a dimension range
400-500 mm."

;;;==================================================
;;; p6 Tolerance Specifications
;;;==================================================
        (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.006))
                              (= ?q2 (mm 0.012))
                              (hasDimension ?s ?q)
                              (measureLT (mm 0) ?q)
                              (measureLTE ?q (mm 3))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.006 mm to 0.012 mm)) when it has a dimension
range 0-3 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.012))
                              (= ?q2 (mm 0.020))
                              (hasDimension ?s ?q)
                              (measureLT (mm 3) ?q)
                              (measureLTE ?q (mm 6))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.012 mm to 0.020 mm)) when it has a dimension range 3-6 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.015))
                              (= ?q2 (mm 0.024))
                              (hasDimension ?s ?q)
                              (measureLT (mm 6) ?q)
                              (measureLTE ?q (mm 10))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.015 mm to 0.024 mm)) when it has a dimension range 6-10 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.018))
                              (= ?q2 (mm 0.029))
                              (hasDimension ?s ?q)
                              (measureLT (mm 10) ?q)
                              (measureLTE ?q (mm 18))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.018 mm to 0.029 mm)) when it has a dimension range 10-18 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.022))
                              (= ?q2 (mm 0.035))
                              (hasDimension ?s ?q)
                              (measureLT (mm 18) ?q)
                              (measureLTE ?q (mm 30))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.022 mm to 0.035 mm)) when it has a dimension
range 18-30 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.026))
                              (= ?q2 (mm 0.042))
                              (hasDimension ?s ?q)
                              (measureLT (mm 30) ?q)
                              (measureLTE ?q (mm 50))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.026 mm to 0.042 mm)) when it has a dimension
range 30-50 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.032))
                              (= ?q2 (mm 0.051))
                              (hasDimension ?s ?q)
                              (measureLT (mm 50) ?q)
                              (measureLTE ?q (mm 80))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.032 mm to 0.051 mm)) when it has a dimension
range 50-80 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.037))
                              (= ?q2 (mm 0.059))
                              (hasDimension ?s ?q)
                              (measureLT (mm 80) ?q)
                              (measureLTE ?q (mm 120))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.037 mm to 0.059 mm)) when it has a dimension
range 80-120 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.043))
                              (= ?q2 (mm 0.068))
                              (hasDimension ?s ?q)
                              (measureLT (mm 120) ?q)
                              (measureLTE ?q (mm 180))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.043 mm to 0.068 mm)) when it has a dimension
range 120-180 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.050))
                              (= ?q2 (mm 0.079))
                              (hasDimension ?s ?q)
                              (measureLT (mm 180) ?q)
                              (measureLTE ?q (mm 250))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.050 mm to 0.079 mm)) when it has a dimension
range 180-250 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.056))
                              (= ?q2 (mm 0.088))
                              (hasDimension ?s ?q)
                              (measureLT (mm 250) ?q)
                              (measureLTE ?q (mm 315))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.056 mm to 0.088 mm)) when it has a dimension
range 250-315 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.062))
                              (= ?q2 (mm 0.098))
                              (hasDimension ?s ?q)
                              (measureLT (mm 315) ?q)
                              (measureLTE ?q (mm 400))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.062 mm to 0.098 mm)) when it has a dimension
range 315-400 mm."

               (<= (hasTolerance ?s (p6 ?q1 ?q2))
                       (and (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (Circular ?circular)
                              (= ?q1 (mm 0.068))
                              (= ?q2 (mm 0.108))
                              (hasDimension ?s ?q)
                              (measureLT (mm 400) ?q)
                              (measureLTE ?q (mm 500))
                              (hasToleranceType ?s ?t)
                              (tolTypep6 ?t)))
:rem "A shaft should have tolerance (p6 (0.068 mm to 0.108 mm)) when it has a dimension
range 400-500 mm."

;;;==================================================
;;; Semantics of Fits
;;;==================================================
        (<= (hasClearanceFitWith ?h ?s)
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (ShaftAF ?s)
                              (hasShapeAttribute ?s ?circular)
                              (or (matesWith ?h ?s)
                                     (and (hasDimension ?h ?d1)
                                             (hasDimension ?s ?d2)
                                             (= ?d1 ?d2)
                                             (matesWith ?h ?s)
                                             (hasToleranceType ?h ?tolH8)
                                             (hasToleranceType ?s ?tolf7)
                                             (tolTypeH8 ?tolH8)
                                             (tolTypef7 ?tolf7)))
                              (hasMinAllowableDimension ?h ?holeMinDim)
                              (hasMaxAllowableDimension ?s ?shaftMaxDim)
                              (measureLT ?shaftMaxDim ?holeMinDim)))
;rem "A hole has clearnce Fit With a shaft if the minimum allowable dimension of hole is
larger than the maximum allowable dimension of shaft."

        (<= (hasTransitionFitWith ?h ?s)
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (ShaftAF ?s)
                              (or (matesWith ?h ?s)
                                     (and (hasDimension ?h ?d1)
                                             (hasDimension ?s ?d2)
                                             (= ?d1 ?d2)
                                             (matesWith ?h ?s)
                                             (hasToleranceType ?h ?tolH7)
                                             (hasToleranceType ?s ?tolk6)
                                             (tolTypeH7 ?tolH7)
                                             (tolTypek6 ?tolk6)))
                              (hasMinAllowableDimension ?h ?holeMinDim)
                              (hasMaxAllowableDimension ?h ?holeMaxDim)
                              (hasMinAllowableDimension ?s ?shaftMinDim)
                              (hasMaxAllowableDimension ?s ?shaftMaxDim)
                              (measureLT ?shaftMinDim ?holeMaxDim)
                              (measureLT ?holeMinDim ?shaftMaxDim)))
:rem "A hole hasTransitionFitWith a shaft if the minimum allowable dimension of hole is
smaller than the maximum allowable dimension of shaft and maximum allowable dimension of
hole is larger than the minimum allowable dimension of shaft."


        (<= (hasInterferenceFitWith ?h ?s)
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (ShaftAF ?s)
                              (or (matesWith ?h ?s)
                                     (and (hasDimension ?h ?d1)
                                             (hasDimension ?s ?d2)
                                             (= ?d1 ?d2)
                                             (matesWith ?h ?s)
                                             (hasToleranceType ?h ?tolH7)
                                             (hasToleranceType ?s ?tolp6)
                                             (tolTypeH7 ?tolH7)
                                             (tolTypep6 ?tolp6)))
                              (hasMaxAllowableDimension ?h ?holeMaxDim)
                              (hasMinAllowableDimension ?s ?shaftMinDim)
                              (measureLT ?holeMaxDim ?shaftMinDim)))
:rem "A hole hasInterferenceFitWith a shaft if the maximum allowable dimension of hole
is smaller than the minimum allowable dimension of shaft."


        (<= (hasAllowanceWith ?h ?s ?allowance)
                       (and (HoleAF ?h)
                              (hasShapeAttribute ?h ?circular)
                              (Circular ?circular)
                              (ShaftAF ?s)
                              (Dimension ?allowance)
                              (hasClearanceFitWith ?h ?s)
                              (hasMinAllowableDimension ?h ?qmin)
                              (hasMaxAllowableDimension ?s ?qmax)
                              (measureMinus ?qmin ?qmax ?allowance)))
        :rem "A hole hasAllowanceWith a shaft if the minimum allowable dimension of hole
is larger than the maximum allowable dimension of shaft which is left intensionally and
assembly features having clearance fit have always positive allowance."


        (<= (hasAllowanceWith ?h ?s ?allowance)
                       (and (HoleAF ?h)

                              (hasShapeAttribute ?h ?circular)

                              (Circular ?circular)

                              (ShaftAF ?s)

                              (Dimension ?allowance)

                              (hasInterferenceFitWith ?h ?s)

                              (hasMaxAllowableDimension ?h ?qmax)

                              (hasMinAllowableDimension ?s ?qmin)

                              (measureMinus ?qmax ?qmin ?allowance)))

:rem "A hole hasAllowanceWith a shaft if the maximum allowable dimension of hole is
smaller than the minimum allowable dimension of shaft which is left intensionally and
assembly features having interference fit have always negative allowance."


               (<= (hasAllowanceWith ?h ?s ?allowance)

                       (and (HoleAF ?h)

                              (hasShapeAttribute ?h ?circular)

                              (Circular ?circular)

                              (ShaftAF ?s)

                              (Dimension ?allowance)

                              (hasTransitionFitWith ?h ?s)

                              (hasMaxAllowableDimension ?h ?qhmax)

                              (hasMinAllowableDimension ?h ?qhmin)

                              (hasMaxAllowableDimension ?s ?qsmax)

                              (hasMinAllowableDimension ?s ?qsmin)

                              (or (measureMinus ?qhmax ?qsmin ?allowance)

                                     (measureMinus ?qhmin ?qsmax ?allowance))))

        :rem "A hole hasAllowanceWith a shaft either when the maximum allowable dimension
of hole is larger than the minimum allowable dimension of the shaft

                or maximum allowable dimension of shaft is larger than the minimum
allowable dimension of the hole."

               "
```

B.3.2 Formalization of Assembly Process Planning Domain Specific Concepts

```
:Ctx APP

:Inst UserContext

:supCtx ARO


:Use APP


;;;==================================================

;;; Properties
```

```
;;;==================================================

:Prop PressFitting
:Inst Type
:sup AssemblyProcess

:Prop ShrinkFitting
:Inst Type
:sup AssemblyProcess

:Prop ManualInsertion
:Inst Type
:sup AssemblyProcess

:Prop MachineAssistedInsertion
:Inst Type
:sup AssemblyProcess

:Prop PressFitMachine
:Inst Type
:sup AssemblyResource

:Prop Furnace
:Inst Type
:sup AssemblyResource

:Prop HeatingTorch
:Inst Type
:sup AssemblyResource

:Prop Operator
:Inst Type
:sup AssemblyResource

:Prop Robot
:Inst Type
:sup AssemblyResource

;;;==================================================
;;; Logic
```

```
;;;====================================================


        (<= (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)
                       (and (HoleAF ?hole)
                                (ShaftAF ?shaft)
                                (or (PressFitting ?assemblyprocess)
                                        (ShrinkFitting ?assemblyprocess))
                                (hasInterferenceFitWith ?hole ?shaft)))
:rem "A hole can have press fitting or shrink fitting assembly process if it has interference fit with the shaft."


        (<= (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)
                       (and (HoleAF ?hole)
                                (ShaftAF ?shaft)
                                (or (ManualInsertion ?assemblyprocess)
                                        (MachineAssistedInsertion ?assemblyprocess))
                                (hasClearanceFitWith ?hole ?shaft)))
:rem "A hole can have manual insertion or machine assisted insertion assembly process if it has clearance fit with the shaft."


(<= (usesAssemblyResource ?assemblyprocess ?assemblyresource)
               (and (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)
               (HoleAF ?hole)
               (ShaftAF ?shaft)
               (PressFitting ?assemblyprocess)
               (PressFitMachine ?assemblyresource)
               (hasAssemblyResource ?manufacturingfacility ?assemblyresource)
               (ManufacturingFacility ?manufacturingfacility)))
:rem "Press fit assembly process can use press fit machine as assembly resource."


(<= (usesAssemblyResource ?assemblyprocess ?assemblyresource)
        (and (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)
         (HoleAF ?hole)
         (ShaftAF ?shaft)
         (ShrinkFitting ?assemblyprocess)
         (or (Furnace ?assemblyresource)
         (HeatingTorch ?assemblyresource))
        (hasAssemblyResource ?manufacturingfacility ?assemblyresource)
         (ManufacturingFacility ?manufacturingfacility)))
:rem "Shrink fit assembly process can use furnace or heating torch as assembly resource."
```

```
(<= (usesAssemblyResource ?assemblyprocess ?assemblyresource)
       (and (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)
              (HoleAF ?hole)
              (ShaftAF ?shaft)
              (ManualInsertion ?assemblyprocess)
              (Operator ?assemblyresource)
              (hasAssemblyResource ?manufacturingfacility ?assemblyresource)
              (ManufacturingFacility ?manufacturingfacility)))
:rem "Manual Insertion assembly process can use human operator as assembly resource."


(<= (usesAssemblyResource ?assemblyprocess ?assemblyresource)
       (and (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)
              (HoleAF ?hole)
              (ShaftAF ?shaft)
              (MachineAssistedInsertion ?assemblyprocess)
              (Robot ?assemblyresource)
              (hasAssemblyResource ?manufacturingfacility ?assemblyresource)
              (ManufacturingFacility ?manufacturingfacility)))
:rem "Machine assisted Insertion assembly process can use robot as assembly resource."

       (<= (isPerformedIn ?assemblyprocess ?manufacturingfacility)
              (and (usesAssemblyResource ?assemblyprocess ?assemblyresource)
               (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)
               (HoleAF ?hole)
               (ShaftAF ?shaft)
               (hasAssemblyResource ?manufacturingfacility ?assemblyresource)
               (ManufacturingFacility ?manufacturingfacility)))
:rem "An assembly process can be performed in a manufacturing facility if the later has
the required assembly resources and that assembly process can use those resources."

       (<= (isAssembledWithIn ?hole ?shaft ?manufacturingfacility)
              (and (HoleAF ?hole)
               (ShaftAF ?shaft)
               (ManufacturingFacility ?manufacturingfacility)
               (hasAssemblyProcessWith ?hole ?shaft ?assemblyprocess)
               (isPerformedIn ?assemblyprocess ?manufacturingfacility)
               (hasAssemblyResource ?manufacturingfacility ?assemblyresource)
               (usesAssemblyResource ?assemblyprocess ?assemblyresource)))
:rem "A hole can be assembled with a shaft in a manufacturing facility if the later has
the required assembly resource.""
```

B.4 Formalization of Case Study Scenario

```
:Ctx CS
:Inst UserContext
:supCtx ARO

:Use CS

;;;==================================================
;;; Relations
;;;==================================================

:Rel canBeUsedAsAssemblyResourceFor
:Inst BinaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig AssemblyResource Product

:Rel hasHandlingAFusedAt
:Inst TernaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig Product HandlingAF Step

:Rel hasToolingAFusedAt
:Inst TernaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig Product ToolingAF Step

:Rel hasHandlingARFusedAt
:Inst TernaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig AssemblyResource HandlingARF Step

:Rel hasToolingARFusedAt
:Inst TernaryRel
```

```
:Inst IntensionalRel
:Inst RigidRel
:Sig AssemblyResource ToolingARF Step

:Rel usesHandlingAF
:Inst BinaryRel
:Inst RigidRel
:Sig Event HandlingAF

:Rel usesToolingAF
:Inst BinaryRel
:Inst RigidRel
:Sig Event ToolingAF

:Rel usesHandlingARF
:Inst BinaryRel
:Inst RigidRel
:Sig Event HandlingARF

:Rel usesToolingARF
:Inst BinaryRel
:Inst RigidRel
:Sig Event ToolingARF

:Rel hasNumOfHandlingAF
:Inst BinaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig Product IntegerNumber

:Rel hasNumOfToolingAF
:Inst BinaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig Product IntegerNumber

:Rel hasNumOfHandlingAFat
:Inst TernaryRel
:Inst IntensionalRel
:Inst RigidRel
```

```
:Sig Product IntegerNumber Event

:Rel hasNumOfToolingAFat
:Inst TernaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig Product IntegerNumber Event

:Rel hasNumOfHandlingARF
:Inst BinaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig AssemblyResource IntegerNumber

:Rel hasNumOfToolingARF
:Inst BinaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig AssemblyResource IntegerNumber

:Rel hasNumOfHandlingARFat
:Inst TernaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig AssemblyResource IntegerNumber Event

:Rel hasNumOfToolingARFat
:Inst TernaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig AssemblyResource IntegerNumber Event

:Rel hasEqualNumOfFeatureUsedWith
:Inst BinaryRel
:Inst IntensionalRel
:Inst RigidRel
:Sig AssemblyResource Product

:Rel hasFeatureLocationMateableWith
:Inst BinaryRel
```

```
:Inst IntensionalRel

:Sig AssemblyResource Product


:Rel hasFeatureSizeMateableWith

:Inst BinaryRel

:Inst IntensionalRel

:Sig AssemblyResource Product


;;;==================================================

;;;                    Axioms

;;;==================================================


(<= (hasHandlingAFusedAt ?Engine ?HAF ?step)

       (and (Product ?Engine)

       (HandlingAF ?HAF)

       (Step ?step)

       (hasHandlingAF ?Engine ?HAF)

       (usesHandlingAF ?step ?HAF)

       (hasAssemblyOperation ?Engine ?AssemblyOperation)

       (hasStep ?AssemblyOperation ?step)))

:rem "A product has handling assembly features used at a prticular step of an operation
if that step is part of the assembly operation and it uses those handling features."


(<= (hasHandlingARFusedAt ?AssemblyResource ?HARF ?step)

       (and (AssemblyResource ?AssemblyResource)

               (HandlingARF ?HARF)

               (Step ?step)

               (hasHandlingARF ?AssemblyResource ?HARF)

               (usesHandlingARF ?step ?HARF)

               (usesAssemblyResource ?AssemblyOperation ?AssemblyResource)

               (hasStep ?AssemblyOperation ?step)))

:rem "An assembly resource has handling assembly resource features used at a prticular
step if that step is part of the assembly resource and it uses those handling resource
features."


(<= (hasNumOfHandlingAF ?Engine ?HAFQuantity)

           (and (Product ?Engine)

              (countf (?HAF)

              (hasHandlingAF ?Engine ?HAF) ?HAFQuantity)))

:rem "A product has handling assembly feature quantity equal to the number of asserted
handling assembly features for that product."
```

```
(<= (hasNumOfHandlingAFat ?Engine ?HAFQuantity ?step)
                 (and (Product ?Engine)
                      (Step ?step)
                      (countf (?HAF)
                      (hasHandlingAFusedAt ?Engine ?HAF ?step) ?HAFQuantity)))
```

:rem "A product has handling assembly feature quantity at a particular step of an operation equal to the number of asserted handling assembly features used at that step."

```
(<= (hasNumOfHandlingARF ?AssemblyResource ?HARFQuantity)
         (and (AssemblyResource ?AssemblyResource)
          (countf (?HARF)
         (hasHandlingARF ?AssemblyResource ?HARF) ?HARFQuantity)))
```

:rem "Handling assembly resource feature quantity can be found by counting the asserted handling assembly resource features in an assembly resource."

```
(<= (hasNumOfHandlingARFat ?AssemblyResource ?HARFQuantity ?step)
                 (and (AssemblyResource ?AssemblyResource)
                      (Step ?step)
                      (countf (?HARF)
              (hasHandlingARFusedAt ?AssemblyResource ?HARF ?step) ?HARFQuantity)))
```

:rem "Handling assembly resource feature quantity at a step can be found by counting the asserted handling assembly resource features at that step."

```
(<= (hasEqualNumOfFeatureUsedWith ?AssemblyResource ?Engine)
       (and (AssemblyResource ?AssemblyResource)
               (Product ?Engine)
               (hasNumOfHandlingAFat ?Engine ?HAFQuantity ?step)
               (Step ?step)
               (hasNumOfHandlingARFat ?AssemblyResource ?HARFQuantity ?step)
               (= ?HAFQuantity ?HARFQuantity)
               (gtNum ?HAFQuantity 0)
               (gtNum ?HARFQuantity 0)))
```

:rem "An assembly resource has equal number of handling assembly resource features at a particular step of an operation if at the same step product has same number of handling assembly features except zero."

```
(<= (hasFeatureLocationMateableWith ?AssemblyResource ?Engine)
       (and (AssemblyResource ?AssemblyResource)
               (Product ?Engine)
               (hasAssemblyOperation ?Engine ?AssemblyOperation)
```

```
                (AssemblyOperation ?AssemblyOperation)
                (usesAssemblyResource ?AssemblyOperation ?AssemblyResource)
                (AssemblyOperation ?AssemblyOperation)
                (not (exists (?HAF ?pointlocation1 ?angularlocation1)
               (and (hasHandlingAF ?Engine ?HAF)
                (HandlingAF ?HAF)
                (hasPointLocation  ?HAF ?pointlocation1)
                (hasAngularLocation  ?HAF ?angularlocation1)
                (not (exists (?HARF ?pointlocation2 ?angularlocation2)
                (and (hasHandlingARF ?AssemblyResource ?HARF)
                (HandlingARF ?HARF)
                (hasPointLocation  ?HARF ?pointlocation2)
               (hasAngularLocation  ?HARF ?angularlocation2)
               (= ?pointlocation1 ?pointlocation2)
               (= ?angularlocation1 ?angularlocation2)
               (matesWith ?HAF ?HARF)))))))))
;rem "An assembly resource has features location mateable with a product if location of
handling assembly features on that product are same as that of handling resource
assembly features on that assembly resource."

(<= (hasFeatureSizeMateableWith ?AssemblyResource ?Engine)
        (and (AssemblyResource ?AssemblyResource)
                (Product ?Engine)
                (hasAssemblyOperation ?Engine ?AssemblyOperation)
                (AssemblyOperation ?AssemblyOperation)
                (usesAssemblyResource ?AssemblyOperation ?AssemblyResource)
                (AssemblyOperation ?AssemblyOperation)
                (hasHandlingAF ?Engine ?holeAF)
                (HoleAF ?holeAF)
                (HandlingAF ?holeAF)
                (hasHandlingARF ?AssemblyResource ?shaftARF)
                (ShaftARF ?shaftARF)
                (HandlingARF ?shaftARF)
                (hasShapeAttribute ?holeAF ?circular)
                (Circular ?circular)
                (hasShapeAttribute ?shaftARF ?circular)
                (hasClearanceFitWith ?holeARF ?shaftAF)))
;rem "An assembly resource has feature size matable with a product if the handling
feature has clearance fit with handling resource feature of an assembly resource."

(<= (canBeUsedAsAssemblyResourceFor ?AssemblyResource ?Engine)
        (and (AssemblyResource ?AssemblyResource)
                (Product ?Engine)
                (hasEqualNumOfFeatureUsedWith ?AssemblyResource ?Engine)
                (hasFeatureLocationMateableWith ?AssemblyResource ?Engine)
                (hasFeatureSizeMateableWith ?AssemblyResource ?Engine)))
:rem "An assembly resource can be used a product if it has same number of assembly features, mateable location and matable feature size with that of a product."
```