A Process Model in Platform Independent and Neutral Formal Representation for Design Engineering Automation

Vibhor Trehan

A thesis submitted to Birmingham City University in partial fulfilment for the degree of

Doctor of Philosophy

September 2018

Faculty of Computing, Engineering and Built Environment (CEBE)

Birmingham City University

Abstract

An engineering design process as part of product development (PD) needs to satisfy ever-changing customer demands by striking a balance between time, cost and quality. In order to achieve a faster lead-time, improved quality and reduced PD costs for increased profits, automation methods have been developed with the help of virtual engineering. There are various methods of achieving Design Engineering Automation (DEA) with Computer-Aided (CAx) tools such as CAD/CAE/CAM, Product Lifecycle Management (PLM) and Knowledge Based Engineering (KBE). For example, Computer Aided Design (CAD) tools enable Geometry Automation (GA), PLM systems allow for sharing and exchange of product knowledge throughout the PD lifecycle.

Traditional automation methods are specific to individual products and are hard-coded and bound by the proprietary tool format. Also, existing CAx tools and PLM systems offer bespoke islands of automation as compared to KBE. KBE as a design method incorporates complete design intent by including re-usable geometric, non-geometric product knowledge as well as engineering process knowledge for DEA including various processes such as mechanical design, analysis and manufacturing.

It has been recognised, through an extensive literature review, that a research gap exists in the form of a generic and structured method of knowledge modelling, both informal and formal modelling, of mechanical design process with manufacturing knowledge (DFM/DFA) as part of model based systems engineering (MBSE) for DEA with a KBE approach. There is a lack of a structured technique for knowledge modelling, which can provide a standardised method to use platform independent and neutral formal standards for DEA with generative modelling for mechanical product design process and DFM with preserved semantics. The neutral formal representation through computer or machine understandable format provides open standard usage.

This thesis provides a contribution to knowledge by addressing this gap in two-steps:

- In the first step, a coherent process model, GPM-DEA is developed as part of MBSE which can be used for modelling of mechanical design with manufacturing knowledge utilising hybrid approach, based on strengths of existing modelling standards such as IDEF0, UML, SysML and addition of constructs as per author's Metamodel. The structured process model is highly granular with complex interdependencies such as activities, object, function, rule association and includes the effect of the process model on the product at both component and geometric attributes.
- In the second step, a method is provided to map the schema of the process model to equivalent platform independent and neutral formal standards using OWL/SWRL ontology for system development using Protégé tool, enabling machine interpretability with semantic clarity for DEA with generative modelling by building queries and reasoning on set of generic SWRL functions developed by the author.

Model development has been performed with the aid of literature analysis and pilot use-cases. Experimental verification with test use-cases has confirmed the reasoning and querying capability on formal axioms in generating accurate results. Some of the other key strengths are that knowledgebase is generic, scalable and extensible, hence provides re-usability and wider design space exploration. The generative modelling capability allows the model to generate activities and objects based on functional requirements of the mechanical design process with DFM/DFA and rules based on logic. With the help of application programming interface, a platform specific DEA system such as a KBE tool or a CAD tool enabling GA and a web page incorporating engineering knowledge for decision support can consume relevant part of the knowledgebase.

Keywords: Design engineering automation, process model, platform independent and neutral formal representation, knowledge modelling, semantic clarity, generative modelling

Table of Contents

List of Figures

List of Tables

Publications

Journal Paper Published [Presented in 5th Advanced Design Concepts and Practice Workshop (ADCP), Hangzhou, China, 2015]

- Trehan, V., Chapman, C., Raju, P., 2015. Informal and formal modelling of engineering processes for design automation using knowledge based engineering. J. Zhejiang Univ. Sci. A 16, 706–723. doi:10.1631/jzus.A1500140

Journal Paper Planned for Submission [Journal of Computing and Information Science in Engineering] -

- Trehan, V., Chapman, C., Raju, P., Farazi, F., Oraifige, I., 2018. A Generative Process Model for Design Engineering Automation (GPM-DEA): Represented in OWL/SWRL

Acknowledgements

The course of my research journey through this PhD has been arduous and full of ups and downs. I have learnt a lot from my mentors and my personal experiences during my research. I would like to say thank you to Prof Craig Chapman and Dr. Pathmeswaran Raju for giving me this opportunity. They provided me with a lot of academic and training support in my initial stages and laying the foundation. During the later duration, Prof Ilias Oraifige provided a lot of support in completion of my PhD. I would also like to thank my colleague Dr. Feroz Farazi who helped a lot in my implementation. I would also like to thank Ralph Boyce, Colin Cadas and Jonathan Butters from Rolls Royce as my industrial mentors who helped provide my pilot use-cases and provided support and guidance when needed. I am especially thankful to Prof Craig and Dr. Raju for providing me with an opportunity to travel to China to present my research findings in a conference. It really gave me good exposure and interaction with experts from all over the world and specialising in my field. The training courses helped expand my viewpoint. I am thankful to Prof Peter Larkham for providing assistance with my writing skills in orientation to academic reading. I am also thankful to Prof Hanifa Shah for supporting me towards the completion stages of my research. I enjoyed working with a few colleagues in my lab; Mani, Shiva, Vijay, Gerald, Maxim and Guolong who were encouraging and our discussions helped a lot towards my personal growth. Our administrator Sue Witton was of tremendous help in solving administrative issue and policies.

On my personal front, I would like to thank my parents, sister and my wife who have always supported me in my difficult times. The love and support of my wife has meant a lot to me during my research. She gave birth to my son, Vihaan during my PhD, which was the best gift, anyone has ever given to me. Last, I would like to thank god in charting out a suitable journey for me to undergo this PhD experience which brought clarity and direction towards my future career.

List of Acronyms

AI = Artificial Intelligence

AML = Advanced Modelling Language

API = Application Programming Interface

BPMN = Business Process Modelling Notation

CAD = Computer Aided Design

CAE = Computer Aided Engineering

CAM = Computer Aided Manufacturing

CCM = Common Computational Model

CFD = Computational Fluid Dynamics

CG = Conceptual Graphs

CGIF = Common Graph Interchange Format

CL = Common Logic

CLIF = Common Logic Interchange Format

CLOS = Common Lisp Object System

CPR = Core Plan Representation

CWA = Closed World Assumption

DARPA = Defense Advanced Research Projects Agency

DEA = Design Engineering Automation

DFA = Design for Assembly

DFD = Data Flow Diagram

DFM = Design for Manufacturing

DL = Description Logic

DSM = Design Structure Matrix

EPC = Event Process Chain

FBS = Function-Behaviour-Structure

FEA = Finite Element Analysis

FOPL = First Order Predicate Logic

FOL = First Order Logic

GDL = General-purpose Declarative Language

GA = Geometry Automation

GPM-DEA = Generative Process Model for Design Engineering Automation

GUI = Graphical User Interface

IBIS = Issue Based Information System

IDEF = Integrated Definition for Functional Modelling

IDL = ICAD Design Language

IGES = Initial Graphics Exchange Specification

ISO = International Standards Organization

KBE = Knowledge Based Engineering

KBES = Knowledge Based Engineering System

KBS = Knowledge Based Systems

KIC = Knowledge Intensive CAD

KIF = Knowledge Interchange Format

KR = Knowledge Representation

LISP = LISt processing

MathML = Mathematical Markup Language

MBSE = Model Based Systems Engineering

MDO = Multidisciplinary Design Optimisation

MDA = Model Drive Architecture

MPN = Modified Petrinet

NIST = National Institute of Standards and Technology

OKBC = Open Knowledge Base Connectivity

OO = Object Oriented

OOP = Object Oriented Programming

OPM = Object Process Methodology

OSTN = Object State Transition Network

OWL = Web Ontology Language

OWA = Open World Assumption

PD = Product Development

PDDL = Planning Domain Definition Language

PDM = Product Data Management

PFN = Process Flow Network

PLM = Product Lifecycle Management

PSL = Process Specification Language

RAD = Role Activity Diagram

RDF = Resource Description Framework

RDFS = Resource Description Framework Schema

RIF = Rule Interchange Format

RuleML = Rule Markup Language

SADT = Structured Analysis and Design Technique

SE = Systems Engineering

SPARQL = SPARQL Protocol and RDF Query Language

STEP = STandard for the Exchange of Product data

SWRL = Semantic Web Rule Language

SWSL = Semantic Web Services Language

SysML = Systems Modelling Language

SQWRL = Semantic Query Web Rule Language

UML = Unified Modelling Language

VRML = Virtual Reality Modeling Language

WPDL = Workflow Process Definition Language

WTM = Work Transformation Matrix

XML = eXtensible Markup Language

1 Introduction

1.1 Research Context

The commercial success of a manufacturing enterprise substantially depends upon the efficiency of product development (PD) (Ulrich and Eppinger, 2012). In order to maximise profits, the PD process should have an optimum balance between achieving product quality, cost and development time (Ulrich and Eppinger, 2012). The main task of engineers in the PD stage is to apply their scientific knowledge to generate solutions for technical problems and optimise them based on requirements and constraints such as material, functional, economic, legal and environmental considerations (Pahl et al., 2007).There are complex interdependencies between the design process and the product involved in engineering design (Chalupnik et al., 2006). Engineering knowledge should be efficiently captured, modelled and retrieved for re-use and enhancing the efficiency of the PD process.

One of the methods to improve the efficiency of the PD process is Design Engineering Automation (DEA). DEA is performed in a virtual engineering environment at various stages of the PD lifecycle (Ovtcharova, 2010). Many tools and methods have been utilised by industries to address various aspects of DEA. Different Computer-Aided (CAx) tools such as Computer-Aided Design (CAD), Computer-Aided Engineering (CAE) and Computer-Aided Manufacturing (CAM) tools assist the PD process at the design and manufacturing stages of a PD process (Shintre and Shakir, 2011). CAD tools allow visualisation and representation of product's shape and form with 2D and 3D models (Bernard, 2005). Advancements in CAD tools have led to DEA with parametric modelling facilities to modify the product's shape with variation in dimensional parameters (Bodein et al., 2009). Knowledge intensive CAD (KIC) allowed representation of additional product and engineering design process knowledge buy restricted within a CAD architecture (Tomiyama and Hew, 2000). These can

be referred as Geometry Automation (GA) (Amadori, 2012). CAE tools assist in analysis of a product's performance such as such as finite element analysis (FEA) and computational fluid dynamics (CFD) (Tyapin et al., 2012). CAM tools allow simulation of the manufacturing and production processes for physical realisation of the designed and analysed product (Corallo et al., 2009). Thus, all CAx tools address some aspects of DEA varying from parametric modelling to knowledge sharing with inclusion of manufacturing knowledge for product design. However, due to different file formats of CAx tools there is loss of knowledge while utilising the combined benefits of these DEA methods (Zhang et al., 2009).

As part of virtual engineering, Product Data Management (PDM)/Product Lifecycle Management (PLM) systems allow storage and representation of product and design process knowledge and provide integration of knowledge between CAD/CAE and CAM tools depending upon their configuration and application. However, a major limitation of PLM systems is lack of representation of product's geometric attributes within a unified knowledge model as they mostly link different knowledge sources from CAx tools through a common platform (Burkett et al., 2003).

In order to address the limitations of these existing virtual engineering applications to address the needs of DEA, KBE as a design method was adopted to provide an integrated approach to DEA. CAx and PDM/PLM systems provide islands of automation in context to a more holistic approach for DEA with KBE. KBE is generally regarded as an umbrella term describing the application of knowledge to automate or assist in the engineering task. It can be applied to a wide variety of design processes (Hew et al., 2001). According to Stokes, Knowledge Based Engineering (KBE) can also be defined as 'The use of advanced software techniques to capture and re-use product and process knowledge in an integrated way' (Stokes, 2001).

1.2 Overview of DEA with a KBE approach

'KBE systems aim to capture product and process information in such a way so as to allow businesses to model engineering design processes, and then use the model to automate all or part of the process' (Chapman and Pinfold, 1999, Pg 259). Alternatively, KBE systems can also be defined as 'an evolution of knowledge based systems pertaining to the engineering domain' (La Rocca, 2011; Rocca, 2012).Figure 1-1 demonstrates a reduction in the product design lifecycle time using KBE vs. Traditional CAD based design methodology.

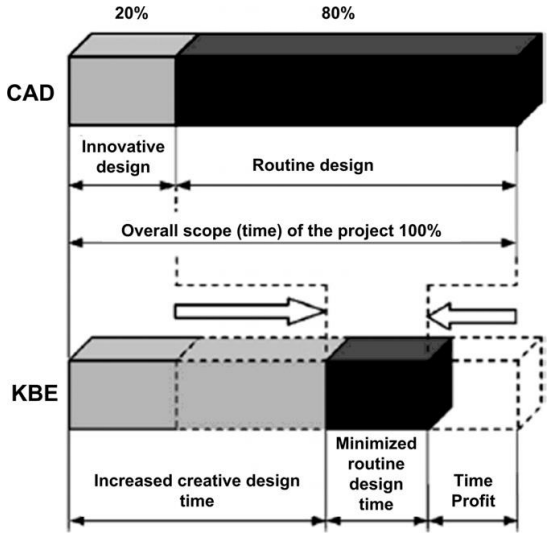


Figure 1-1: KBE vs. Traditional CAD (Skarka, 2007, Pg 678)

The creation of an informal model is the first step in knowledge modelling of an engineering design process and is considered to be one of the most critical step in developing a KBE system (KBES) (Pinfold et al., 2008). The most integral purpose of creating the informal model is to formulate neutral formal representation of the knowledge model for machine interpretation, which can assist the designer for achieving DEA as well as the knowledge engineer for developing automation application using a KBES. The abstraction of engineering knowledge in context of KBE environment can be stated as 'the process in which the engineering and design knowledge is structured and analysed for being represented in terms of objects and engineering rules in a computer understandable language or code'

(Bermell-Garcìa et al., 2001). As KBE as a design method allows inclusion of both product's geometric and non-geometric knowledge for design, analysis, manufacturing and design process decision making, knowledge modelling through knowledge management techniques constitutes a major bottleneck. The challenge is to perform the abstraction in a neutral (open standards) format with semantic clarity to ensure re-usability of the domain knowledge of engineering design process for DEA (Jubierre and Borrmann, 2015). Open standard usage becomes a key issue when the engineering design knowledge has to be transferred between different KBE applications (Bermell-garcia et al., 2007). Thus engineering design knowledge should be represented in open architecture as neutral representation for DEA in context to KBE approach (Penoyer et al., 2000; Zhang et al., 2009).

Various methods and techniques have been used for knowledge acquisition and representation in context to DEA with KBE approach. Some of them are used as informal representation for human interpretation and exchange of design process knowledge such as IDEFx (Integrated Definition for Functional Modelling), Model Based Systems Engineering (MBSE) methods such as Unified Modelling Language (UML) and Systems Modelling Language (SysML) for DEA. Other techniques such as W3C standards in the form of Web Ontology Language (OWL), RuleML and International Standards Organization (ISO) standards such as Process Specification Language (PSL) have also been investigated for machine interpretation of design process knowledge with axioms as formal representation for DEA. Investigation of existing KBE methodologies such as Methodology and tools oriented to knowledge-based engineering applications (MOKA), Knowledge Nurture for Optimal Multidisciplinary Analysis and Design (KNOMAD), Knowledge Capture Methodology (KCM), and Knowledge Oriented Methodology for the Planning and Rapid Engineering of Small-Scale Applications (KOMPRESSA) has revealed a few shortcomings to address DEA. Some of the identified shortcomings to enable DEA with KBE as a holistic approach are

neutral representation techniques of an engineering design process model with uniform axioms and preserved semantics that will allow usage across multiple platforms with open standards. The next sections will discuss the aims and objectives in order to address the existing challenges.

1.3 Aim and Objectives

This aim of this research is to provide a coherent method to develop platform independent and neutral formal representation of an engineering process model, with focus on mechanical product design process with manufacturing knowledge, and semantic clarity for DEA. This coherent method will capture various knowledge entities and relationships such as activity, product attributes, rule, function and behaviour as Meta Model, identified with literature analysis in an informal process model (for human aid and interpretation). The 2nd step will provide a method to represent the schema of the structured process model in neutral formal representation (for machine/system interpretation) with open standards for DEA with KBE as a holistic approach. This will include generative modelling capability by building queries as per a set of generic predefined functions. It will perform DEA with effect of the process model on product attributes with the help of inference (automated reasoning) and querying.

In order to achieve the aim, the following objectives have been developed–

1. To investigate different approaches for Design Engineering Automation (DEA) including CAx, PLM and KBE for product and process based automation.
2. To analyse and compare various informal and semiformal process modelling methods to capture various aspects of an engineering design process with focus on mechanical product design with design for manufacturing knowledge for automation.
3. To analyse and compare state of the art in existing formal representation (machine readable) techniques and standards.

4. To develop and build a detailed informal/semiformal process model with explicit relationships between identified knowledge entities of a mechanical product design process with design for manufacturing knowledge.
5. To formalise the process model in platform independent and neutral formal representation standards for DEA with semantic clarity. This will incorporate generative modelling capability by generating the activities, objects of the process and rules based on logic as per set of developed generic functions.
6. To perform experiments in order to validate and verify the process based knowledge model with its platform independent and neutral formal representation for re-usability, transparency and accuracy.

1.4 Research Method

The research method in order to meet the aim and objectives of this research is to hypothesise and test. The research hypothesis is described below.

1.4.1 Research Hypothesis

The hypothesis of this research work is -

> "Platform independent and neutral formal representation of an engineering design process model with focus on mechanical product design and manufacturing knowledge built on standardised concepts and relationships, structured and well defined axioms along with semantic clarity can achieve the requirements of design engineering automation (DEA) enabling generative modelling and re-usability of knowledge"

In order to test the research hypothesis a two-step strategy has been developed. The first step involves careful analysis of informal and semiformal modelling standards, which provide a coherent method of knowledge modelling of an engineering design process with focus on mechanical product design and manufacturing knowledge for automation. The second step

covers the schema mapping of the developed structured process based knowledge model, as a method in neutral formal representation with machine interpretable semantics enabling DEA with generative modelling.

1.4.2 Research Design

The research design consists of the following four building blocks – (1) Literature Review, (2) Use Case Collection and Analysis, (3) Development of process model as GPM-DEA and its implementation in Neutral Formal Representation Standards along with (4) Test for Transparency, Accuracy and Reusability of Knowledgebase. It is illustrated in Figure 1-2.

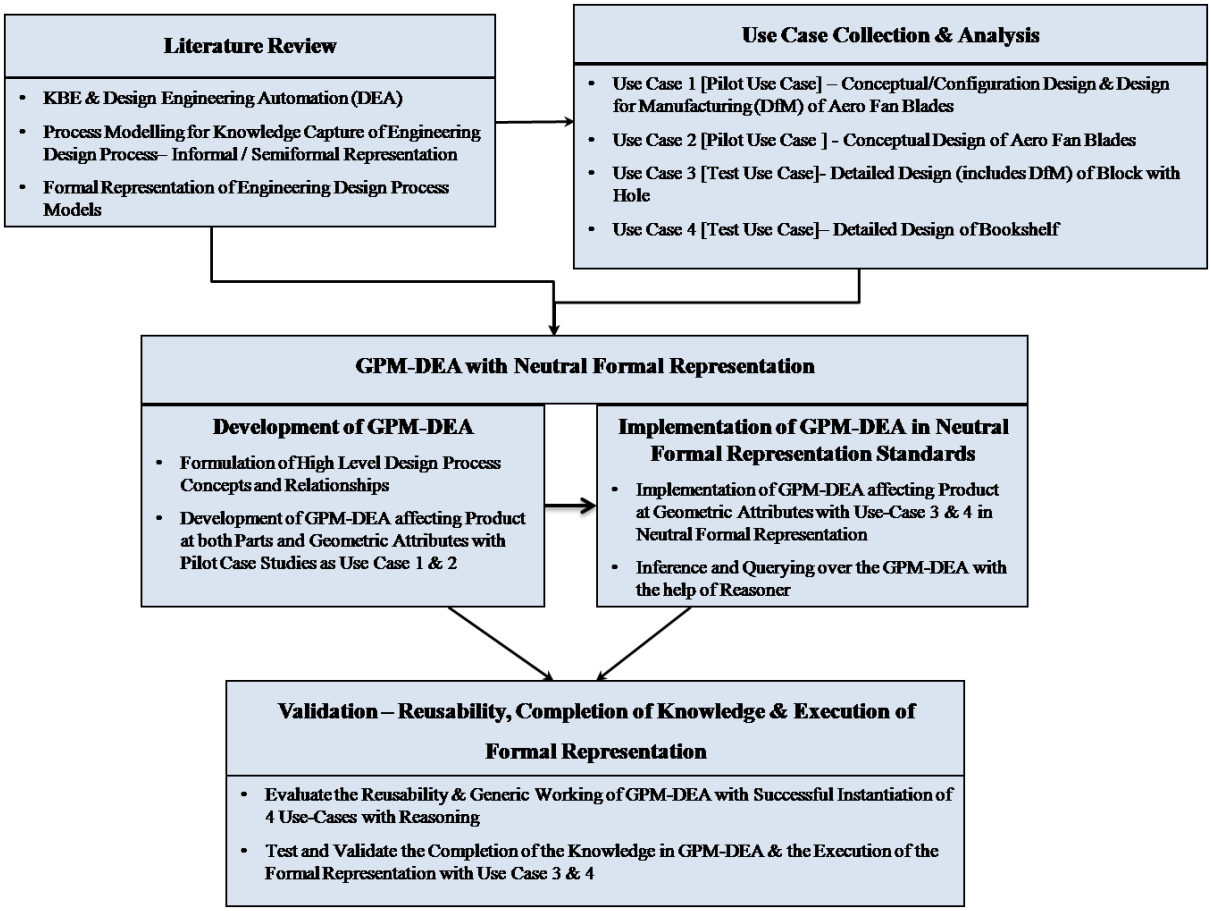


Figure 1-2: Research Design

In order to address the research gap, literature review and analysis, use case collection and analysis along with comparative analyses has been the main cornerstone for the development work with experimental verification of the developed process model. The literature review consists of broadly three topics – DEA and KBE in context to DEA with knowledge entities

for engineering design, informal process modelling for knowledge capture of an engineering design process and formal representation standards for representing an engineering process model at a system level for machine interpretation instead of natural text for human interpretation.

Use cases 1 and 2 have been used as pilot case studies. Use cases 1 has been collected from the industrial partner from SILOET2 grant such that knowledge can be accessed in the form of engineering design intent for mechanical design and design for manufacturing (DFM) aspects with manufacturing guidelines of jet engine fan / compressor blades spanning conceptual and embodiment design stages. Due to commercial sensitivity, the analysed knowledge in Use-case 1 won't be shared in detail. Use case 2 has been compiled from the literature review and includes jet engine fan blades conceptual design stage. All the Use cases have helped in the initial development of process model as GPM-DEA. Test use cases 3 and 4 have been compiled with the help of literature review and analysis. Use case 3 has been devised in terms of creating a hole in a block to test the effect of GPM-DEA at product's geometric attributes for virtual representation in detailed design stage. Similarly, Use case 4 has been devised and analysed from literature review with added knowledge in terms of bookshelf design process as a KBE method from LinkedDesign project. Both Use case 3 and 4 have been validated with the help of reasoner/inference and query as execution results, which are also compared with specific rule implementation in KBES. Targeting DEA for mechanical product design process with Design for Manufacturing (DFM) knowledge, GPM-DEA through its neutral representation format will also enable generative modelling with the aid of developed generic functions for query and reasoning and allow for ease of exchange and re-usability of knowledge.

Qualitative methods is adopted for data collection and analysis from use-cases based on document data in order to fully comprehend the research problem and develop an initial

prototype of the process model (Creswell, 2003). Comparative analysis has been performed for both informal (natural language) and formal (machine interpretable) standards for developing the process model and its implementation in neutral formal standards along with results from pilot use-cases (Rihoux and Ragin, 2009). The method of schema mapping of engineering knowledge from informal/semiformal process model to formal representation with preserved semantics and experimental verification to test the research hypothesis follows an ontology development approach by (Noy and McGuinness, 2001). The method of system development and experimental verification with test use cases using ontology development also aligns with engineering design optimisation and DEA with DFM aspects (Ahmed et al., 2007; Witherell et al., 2007). The proposed ontology development method aligns its principles with the research aims and objectives and aids in the verification by testing DEA capabilities with the help of supporting tools such as Protégé and Topbraid with assertion of axioms and reasoning / inference and query capability.

1.5 Research Scope

This research thesis is part of a larger research project, Platform Independent Knowledge Model (PIKM), where the initial case selection is based around the SILOET 2 grant as access to materials and experts can be built into the project. Two steps are critical in development of the process model in context to this research –

1. The first step is the structured knowledge modelling of domain knowledge as informal process model. The scope of the knowledge modelling as part of pilot use-cases is mechanical design along with DFM process of the compressor blades. The knowledge modelling is initially performed from the existing technical documentation of the design intent or specification of the industrial partner such as design rules and aids along with materials and mechanical methods as the collated knowledge. The collated knowledge is the raw and unstructured informal knowledge. This is followed by the

breakdown of the collated knowledge as per the engineering design process knowledge entities comprising of activities, input and output relationships, functional requirements and behaviour, constraints/rules, logic and product knowledge for efficient product realisation in the form of topological and geometric configuration along with manufacturing processes and rules. This is an integral step of knowledge analysis for developing a generic process model.

2. The second step is the structured method of schema mapping of the developed process model to platform independent and neutral formal representation. The platform independent and neutral representation with preserved semantics outside of a CAx system should enable DEA with KBE aspects such as generative modelling with the aid of suitable reasoning and query method.

Thus, this constrains the focus of this research. In order to meet the aims and objectives, the research scope includes the following aspects –

1.5.1 Design Engineering Automation (DEA)

All virtual engineering approaches for DEA such as CAx tools; PDM/PLM systems, workflow automation and KBE have been discussed. Various knowledge entities of an engineering design process with focus on mechanical product design and DFM knowledge in context to automation have been elaborated along with knowledge modelling methods.

1.5.2 Informal Process Modelling

Discussion and analysis of various informal (natural language) modelling methods has been performed in context to knowledge modelling of engineering processes. This includes methods such as IDEFx series, Design Structure Matrix (DSM), Business Process Modelling Notation (BPMN), Signposting, Role Activity Diagram (RAD) along with semiformal modelling languages in the form of Model Based Systems Engineering (MBSE) standards

such as UML, SysML to represent complete domain knowledge of a mechanical design process with manufacturing knowledge for automation with KBE as a holistic approach.

1.5.3 Formal Representation

This includes detailed analysis of formal (machine interpretable) representation in the form of Frames and Frame based languages, Description Logic (DL) and First Order Logic (FOL), Schema based representations and Object Oriented (O-O) languages. It also include discussion and analysis of ontology languages such as PSL, OWL, IDEF5 and rule languages such as RuleML, Rule Interchange Format (RIF) and Semantic Web Rule Language (SWRL). The informal or semiformal process model aspects for DEA should map onto suitable formal standards. The formal representation framework will have well-defined syntax, axioms and semantics and will be compliant with International Standards for process exchange and product model definition (Grüninger and Menzel, 2003; Pouchard et al., 2005). The execution of neutral formal representation layer will be similar to the functioning of a KBES.

1.5.4 Development of Process Model

After careful analysis of existing knowledge modelling standards based on standardised engineering concepts and relationships, along with review of an integrated approach of existing methods with modifications, a Generative Process Model (GPM-DEA) has been developed. This has been developed with the aid of literature analysis, industrial and literature based pilot use cases and comparative analysis of informal/semiformal standards as per the requirements of DEA for mechanical product design process with DFM knowledge. This is performed using DrawIo tool, which supports UML/SysML, and IDEF0 constructs along with additional concepts and relationships.

1.5.5 System Development - Neutral Formal Representation of Process Model

The implementation of process model in platform independent and neutral representation as system development has been performed after comparative analysis of formal representation

standards based on the requirements of DEA in context to KBE. The results have shown OWL/SWRL as a suitable candidate. The ontology development is performed using Protégé (Horridge et al., 2011) which supports both OWL/SWRL and Topbraid (Composer, 2011) for OWL.

1.5.6 Experimental Verification

Experimental verification of the developed system of process model with ontologies as neutral formal representation has been performed with the aid of test use-cases. Important aspects include testing of generic working, re-usability and traceability of concepts and relationships such that the coherent and structured model can represent complete domain knowledge of a mechanical design process with DFM knowledge for automation. The other aspect involves the accuracy of DEA capability with generative modelling of the detailed process model through reasoner/inference and query results based on set of developed generic functions using SWRL and its comparison with specific rule implementation in platform specific DEA system / KBES such as Advanced Modelling Language (AML) and ParaPy.

1.6 Thesis Structure

Figure 1-3 illustrates the outline of the thesis. The thesis is divided in 8 chapters. Chapter 1 is the introduction. Chapter 2 provides existing literature review with overview of DEA and various methods in virtual engineering for DEA. It leads to the identification of the research gap, which this thesis addresses. Chapter 3 then elaborates on literature analysis with informal and formal standards for knowledge modelling of engineering design process. This leads to refinement of the research gap. Chapter 4 addresses the novel aspects of this research with the help of pilot use-cases collated from both industry and literature. It discusses experimentation of neutral formal representation languages including ontology language such as PSL and OWL, rule languages such as RuleML and existing MBSE languages such as

UML, SysML with design process knowledge entities. The generative process model for DEA in the form of GPM–DEA is then developed with the help of pilot use-case analysis and compiled requirements of DEA. Chapter 5 discusses the implementation of GPM-DEA schema in OWL/SWRL as the developed method of using ontologies. This is performed as per the comparative analysis of neutral formal representation standards against the requirements. Chapter 6 compiles two additional test use cases from literature for instantiation in GPM-DEA and in OWL/SWRL as proof of concept. Chapter 7 illustrates the experimental verification by testing the reasoning and inference accuracy of the developed system with the help of Protégé as supporting tool. The results are compared with the implementation results of use-case rule outputs inside platform specific DEA system such as AML and ParaPy. Chapter 8 presents the final conclusion based on discussion, contribution to knowledge, limitations and possible extensions for future research.

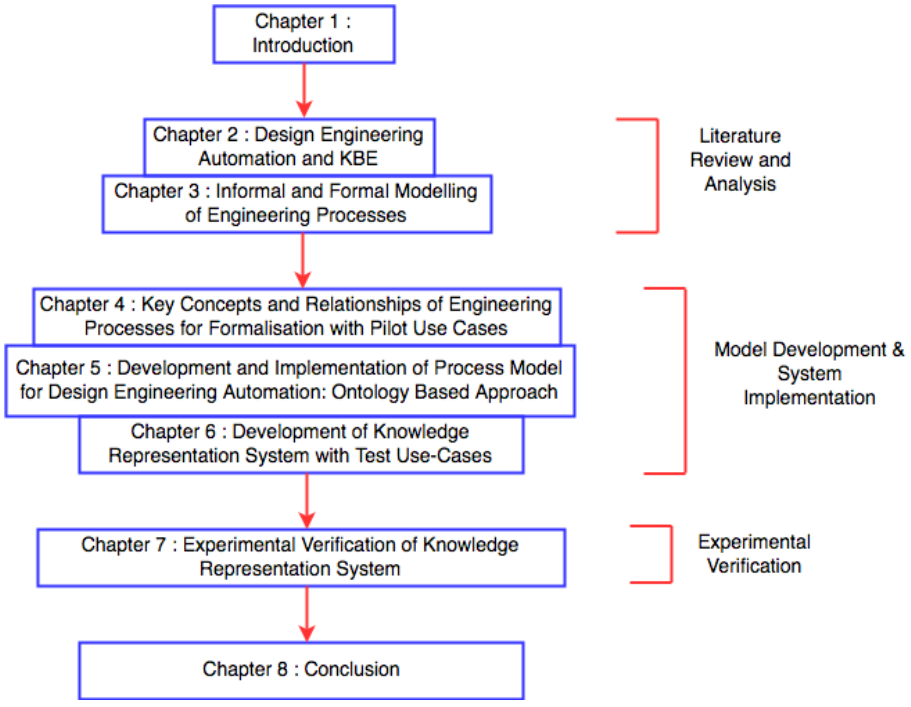


Figure 1-3: Thesis Structure

2 Design Engineering Automation and KBE

2.1 Introduction

The goal of the Product Development (PD) process is to transform customer requirements into a physical product. A robust PD process needs to achieve optimum product performance and quality with short lead-time to market and reduced costs (Ulrich and Eppinger, 2012). Design Engineering Automation (DEA) techniques greatly help solve this purpose. In order to enhance PD efficiency with DEA, virtual engineering methods were adopted by industries worldwide (Bernard, 2005; Zhang et al., 2010). It consists of various domains such as CAx consisting of Computer Aided Design (CAD)/Computer Aided Engineering (CAE)/Computer Aided Manufacturing (CAM); information systems such as Product Data Management (PDM)/Product Lifecycle Management (PLM), decision support tools and KBES, with all systems represented in heterogeneous formats. However, it was realised that interoperability is one of the key areas of improvement in order to prevent compatibility issues between different virtual engineering applications and file formats (Bernard, 2005). In order to provide holistic and complex DEA, the scope of neutral representation of engineering design process for interoperable product realisation in context to KBE has been recognised. This chapter provides an overview of various virtual engineering aspects as part of PD. This will lead to discussion on DEA and KBE with existing models for DEA.

2.2 Engineering Design Process for Product Development

Product Development (PD) process can be stated as ‘a sequence of activities that an organization follows in order to conceptualise, design and manufacture a product commercially’ (Ulrich and Eppinger, 2012). PD process can be divided broadly into five stages as per Ulrich (Ulrich and Eppinger, 2012) – ‘requirements analysis & conceptual design, systems development & configuration design, detailed design, testing & refinement,

and production'. The PD process consists of engineering design, analysis for testing and manufacturing which span all these five stages. The first stage of the engineering design process is the identification of customer requirements, which are then translated into functional requirements of the product as design specifications or design intent. The functional requirements drive the engineering design process which are used to specify product profiles utilising engineering knowledge and creative thinking (Chen et al., 2008).The engineering design process is considered to be a set of comprehensive and knowledge intensive activities depending upon existing engineering knowledge which consists of bothdesign and manufacturing knowledge (Chen et al., 2008; Peng et al., 2017).According to Pahl and Beitz (Pahl et al., 2007), engineering design is very complex and requires a very systematic approach. An engineering design process for PD can be subdivided into various categories such as conceptual design, embodiment or configuration design and detailed design (Pahl et al., 2007; Ullman, 2010; Zeng and Gu, 1999). All the stages are described as follows -

2.2.1 Conceptual Design Stage

The conceptual design stage encompasses high-level concepts to meet design specifications or design intent as requirements (Pahl et al., 2007; Zeng and Gu, 1999). Concept generation is very crucial at this stage (Ullman, 2010). Conceptual design process includes basic building of physical structure of the product guided by design specifications as functional requirements of the design process or product's function (Qin et al., 2003; Viola et al., 2012). The analysis of functional requirements or product' function is very crucial at this stage as design specifications can be highly abstract (Wang et al., 2002). Division of functions as sub functions can be achieved by various ways such as brainstorming (Ullman, 2010). Some examples of function are - "increase pressure, transfer torque and reduce speed" (Pahl et al., 2007, pg 31).

2.2.2 Embodiment Design Stage / Configuration Design Stage

The configuration design stage or embodiment design focuses on the refinement of initial concepts to product configuration at the component and subcomponent levels along with the development of design parameters. It greatly assists the designer in concept evaluation and selection. It also helps in technology readiness by identifying critical parameters (Ullman, 2010). Various methods such as Pugh's (Pugh, 1991), decision matrix can be used at this stage for risk and feasibility analysis of generated concepts.

2.2.3 Detailed Design Stage

The detailed design stage focuses on the development of detailed parameters of the product architecture and structure such as form with the assistance of geometric dimensions and tolerances, fit as components with parts and assemblies, features and material allocation (Pahl et al., 2007; Zeng and Gu, 1999). Product evaluation is very critical at this stage before proceeding to the manufacturing stage (Ullman, 2010). Product evaluation involves performance analysis of product's function such as electrical energy, mechanical energy and thermal energy within the prescribed boundary conditions.

The boundary between all stages of the design process overlaps due to the iterative nature of the design process. Design for manufacturing (DFM) is very crucial stage of the design process and mainly comes under configuration or embodiment design although it can be considered at conceptual design and detailed design stage as well. DFM includes manufacturing and production feasibility, lifecycle and quality aspects (Wuest et al., 2015). Thus it includes manufacturing knowledge as feedback or inputs at the design stage, which may include manufacturing processes for example machining processes such as welding, drilling and processes such as moulding, casting for the product specification. Similarly, design for assembly (DFA), design for ergonomics, design for recycling are some of the other crucial aspects of the embodiment or configuration design stage (Pahl et al., 2007). All

techniques such as DFM, DFA, design for ergonomics, and design for recycling are part of DFX techniques for improved productivity of the engineering design process (Elgh, 2006).

Collection and representation of design knowledge with the help of computing is crucial for all these phases. Computer based data processing in the form of Computer Aided Design (CAD) is very prevalent among designers. However, routine tasks should also be taken by a computer to allow designers to focus on new design tasks. The development of knowledge based systems (KBS) for engineering design can be used as a computer tool for knowledge modelling and retrieval, which should incorporate both design process and product knowledge. These systems assist the designer in analysis and optimisation of solutions by providing decision-making capability (Pahl et al., 2007). It is widely acknowledged that for knowledge storage and re-use for engineering design, capture and representation of abstract forms such as high level concepts and function are extremely beneficial for design evaluation and rapid retrieval of knowledge as query for archive designs as well as the complete design lifecycle from conceptual design to detailed design (Andrews et al., 1999; Ullman, 2002). Thus, capture, representation and querying of design intent will greatly improve the efficiency of the engineering design process as part of PD. Functional requirements are very crucial to generate artifacts for design optimisation and evaluation process (Roy et al., 2001). This includes non-geometric knowledge pertaining to the conceptual design and configuration design stage affecting the topology of the product from the functional requirements as goals of the design process. Thus, a suitable domain specific model of engineering design should link decomposed functional requirements of the engineering design process to the form of the product (Roy et al., 2001).

Figure 2-1 illustrates various stages of design lifecycle with representation methods for engineering knowledge.

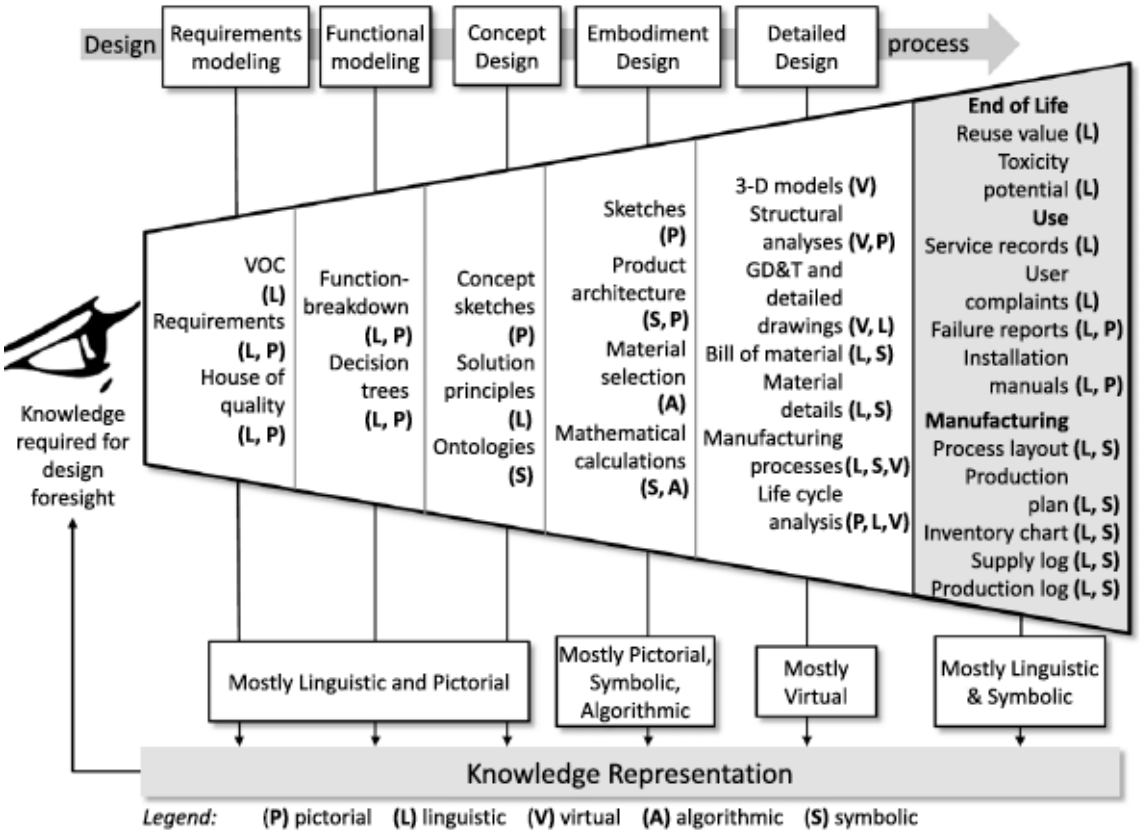


Figure 2-1: Stages of Engineering Design with Knowledge Representation Methods (Chandrasegaran et al., 2013, Pg 208)

2.3 Product Development: Advancement with Virtual Engineering

Virtual engineering helps transform the physical engineering design process in virtual system domain at all stages of the product lifecycle. The engineering design, analysis process and the manufacturing process as part of PD are realised in the virtual world with CAx tools (platforms) such as CAD, CAE and CAM (Frank et al., 2014; Shintre and Shakir, 2011). CAE generally consists of CAD and CAM tools along with analysis of CAD models such as structural analysis, fluid analysis, and thermal analysis depending upon product's functional requirements (Ćatić and Malmqvist, 2007).

2.3.1 CAD & Geometry Automation: Parametric Modelling

CAD tools allow building and visualisation of product's geometric shapes based on points, curves, surfaces, and volumes along with features (Bernard, 2005; Shyamsundar and Gadh,

2002). The underlying representation in most CAD tools is based on B-Rep and Constructive Solid Geometry (CSG). They also provide compilation of Engineering Drawings (ED) and Bill of Materials (BOM) thus illustrating the product structure (Shyamsundar and Gadh, 2001). CAD provides support during the configuration and most importantly detailed design stage of the PD process. Updates and enhancements in CAD tools such as CATIA Knowledgeware provide Geometry Automation (GA) (Amadori, 2012) through parametric modelling (Bodein et al., 2009), knowledge based CAD (Nomaguchi et al., 2004) and Knowledge Intensive CAD (KIC) (Tomiyama and Hew, 2000). Parametric CAD systems utilise Geometric Constraint Solvers (CSG) for parametric modelling (Jubierre and Borrmann, 2015). They mainly affect geometric attributes in the form of points, lines and circles as constraints. This allows for modelling of intelligent automation through variant design in terms of product' geometric parameters. However, the file format of these CAD tools enabling design engineering automation (DEA) are in proprietary formats and are still limited to shape and form variation (Frank et al., 2014).

2.3.2 CAE & Analysis

CAE tools allow the virtual simulation and analysis of 3D CAD models as geometric product representation (Ulrich and Eppinger, 2012). CAE includes processes such as finite element analysis (FEA) and computational fluid dynamics (CFD) analysis in the form of thermal analysis, flow analysis, stress analysis, aerodynamic analysis and kinematic analysis with CAD model as the master model (Tyapin et al., 2012). Some of the CAE operations include meshing and applying boundary conditions in order to perform accurate analysis in the form of preprocessing and postprocessing. CAE provides a major platform during testing and refinement stage of the PD process. The results of the CAE analysis models are evaluated as per the formulated functional requirements. However, the transition of geometric product model from CAD to CAE tools requires transformations due to heterogeneous file formats

between CAD and CAE systems (platforms) thus limiting the combined advantage (Corallo et al., 2009).

2.3.3 CAM & Manufacturing

CAM tools allow simulating and performing the production/manufacturing processes as physical processes for realisation of the product with the help of virtual environment (Corallo et al., 2009). CAM tools generally include tool paths such as CNC programming and machining operations, manufacturing methods, tool cutting data such as speed and feed, clamping, jigs and fixture strategy along with product's physical properties such as material, features, tolerances and surface finish (Helgoson and Kalhori, 2012). These may include operations such as milling, drilling and boring. CAM provides a major platform during the production stage of the PD process. However, the transfer of knowledge from the CAD/CAE stage to CAM stage is highly complex due to variation in platform representations (Corallo et al., 2009; Zhang et al., 2009).

Due to heterogeneity in CAx tools (platform) representations, there is loss of valuable knowledge. Thus there is a lack of coherent engineering design knowledge for PD process in a multidisciplinary environment, which can be re-used (Zhang et al., 2009). In order to overcome the loss of knowledge in CAx tools, various environments have been devised as part of Concurrent Engineering (CE). Concurrent Engineering (CE) provides utilisation of varied knowledge inputs simultaneously to speed up PD process by integrating down-stream processes such as analysis and manufacturing in the early stage of engineering design (Chapman and Pinfold, 1999). One example is Computer Integrated Manufacturing Environment (CIM) to overcome the loss of knowledge between CAD ad CAM systems due to lack of neutral formats as well as the overall content of CAD knowledge (Natekar et al., 2004). CIM allows for feature recognition in order to convert from product form and shape in CAD to manufacturing process planning in CAM using a neutral format such as Initial

Graphics Exchange Specification (IGES). The feature recognition algorithm is written on top of IGES for extraction of CAD features to CAM operations. Advancements in proprietary systems such as Unigraphics solutions UG NX5 also provide integrated CAD, CAM and CAE systems as part of CE, thus providing a unified platform for engineering design with rich semantic product data knowledge for cross functional PD (Liu et al., 2010).

Thus in order to facilitate efficient knowledge transfer between CAx tools, neutral formats such as IGES and STandard for the Exchange of Product model data (STEP) should be utilised. STEP retains most of the product's model knowledge while transfer between different CAD platforms (Pratt, 2001). However, these neutral formats in the form of STEP mainly represent product's geometric knowledge pertaining to detailed design and manufacturing including 3D model but don't contain other aspects of engineering design process knowledge covering all aspects of product's lifecycle (Främling et al., 2012).

2.3.4 Product Data Management (PDM)/Product Lifecycle Management (PLM)

Other attempts in the field of virtual engineering to overcome the loss of knowledge between different CAx tools are Product Data Management (PDM)/Product lifecycle Management (PLM) systems. According to John Stark, PLM systems can be defined as follows – 'PLM is the business activity of managing a company's products all the way across their lifecycles,from the very first idea for a product all the way through until it is retired and disposed of, in the mosteffective way' (Stark, 2011). The initial versions of PLM systems were generally referred as PDM systems. Generally PDM systems allow integration of disparate knowledge between various CAD, CAE and CAM tools (Bruun et al., 2015; Ćatić and Malmqvist, 2007). They consist of product geometry knowledge, assembly and functional relations, analysis and manufacturing knowledge depending upon their configuration. PLM is like an extension to PDM facilities for more comprehensive coverage and can also provide workflow automation. Some examples of PDM/PLM systems are

TeamCenter PDM, Collaborative product development (cPDM) and virtual product development (VPDM) (Bruun et al., 2015). Similarly, other versions of PLM systems to address the needs of CE are Product Life Cycle Systems (PLCS), which provide integration of CAD, CAM, CAE and Product Information Management (PIM) systems thus allowing coherent flow of knowledge for collaborative environment (Penoyer et al., 2000).

2.3.4.1 Workflow Automation

Workflow automation in context to PLM systems can be performed in tools such as Isight (H Wenzel et al., 2011). Workflows and their execution logic can be shared and exchanged between heterogeneous design platforms as platform independent representation using neutral format in the form of eXtensible Mark-up Language (XML). The building blocks of the simulated workflows are individual components such as the object parameters, sub processes and connected components (H. Wenzel et al., 2011).

2.4 Design Engineering Automation - CAx, PDM/PLM

2.4.1 Design Engineering Automation (DEA)

As illustrated, various CAx and PDM/PLM systems fulfill automation techniques for engineering design process. Automation methods satisfy the following objectives as part of a PD process - reduce lead-time and costs, improve quality of products and provide variation in product design process as per changes in customer requirements (Cederfeldt and Elgh, 2005). Design Engineering Automation (DEA) can be defined as capturing and formalising engineering design knowledge consisting of a set of building blocks for automated design and PD processes for satisfaction of customer requirements (Frank et al., 2014). DEA provides added value by optimisation of PD process and incorporating all types of knowledge for automation including both product's geometric knowledge as well non-geometric knowledge in the form of engineering design process knowledge.

According to work performed by Cederfeldt & Elgh, DEA refers to 'Engineering support by implementation of information and knowledge in solutions,tools, or systems, that are pre-planned for reuse and support the progress of thedesign process. The scope of the definition encompasses computerized automation oftasks that directly or indirectly are related to the design process in the range ofindividual components to complete products' (Cederfeldt and Elgh, 2005, Pg 2). DEA can be categorised into two types – information handling (knowledge representation and retrieval with inference or automated reasoning) and knowledge processing (Elgh, 2008; Nan and Li, 2012).

The purpose of DEA is to provide support in following areas (Elgh, 2008, 2007) –

- Design synthesis - this includes optimisation of design parameters and product geometry and decision support for engineering design with the assistance of functional requirements and manufacturing constraints
- Design analysis - this includes model analysis for testing such as finite element analysis, geometry preparation for analysis in the form of meshing, preprocessing and post processing and evaluation of design characteristics
- Plan for manufacture – this includes manufacturing processes for physical production of the designed parts and components. This may include production methods, sequence of operations and tooling description such as fixture and jigs

2.4.2 CAx, PDM/PLM for DEA

All CAx tools such as CAD, CAE, CAM along with PDM/PLM systems comprise main virtual engineering applications for engineering design process and enabling some aspects of DEA (Ćatić and Malmqvist, 2007). However, there is a difference in the knowledge content of CAx tools such as CAD systems and PDM/PLM systems. The main strength of CAx tools specifically CAD systems is the representation of product's geometric knowledge as part of detailed design stage which is a narrow part of PD in its proprietary platform representation

(Foufou et al., 2005). On the other hand, PDM/PLM systems are tailored to represent product's non geometric knowledge in its proprietary format such as requirements analysis, product function and behaviour and are used as linking or information management system for product's geometric knowledge by documenting and managing CAD files, drawings, CAE and CAM files along with product related documents in different computer formats (Bruun et al., 2015; Burkett et al., 2003). The most comprehensive usage of PDM/PLM as database management systems is a common platform for knowledge access and integration across various CAx tools and product definitions across different formats (Penoyer et al., 2000).

Thus, one of the limitations of this existing virtual engineering approach is lack and ease of integration of geometry kernels as part of CAD systems within a unified PDM/PLM system representation for DEA (Penoyer et al., 2000). Also, individual automation applications such as workflow automation using Isight and excel based macros for specific purposes are very rarely linked to CAx tools or PDM/PLM systems. Another cause of concern of these individual virtual engineering approach is that the representation of individual CAE and CAM tools is specific, knowledge management is very rigid with respect to the underlying platform along with lack of an integrated, unified and structured approach for DEA (Ćatić and Malmqvist, 2007).

Another method of addressing the needs of DEA is solved through Knowledge based Engineering (KBE). As illustrated with the help of Fig 2-2, all CAx tools such as CAD, CAE and CAM along with PDM/PLM systems provide small isolated islands of DEA in context to a KBE approach, which provides an integrated and unified approach for DEA (Ćatić and Malmqvist, 2007).

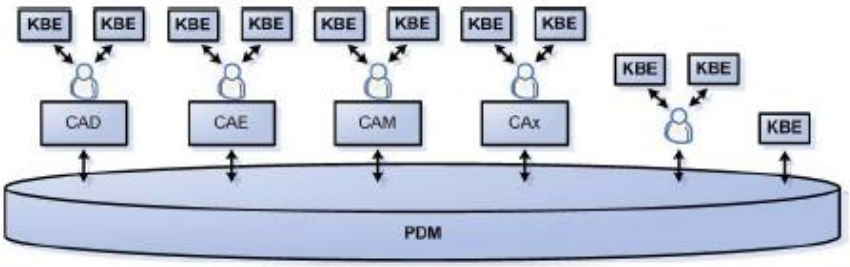


Figure 2-2: KBE with respect to CAx, PDM/PLM (Ćatić and Malmqvist, 2007, Pg 1)

2.5 Knowledge Based Engineering (KBE)

'Knowledge Based Engineering (KBE) represents an evolutionary step in Computer-Aided Engineering (CAE) and is an engineering method that represents a merging of Object-Oriented Programming (OOP), artificial intelligence (AI) and Computer-Aided Design (CAD) technologies, giving benefit to customised or variant design automation solutions' (Chapman and Pinfold, 2001, Pg 905). One of the main objectives of KBE systems is to reduce the time and cost of product design lifecycle by automating repetitive and non-creative design tasks (Cooper and LaRocca, 2007; Sandberg, 2003).

A few examples where usage of KBE technology has led to a decrease in product design life cycle time are demonstrated as follows. In the automotive domain, Chapman and Pinfold utilised a KBE system (KBES) in the form of Advanced Modelling Language (AML) (TechnoSoft Inc, 2003) for automation of geometry creation and finite element (FE) analysis process using meshing and applying boundary conditions (Chapman and Pinfold, 2001; Pinfold and Chapman, 2001). Pertaining to the aerospace domain, a KBES was employed by La Rocca and Van Tooren for automation and generation of blended wings and low pressure turbines (La Rocca and Van Tooren, 2007).

The focus of KBE is knowledge capture and representation of both geometric and non-geometric knowledge to enable product and process centred automation for all stages of the engineering design process lifecycle including conceptual design, embodiment design and

detailed design including the manufacturing phase (Corallo et al., 2009; Prijic et al., 2005). Traditionally, in design engineering the output of the preliminary and the detailed design is in the form of a geometric CAD model directly created from requirements or problem definition.KBE as a design method captures product and process-based data and helps in building a virtual prototype in a system which encapsulates rules, requirements, product attributes, features and rationale for building a geometric model along with downstream processes such as material selection for static and dynamic analysis, and manufacturing capability enabling complex design automation. It enables generative modelling along with feature based parametric modelling and reasoning mechanism by acting as an expert system (Cooper and LaRocca, 2007; La Rocca and Tooren, 2012). KBE adds a major dimension also referred to as product decomposition (Calkins et al., 2000) and helps in developing a complete repository of design engineering knowledge for efficient product design & realisation process.Thus it gives options to the designer to test the geometric model for realisation more efficiently due to the availability of a complete knowledgebase.

A system implementation of KBE can be defined as 'the use of dedicated software language tools in order to capture and re-use product and process engineering knowledge in a convenient and maintainable fashion' (Cooper and LaRocca, 2007). A system implementing KBE is dynamic such that it offers true engineering automation including application development, geometric modelling, application deployment and tools integration (Calkins et al., 2000).

2.5.1 KBE and CAx, PDM/PLM for DEA

KBE as an area of artificial intelligence (AI) provides a unified and integrated approach for complex DEA and effectively combines CAx and PDM/PLM automation facilities along with assistance in knowledge re-use and decision making (Zhang et al., 2009). The formal representation in a KBES performs DEA with various reasoning mechanisms, which can vary

such as rule-based reasoning and case-based reasoning. A successful KBE implementation depends upon various stages – knowledge acquisition, knowledge representation and reasoning (Zhang et al., 2009). Thus, KBE offers enrichment of CAD models with non-geometric knowledge and also assists knowledge management (KM) with knowledge acquisition and representation of engineering design knowledge (Cooper et al., 1999). One of the distinct advantages of KBE approach towards DEA is generative modelling capability, which ensure that engineering design knowledge is generated as instantiated data from requirements analysis by explicit declaration of codified knowledge. It also offers multiple view-points such as design, analysis, manufacturing, ergonomics within a unified environment (Bermell-garcia et al., 2007).

Thus KBE tools capture design rules with much higher granularity in contrast to PDM/PLM systems as they combine the knowledge content of CAx and PDM consisting of both geometric and non-geometric knowledge. The most important aspect of a KBE approach for DEA is the integration of geometry kernel closely integrated with non-geometric knowledge (Bermell-garcia et al., 2007; Ćatić and Malmqvist, 2007). This includes product's form and geometry in the CAD environment, topological variation in product design with both parametric and generative modelling and non geometrical knowledge which is generally contained in PDM systems thus providing a systematic approach for knowledge acquisition, re-use for automation and efficient decision making (Sandberg et al., 2017; Sorli et al., 2012).

Generative modelling is one of the most important aspects of DEA with KBE approach. KBE as a design method not only enables generative modelling at the detailed design stage with GA but also at the conceptual and embodiment design stage (Isaksson, 2003). This provides flexibility in design variation at all stages of engineering design, which is not possible with any CAx tool. The generative modelling capability captures both product

and engineering design process knowledge and can be used for DEA and design evaluation based on the product's functional requirements 'on the fly' basis (Isaksson, 2003).

Thus, KBE approach combines capabilities of CAx tools and PDM/PLM systems for complex DEA enabling knowledge re-use and decision making in a modular and integrated environment (Isaksson, 2003).

2.5.2 Achieving DEA with KBE – Integral Features

Traditional DEA approaches follow procedural style of programming where the knowledge or the design intent from a source is hard coded and integrated to a system or an application (Prasad, 2006). In procedural programming style the sequence of steps has to be explicitly mentioned. However, KBESoffer slightly different approach to conventional DEA. They follow declarative style as against a purely procedural style (Cooper and LaRocca, 2007; Prasad, 2006). This means that the sequence of steps for a process in the form of design intent doesn't need to be explicitly mentioned during execution. The system or the application will automatically determine which activity to implement based on the requirements. KBES offer functional coding style which states that the code returns values to the user instead of simply modifying or updating the model (Cooper and LaRocca, 2007). KBES follow Object-Oriented(O-O) representation with high probability of embedding LISP based dialects (Cooper and LaRocca, 2007; La Rocca, 2011; Rocca, 2012)along with being dynamic, which means the formal design intent will update, and new concepts and relationships are inferred as changes at runtime. Conventional DEA is not dynamic in nature.

Some of the integral features of KBES are listed as follows -

- Functional and declarative style as opposed to pure procedural style in conventional DEA – it supports both declarative and procedural paradigm
- Dynamic data types

- Runtime value caching and Dependency tracking
- Demand driven
- Generative modelling
- Tight linkage with geometry
- High level indicating that a small amount of code enables manipulation of large number of objects as being opposed to problem specific. This enables generic and re-usable code with instances as compared to limited re-use of hard coded knowledge in conventional DEA
- Knowledge models are enriched with process and product knowledge as compared to specific domain in conventional DEA

(Cooper and LaRocca, 2007; Prasad, 2006, 2005; Rocca, 2012; Van der Velden, 2008)

There are a lot of crucial differences between CAD centered automation applications such as CATIA Knowledgeware and Siemens NX Knowledge Fusion and pure KBE based DEA. CAD centered automation lays emphasis on alteration of product models primarily through geometric features based approach for pure geometry automation (GA). CAD based automation doesn't include function and behaviour of the product (Kopena and Regli, 2003; Umeda and Tomiyama, 1997). Furthermore, the design intent in a parametric CAD for GA doesn't capture the complete design intent (Ullman, 2002). This reduces the creativity for innovation based on 'what-if' analysis of the design intent (Jubierre and Borrmann, 2015). This differs from the pure KBE based DEA through a knowledge based approach (Colombo et al., 2014; Prasad, 2006)where knowledge is managed through high level of abstraction encapsulating engineering rules based on logic, product structure, function and behaviour. KBESor KBE applications can deal with both geometric and non-geometric knowledge of the product as part of the system and can incorporate design process knowledge (Prasad, 2006;

Skarka, 2007). CAD based automation applications like CATIA Knowledgeware and Siemens NX Knowledge Fusion do provide purely GA facilities through a platform specific code but don't enable KBE features such as full generative modelling, dynamic data typing, run time caching and dependency tracking (Cooper and LaRocca, 2007). For example, CATIA v5 Knowledgeware uses C++, visual basic and component application architecture (CAA) language in order to provide GA facilities (Prijic et al., 2005). In the research of Lin (Lin et al., 2013), CATIA was used as a platform to enable parameter based design space exploration and automation by providing variable input parameters to the geometric form of the product. In case of a change of system the platform specific code will have to be re-written, thus limiting the abstraction and re-usability of the engineering process knowledge along with increased maintenance (Sanya and Shehab, 2014).

With recent advancements, web based approach has gained acceptance in the KBE community for information sharing and exchange (Liu and Xu, 2001). It also offers advantages such as open architecture, uniformity in information modeling and O-O structure (J Kulon et al., 2006).

2.5.3 KBE lifecycle and Methodologies

According to Stokes, KBE lifecycle consists of the following 6 stages(Stokes, 2001) –

- Identify: Identification of technical and business requirements for DEA for providing an initial specification of a KBES
- Justify: Assessment of existing processes for implementation of KBE for DEA benefits and risk analysis
- Capture: Knowledge acquisition in the form of input of engineering design process and product knowledge collected from domain experts, documents such as design guidelines and manuals for conversion to structured formal representation. It caters to the needs of the domain experts for validation of domain knowledge as informal model

- Formalise: Development of a framework for conversion of the structured captured informal knowledge to a formal representation model (machine readable for system interpretation) with neutral semantics for interoperable usage through open standards. This ensures re-usability of the domain knowledge as neutral formal representation
- Package: The neutral formal model is used for compilation and execution as the source code in a KBES or KBE application. This phase covers the transformation of the neutral formal representation to the platform specific representation inside the KBES. In order to validate the functioning of a KBES, running queries and reasoning as execution of the source code is performed to demand data from the KBES source code
- Activation: Verification of the installation of KBES for multiple users. Documentation, training support and infrastructure may be provided for effective deployment within the organisation

There are various methodologies for implementing KBE. A methodology termed as Knowledge-Oriented Methodology for the Planning and Rapid Engineering of Small-Scale Applications (KOMPRESSA) with its diagrammatic ways of capturing knowledge in the form of a component diagram was initiated for smaller KBE applications (Bancroft et al., 2000; Chapman et al., 2007). In Knowledge Capture Methodology (KCM), capturing and structuring of knowledge is performed from a designer's point of view. It breaks down the product knowledge into parts, assemblies, features and the relationships between the geometric features and the components to formulate product semantics (Chapman et al., 2007; Terpenny et al., 2000). Both KOMPRESSA and KCM were targeted for product modelling and automation. Knowledge Nurture for Optimal Multidisciplinary Analysis and Design (KNOMAD) as a methodology laid emphasis on activity diagrams for processes and representation of multidisciplinary knowledge including design and manufacturing (Verhagen et al., 2012).

Methodology and Tools Oriented to Knowledge Based Engineering Applications (MOKA) (Skarka, 2007) as a methodology was initiated for larger applications. It encapsulated both product and process based modelling. Rapid Application Development (RAD) (Beynon-Davies et al., 1999) is another methodology which directly encodes the knowledge on to an application with the help of packaging stage whereas other methodologies such as KOMPRESSA, KCM and MOKA build an independent knowledge book or the knowledge model external of the application and then map the knowledge model on to the KBES or a KBE application. This is the combination of formalise and package stage in the KBE lifecycle.

Thus RAD provides a quicker way of achieving an end KBE application by directly packaging the captured knowledge whereas other methodologies such as KCM, KOMPRESSA and MOKA are slightly more time consuming as they develop anindependent knowledge model with the help of formalise stage and then focus on translation to the end KBE application or KBES in the packaging stage. However, the advantage of developing an independent knowledge model is the translation of the independent knowledge model to multiple end KBE applications through its neutral formal representation enabling re-use of the domain knowledge both at human and system level.

Except for the fundamental difference between RAD and KCM, KOMPRESSA and MOKA which allow building of an independent knowledge model as compared to direct population of knowledge into the end KBE application in RAD, methods of capturing the structuring knowledge varies slightly between all three methodologies.Careful considerations should be adopted while implementing these methodologies such as the end KBE application should reflect continuous changes with the independent KBE model being the master model. Another methodology is referred as CommonKADS, which stands for Common Knowledge Acquisition and Documentation Structuring or Common Knowledge Acquisition and Design

Support (Schreiber et al., 2000). It defines six modules – organisation, task, agent, knowledge, communication and design models.

All these KBE methodologiesin the form of MOKA, KNOMAD, KCM, KOMPRESSA, CommonKADS and RAD offered major advantage in terms of abstraction and decomposition of knowledge in different forms, as discussed, with the help of Table 2-1, before the end KBE application development and provide more functionality to knowledge management. As observed from Table 2-1, all the methodologies have different ways of capturing data for knowledge modelling and aid in process improvement through diagrammatical and visual ways.All of these methodologies were successful only in knowledge acquisition and analysis stage for engineering design process improvement.

Table2-1: Existing KBE methodologies and area of focus

Existing KBE Methodologies	**Focus for Knowledge Modelling**
MOKA	Focus on both product and process modelling. ICARE forms for knowledge capture and MML for formalised knowledge
KNOMAD	Activity diagrams for processes and representation of multidisciplinary knowledge focusing both on product and process modelling
KCM	Product modelling in the form of parts, assemblies and features
KOMPRESSA	Product modelling in the form of diagrammatic ways of capturing knowledge such as component diagrams
RAD	Product modelling and direct implementation of knowledge on to the application
CommonKADS	Focus on both product and process modelling though UML notations and diagrams

MOKA, being one of the most comprehensive, lays emphasis on two stages of the KBE lifecycle as shown in Figure 2-3. First it captures knowledge in an informal manner in the form of ICARE (Illustration, Constraints, Activities, Rules and Entities) and then converts it to a formal manner. MOKA utilised Unified Modelling Language (UML) notation and

extended it to develop Moka Modelling Language (MML) as a means of producing a formal knowledge model (Chapman et al., 2007; Stokes, 2001).

CommonKADS also utilises object-oriented (O-O) modelling and uses UML notations such as class diagrams, use-case diagrams, activity diagrams and state diagrams in order to represent domain knowledge (Schreiber et al., 2000). Thus both MOKA and CommonKADS utilise UML based notations for knowledge representation. Even CommonKADS utilised similar stages of developing an informal based model initially and then developing the formal implementation.

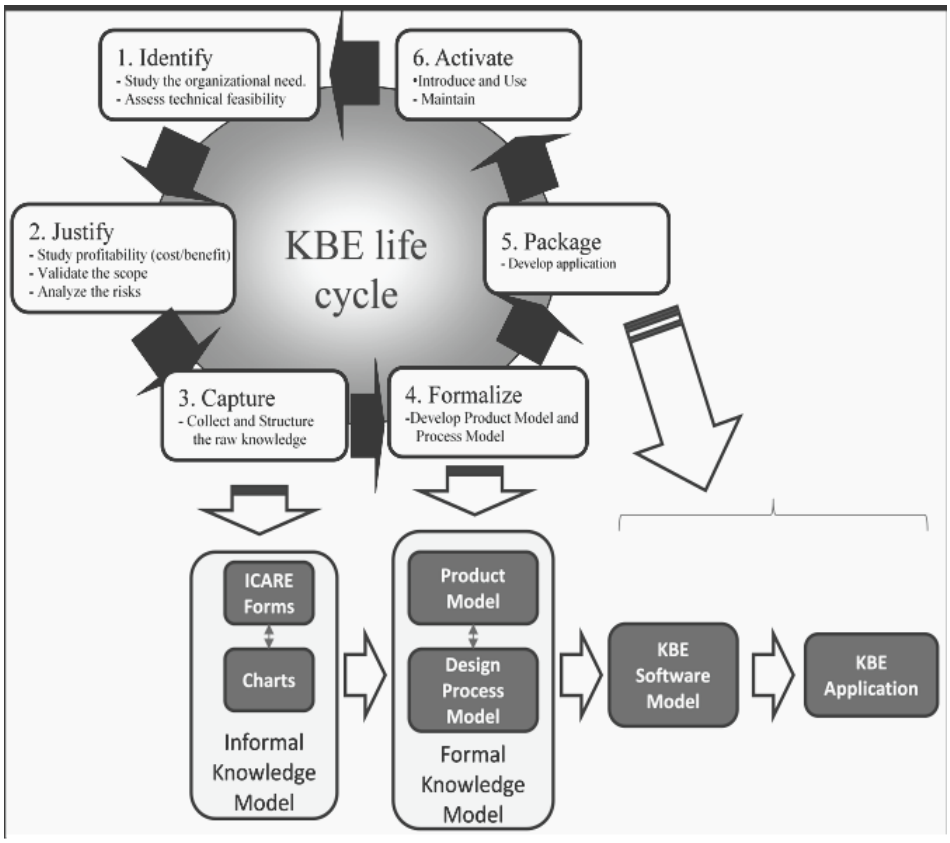


Figure 2-3: MOKA methodology in KBE lifecycle (Lohith et al., 2013)

Various tools such as PCPACK can be used which help in building inter-connected knowledge representation models. PCPACK supports knowledge capture, analysis and modelling of knowledge using both MOKA and CommonKADS methodology (La Rocca, 2011; Nan and Li, 2012; Schreiber et al., 2000). CommonKADS offers major advantage in

terms of adding structure to knowledge capture and representation. However, it lacks the accuracy and specialisation pertaining to knowledge capture and representation for engineering design (Sanya and Shehab, 2014).

In spite of strengths in managing engineering knowledge throughout the product lifecycle MOKA was revealed to have a few shortcomings e.g. MML did not comply with Object Management Group (OMG) requirements (Abdullah et al., 2005), the formal knowledge model could not be mapped to a KBES (KBE system) application to assist in process automation (Chapman et al., 2007; Prasad, 2006).

This piece of research initially intends to bridge this gap in correct syntactical and semantic mapping of an informal process model capturing all knowledge types and relationships of an engineering design process to a platform independent and neutral (interoperable usage through open standards) formal representation framework. It is very important to maintain traceability between the informal process model with captures the engineering design knowledge and the formal representation of the informal model (Verhagen et al., 2012). The formal representation should be computer readable and understandable (Klein et al., 2014) and fulfill the requirements of design engineering automation (DEA) as part of this research. The neutral formal representation framework of the process model will enable DEA similar to a KBES implementation with the help of suitable inference and querying mechanism as execution of its code.

2.6 Engineering Design Process Decomposition: Classification of Knowledge

According to ISO 10303-49, process can be defined as ‘a particular procedure for doing something involving one or more steps or operations. The process may produce a product, a property of a product, or an aspect of a product’ (Michel, 2005). Engineering design process

as a part of PD can be stated as a ‘process of converting design requirements into verified solutions’ (Isaksson, 2003). In the context of this research the engineering design process for product realisation should cover all stages of its lifecycle.All concepts required for modelling engineering design process will be discussed in this section. The type of specific concepts of the design process decomposition such as activity, inputs, outputs, resources, engineering rule, rationale, product function and behaviour will govern the selection of suitable formal representation techniques for the developed process model.

2.6.1 Engineering Design Activity

An engineering design process consists of various activities for creation and evaluation of products by changing their state (Isaksson, 2003). Design process activities consume some inputs and produce outputs with the help of resources and methods in order to convert functional requirements to verified solutions(Ding et al., 2009). All design process activities are highly interdependent and require knowledge such as inputs, outputs, resources and methods in the form of rules from other dependent design activities in order to be completed efficiently (Zhang et al., 2013). Each activity can be associated with an ID for system interpretation. Inputs can be defined as any entity that are consumed or modified during an activity and converted to outputs. Similarly, outputs can be defined as entities produced by an activity (Ding et al., 2009).Resources can be defined as the entities that provide support for the completion of an activity (Ding et al., 2009; Zhang et al., 2013). The methods govern the conversion of inputs to outputs and can be represented with the help of engineering rules based on logic and mathematics thus governing the conversion of inputs to outputs during an activity.

2.6.2 Engineering Rules

Engineering rules containing both design and manufacturing rules are often described by containing two important parts: product and process knowledge (Stokes, 2001). Product rules

contain clauses or criteria for relationship between different components of a product. Process rules contain criteria for different task sequence and selection based on requirements or constraints.

According to La Rocca, 5 different types of product rules can be described –

- Logic rules: rules based on logical statements and also containing conditional ‘If-Then’ and ‘If-Then-Else’ expression
- Math rules: contain mathematical formulae and comparison symbols
- Geometry handling rules: parametric and geometry manipulation rules governing the dimensions as size of the product
- Configuration selection rules: combination of logic and math rules governing the topology of a product. This includes the positioning of the product as position co-ordinates and orientation vector in the virtual space
- Communication rules: rules governing communication of system code with external formats (La Rocca, 2011)

Similarly, La Rocca describes process rules in the following ways –

- Process sequence: rules governing process sequence steps and input-output relationships
- Optimization: rules defining optimisation of process through functionality and constraints. This includes interdependencies between tasks (La Rocca, 2011)

As engineering rules are often based on logic, the type of logic will govern the suitable representation technique. According to logic, engineering rules can further be classified as one of the following types –

- Transformation – it includes simple statements that links to other statements and may thus be a statement declaration

- Derivation – it includes infer on facts within a statement and may thus be an implication declaration
- Reaction – it includes both trigger and production rule in the form of antecedent and consequent. Trigger rules have events in their antecedents and production rules have facts in their antecedents. 'If' part is called an antecedent and 'Then' part called a consequent and they are linked by logical operators such as 'AND' and 'OR'. Production rules can include nested facts in both antecedent and consequent. In order for the consequent to be true the antecedent need to be true. This is the reason for the antecedent and consequent facts based statement to be named a production rule. An example is –

(Reijnders, 2012)

For this thesis with focus on DEA, the engineering rules will contain all engineering design rules based on logic and math along with heuristic rules, production rules and process rules. These may be geometry handling rules as well as configuration rules and process sequencing and optimization rules (Chapman and Pinfold, 1999). They can be broadly classified as –

- Logic based Rules - rules based on engineering logic. These rules can include production rules, geometry rules, configuration rules and process rules. The process rules contain both process sequencing rules as well as optimization rules.
- Math based Rules - rules containing mathematical symbols and formulae. These rules can also include of production rules, geometry rules, configuration rules as well as process rules containing both sequencing and optimization rules.
- Production Rules - all statements in the form of 'If', 'Then' and 'Else' containing an antecedent and consequent linked by an operator such as 'AND' and 'OR'. These can

be either logic based rules and math based rules. Some rules can also overlap as demonstrating features of geometry rules, configuration rules and even production rules. In fact, all production rules are either geometry and configuration rules but they are expressed in 'If', 'Then', 'Else' representation.

- Heuristic Rules - rules not based on logic. Sometimes, engineering rules are rules of thumb and not based on logic statements. However, they may be geometry rules, configuration and even process rules based on rule of thumb. Heuristic rules are thus disjoint with logic rules, which means a rule can either be a heuristic rule or a logic rule but can't be both.

2.6.3 Function and Behaviour: Engineering Design Process

In order to create an efficient DEA system, it should be able to capture and represent the design intent in the form of process structure, function and behaviour and in context to the product (Brunetti and Golob, 2000). In engineering design process, a model or a framework should include function, behaviour, structure (F-B-S) and all design activities for a complete process description (Gero and Kannengiesser, 2007a). Alternatively, in order to describe an engineering design process for realization of a physical product, its function, behavior and structure (F-B-S) need to be defined (Alvarez Cabrera et al., 2009; Tomiyama et al., 2013).

'Function' is defined by an effect of a product or a component (Szykman et al., 2000a) or the purpose of the product or a component (Foufou et al., 2005; Patil et al., 2005). Thus 'Function' can also be described as what the object is for (Gero and Kannengiesser, 2004). 'Behaviour' can be described as a method of how a product or a component implements its function (Foufou et al., 2005; Patil et al., 2005). It can also be described as what the object does as deduced from its structure in the form of attributes (Gero and Kannengiesser, 2007b). F-B-S as function-behaviour-structure are artifacts that offer extremely high value during the conceptual and preliminary design phases (Erden et al., 2008). Regarding function in context

to engineering design process, it can be defined as a requirement that a design process is going to perform with the change in state of the product. Fulfilling functional requirement as product's function is one of the key aspects of a product design process (Bluntzer et al., 2009). Similarly, process behaviour can be stated as a method or utilisation of how the design process is going to achieve its function (Reddy et al., 2015).

If we consider either product or process as an artifact and then define function and behaviour, we can state function as what the artifact is supposed to do or the satisfaction of artifact's requirements. The behaviour can be stated as a method of how the artifact performs its function (Fenves et al., 2008). The process function can be stated equally as functional requirement of the design process. The function or functional requirement of a process governs the flow of energy, material, inputs and outputs of a process (Wang et al., 2002).Both function (as functional requirements) and behaviour along with product parameters and manufacturing knowledge have also been modelled as artefacts in context to DEA systems for all stages of design lifecycle from conceptual, embodiment to detailed design (Bhaskara, 2010; Brunetti and Golob, 2000; Chulvi et al., 2007; Roy et al., 2001).

2.6.4 Product Knowledge for Engineering Design Process

'Feature' can be described as associated knowledge of a component which aids in identifying its function (Patil et al., 2005). Feature can also be defined as 'an information unit representing a region of interest within a product (Brunetti and Golob, 2000). 'Form' can be defined as a physical layout of a component (Szykman et al., 2000a). 'Fit' describes the relationship of a component with other components and assemblies (Pinfold et al., 2008). Form, fit and features constitute the structure of a product. 'Form', 'Fit' and 'Features' entail rules and constraints governing product geometry, structure and material. A key characterization of product's state can be stated as the change in attributes of a physical product (Alvarez Cabrera et al., 2009). Correlating F-B-S we can state the behaviour of the

object is dependent upon its attributes and helps in achieving the function of the object. Behaviour of the product and its function alter its attributes indicated by change of state.

'Rationale' or 'Design Rationale' can be described as reasons behind design decisions (Medeiros et al., 2005). 'Rationale' can also be stated as the reason or explanation behind the design and specification of an artifact (Poorkiany et al., 2016). It includes the background knowledge which helps in reasoning and decision making for a particular design choice (Regli et al., 2000). For a process-based system, design rationale is descriptive capturing issues and available options illustrating design progress aiding in design process decision-making. In this research, Design Rationale as a concept or knowledge type is captured in a process-oriented approach.

2.7 Knowledge Modeling for Engineering Design Process

Knowledge modelling as an integral part of knowledge management is a critical activity in development of a knowledge based system (KBS) or a framework which helps in fulfilling DEA through KBE(Isaksson, 2003; Milton, 2007; Schreiber et al., 2000). Knowledge modelling process should ensure that the complete engineering knowledge of a product design process is captured, represented and processed efficiently. As discussed earlier, knowledge acquisition will be performed with mechanical design process as the main focus along with inclusion of both geometric and non-geometric knowledge of the product including process function, behaviour and structure (F-B-S) (Tomiyama et al., 2002).

2.7.1 Systems Engineering (SE)

Systems engineering can be defined as a multidisciplinary approach towards system specification, design, validation and verification(Krasner, 2015). The function of Systems Engineering (SE) is to 'guide the engineering of complex systems' (Kossiakoff et al., 2011). Thus SE deals with interrelated components, subsystem and parts, which form a complex

system and interact with each other and external elements in order to fulfil the system objective.A number of lifecycle models were initially developed for systems engineering purposes in the form of design, development and testing of the system such as Waterfall, Spiral and Vee models. Waterfall and the spiral model have been extensively used with modifications in various software development projects whereas “Vee” models have been used with variations in the systems engineering and development. Most of the existing SE standards have evolved from Department of Defense (DoD-MIL-STD 499) (Estefan, 2007).

2.7.2 Model Based Systems Engineering (MBSE)

Model based systems engineering (MBSE) is a model centric approach which helps understand the complex system behaviour, relation of requirements to functions and provides a complete view of the system model with the help of formalised and semantically rich visual modelling languages and tools(Krasner, 2015). According to International Council on Systems Engineering (INCOSE), MBSE can be defined as ‘the formalized application of modelling to support system requirements, design, analysis, verification and validation activities, beginning in the conceptual design phase and continuing throughout development and later lifecycle phases(INCOSE, 2007). Some of the important MBSE approaches are Object Management Group (OMG) visual modelling languages and standards in the form of UML and Systems Modelling Language (SysML). SysML was developed with collaboration between OMG and INCOSE and derived a lot of features from UML version 2.0. INCOSE object-oriented systems engineering method (OOSEM) uses a top down model based approach based on OMG SysML standards(Estefan, 2007). Model Drive Architecture (MDA) was an approach initiated by OMG in order to drive interoperable and re-usable architectural frameworks for systems. Dov Dori’s Object-Process Methodology (OPM) is another crucial formal paradigm to model based systems development and support(Dori, 2002).

2.7.3 Utilisation of SE and MBSE for Engineering Design: Product Development

Vee Model was utilised for knowledge capture of design process for complex product development by (Woestenenk et al., 2011). It is illustrated with figure 2-4.

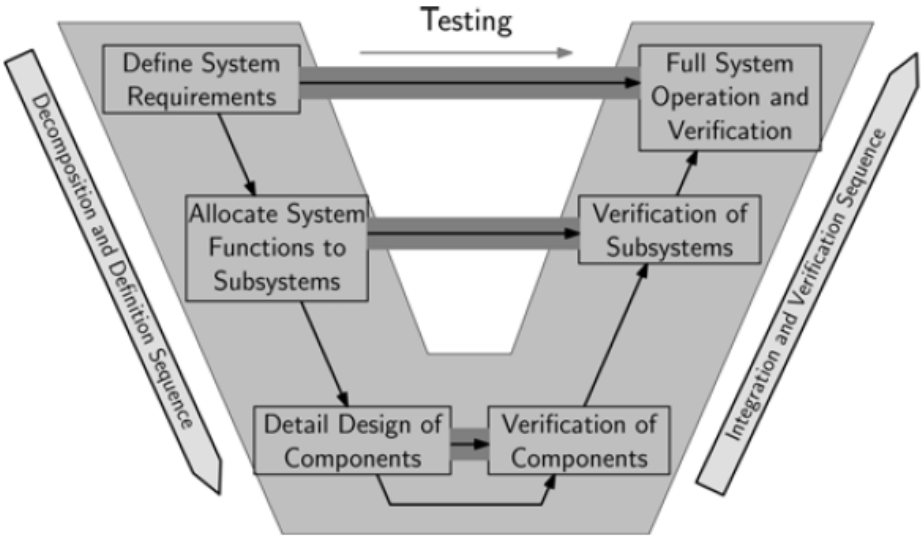


Figure 2-4: Vee Development for Engineering Design Process (Woestenenk et al., 2011)

As it can be observed, various steps include formulation of system function and requirements, detailed design and then verification of both systems and detailed components. The validation steps as testing are in synchronisation with the initial step of functional requirements analysis and detailed design. Some of the crucial points while following the Vee development process for engineering design (Woestenenk et al., 2011) are –

- Appropriate methods and language for capture of the complete engineering design knowledge in terms of concepts, decomposition and relations
- Capture and representation of functional requirements and structural decomposition for high level models along with inclusion of design activities, components and product parameters for detailed models
- A mechanism or a method to define and populate the knowledge models indicating the flow of information from functional requirements through to design activities and

product parameters which can be applied for generic use-cases and can be tracked in context of wider engineering design domain

- An equivalent machine interpretable formal representation of high level and detailed models for providing automation in engineering design along with a tool that can support the updating and modifications in the developed models

Thus the knowledge capture and representation stage for development of process model for DEA with KBE approach will adopt principles of "Vee" development model stages as an integral part of MBSE.

2.8 Existing Models and Frameworks for Engineering Design and Manufacturing Processes enabling DEA – KBE perspective

Many frameworks and applications exist for automation purposes in PD cycle. Most of them focus on product modelling and generation through models and framework along with various specific aspects of engineering design, analysis and manufacturing processes. Interestingly, none of the methodology or framework provides capturing of a generic and re-usable process and product domain knowledge, which can be utilised for developing a KBE application (Verhagen et al., 2012). Some of the crucial frameworks and models that have been developed for product development and addressed for knowledge based design and DEA purposes are discussed here.

Table 2-2: Existing Models and Frameworks for Design Engineering Automation (DEA)

Model / Framework - DEA	Description	References
Design and Engineering Engine (DEE)	In addition to KBE methodologies in the form of MOKA as discussed earlier, DEE is another model, which involves multidisciplinary design optimization approach (MDO). It includes of three modules – design process optimisation module, multi model generator (MMG) and detailed analysis module. Thus, DEE provides improved facilities as compared to MOKA by including detailed analysis and MDO and laid the foundation for KNOMAD methodology. However, it offers some limitations by not	(Curran et al., 2010; La Rocca et al., 2002; La Rocca and Van Tooren, 2007; Reddy et al., 2015)

	providing a method for knowledge capture and formalisation of captured knowledge along with its delivery in the mainstream processes.	
Linked knowledge in manufacturing, engineering and design for next generation production (LinkedDesign)	Linkeddesign project focussed on both KBE and GA based DEA. They explored various methods of knowledge acquisition and codification as formal representation of engineering knowledge with MOKA methodology as the basis and UML based product representation. One of the key focuses was identification of neutral formal representation standards with preserved semantics, which can represent the engineering knowledge as domain knowledge for DEA that can be re-used by both KBE applications such as AML as well as CAD based GA applications such as Siemens NX KF and CATIA Knowledgeware. For knowledge codification as neutral formal representation, various standards were identified such as STEPstandard as an ISO 10303 with focus on Application Protocol (AP) 242, XML representation of AP242 and ontology / rule languages such as Web Ontology Language (OWL)/Semantic Web Rule Language (SWRL) and Rule Interchange Format (RIF). A major contribution of the Linkeddesign project was the recommendation of RIF for neutral standard representation and exchange of engineering rules. However, it was not demonstrated that an engineering design process could be represented in RIF and whether the process model is relevant for DEA along with a requirement to further validate RIF. OWL/SWRL was identified as a strong possibility of formal representation or codification of engineering knowledge with preserved semantics.	(Colombo et al., 2014; Lützenberger et al., 2012; Mocan et al., 2015; Perales and de la Maza, 2015)
Reijnders	Post MOKA, another contribution was made by Rejinders in developing platform independent and formal representation of engineering design knowledge for aerospace industry for DEA with a KBE approach using a combination of OWL, RIF and MathML using a commercial implementation tool Allegro Graph based on Allegro Common Lisp platform. Although product and process knowledge was represented, the main focus of the captured and represented knowledge was based on engineering rules for product design. MOKA ICARE forms were used as informal representation with the corresponding platform independent formal representation of rules in RIF-Production Rule Dialect (PRD) and Content MathML. Although it offered successful formalisation of design knowledge, the predicates of the rules such as the antecedent and the consequent couldn't be queried due to integration between RIF-PRD and OWL leading to loss of contextual relevance of rules with co-related knowledge. It was also recognised that single rules related to an object or a process were easily modelled, but multiple rules were difficult to implement.	(Reijnders, 2012; Van Tooren et al., 2003)
Sanya and Shehab	Following the MOKA methodology, Sanya and Shehab performed work for the aerospace industry for development of platform independent knowledge models using OWL/SWRL to formalise the design knowledge with Protégé as a tool. The building of platform independent knowledge models for DEA with KBE approach also helps in building of dynamic, portable and adaptable systems and supports re-usability of knowledge.	(Sanya et al., 2011; Sanya and Shehab, 2015, 2014)

	Although the knowledge model was based on functional requirements as the basis, the focus was on design parameters, constraints and rules for specific aerospace components such as compressors and turbines based on feature and shapes such as sleeve, panel and flanges. It also recognised that using semantic web based languages such as OWL ontology for DEA with a KBE approach, there was a lack of standard method based on a set of activities which would deploy the OWL model for use in KBE applications with a lack of widely adopted ontology development for engineering design and DEA. It was also recognised that there was lack of research between ontology development and engineering design.	
J Kulon: Hot Forging Process	A KBE model for automation of hot forging process with focus on the product model was developed by. In order to include relevant product knowledge, the model included design rules, production rules, and material information. The automation application method consisted of an integrated relational database over the web browser with requirements, design rules and product modelling key concepts such as components, material and manufacturing rules and complex interdependencies within the domain concepts. The visualisation of the product geometry and structure was done over the web with the help of Virtual Reality Modeling Language (VRML). However, the design and production rules pertain to product functionality, structure and behaviour instead of process-centred approach.	(J Kulon et al., 2006; J. Kulon et al., 2006; Qin et al., 2003)
Adaptable Methodology for Automation Application Development (AMAAD)	A KBE system application for aerospace design and analysis process was developed in a commercial environment based on MOKA and CommonKADS methodology. The AMAAD methodology focused on a re-usable, generic and high-level model. It laid emphasis on object-oriented (O-O) UML based notation and Integrated Definition for Functional Modelling (IDEF0) notation as part of agile development with MBSE approach. It involved knowledge acquisition and knowledge modelling after requirements specification before proceeding to system development and validation. The output of the developed system could be integrated with CAD architecture through platform independent and neutral format. However, a major limitation was it didn't provide a structured method to conduct the individual and detailed activities along with association of these activities with complex system working and its attributes required to achieve DEA with a KBE perspective	(Van Der Velden et al., 2012; Van der Velden, 2008)

2.9 Synthesis and Findings of DEA Review

MOKA methodology focused on development of neutral formal representation of the domain knowledge in the form of ICARE forms for the development of an independent model of the engineering design process knowledge at the system level or machine interpretable level for DEA. It recommends XML as the basis for development of neutral model for system

interpretation of the informal model but doesn't provide a detailed method for developing the neutral model (Stokes, 2001). The neutral formal model will be the basis of the software code as the source code of a KBE application or tool. This is one of the research gaps that this research will satisfy by providing a detailed method for development of a neutral formal process model for DEA with MOKA methodology as the basis.

As per Wagner (Wagner et al., 2003, 2001) the problem in knowledge acquisition and modelling in context to an expert system for automation, is the method of acquiring both structured and non structured domain knowledge for decomposition into fragments and representing it in the appropriate computer format for example an expert system shell. As KBES are expert systems with geometry kernel for the engineering domain, knowledge modelling is extremely critical for DEA in context to KBES.

KBES allow integration of rule-based design, geometry manipulation and computational capability in the form of forward and backward chaining as inference or reasoning mechanism for knowledge processing, which differentiates KBES from traditional CAD and expert systems and allows KBES to combine their individual capabilities for complex problem solving (La Rocca, 2011; Rocca, 2012). As stated earlier, the main contribution of MOKA methodology was the capture stage through ICARE forms and formalise stage through MML as visual representation. It tried to address automatic generation of KBES source code from MML as proof of concept for preliminary analysis even though MML didn't comply with OMG requirements (Abdullah et al., 2005). As PCPACK can be used for MOKA methodology requirements, PCPACK was used as a knowledge modelling and representation tool for MML diagrams and produced an internal XML representation as neutral formal representation for conversion to the source code in a KBES. However issues were encountered for mapping of the neutral formal knowledge model to a KBES such as lack of semantic clarity of XML, which causes multiple translators to interpret the XML

based neutral formal knowledge model (La Rocca, 2011). Also, lack of focus on other knowledge representation (KR) for development of formal models was a major shortcoming of MOKA which can lead to knowledge accessibility and re-use issues (Curran et al., 2010; Verhagen et al., 2012). Thus, the formal knowledge model from MOKA as MML was unable to assist in DEA using KBE methodology and application (Chapman et al., 2007).

KNOMAD as a methodology tried to integrate multidisciplinary knowledge for design optimisation and DEA (Curran et al., 2010). The various steps include – (K)nowledge Capture, (N)ormalisation, (O)rganisation, (M)odelling, (A)nalysis, (D)elivery. For the (M)odelling stage, it adopted the MMG approach by DEE and built upon it to provide a structured methodology for DEA through KBE. It provided tools such as Protégé to support ontology construction using Web Ontology Language (OWL) for both products and processes allowing for knowledge traceability and application deployment. However, various areas of improvement were identified such as a clear, structured and concise knowledge modelling and analysis approach or method along with the validation of the method with original case studies (Curran et al., 2010).

Thus, it is identified from the literature that most of the KBE methodologies including KNOMAD and MOKA being the most comprehensive, there is a lack of process oriented approach to capture engineering design with manufacturing knowledge for representation in a platform independent and neutral formal manner with preserved semantics (Chapman et al., 2007; Rocca, 2012; Verhagen et al., 2012). Most applications developed, as KBES are case based and ad-hoc with no adherence to existing structured methodologies (Rocca, 2012; Phillip Sainter et al., 2000). Also most of the applications developed are black box, with lack of knowledge transparency and traceability issues for DEA (Ammar-Khodja et al., 2008; J Kulon et al., 2006; J. Kulon et al., 2006). This includes lack of semantic clarity in the design intent for example engineering rules and their relevance to the product and process

knowledge. The knowledge is decoupled from original context, documentation is not explicitly stated with their clear semantics such as co-relation of engineering rules in the form of formulas and equations. This leads to lack of knowledge sharing, traceability and re-use as well which is enhanced by the difficulty of knowledge sharing across different proprietary platform specific KBES or KBE applications (Verhagen and Curran, 2010; Verhagen et al., 2012). Formalisation is the key to enhance re-usability and sharing and address the needs of DEA with application development. However, the key problem is an unstructured knowledge modelling process, which leads to unstructured knowledge codification as formal representation (Klein et al., 2014).

There is a lack of capture and representation of non geometric knowledge in most KBE applications for re-use such as project constraint reasoning, problem solving methods and solution strategies as part of design intent (Baxter et al., 2007). As stated by Pablo Bermell-Garcia, 'using current data exchange standards, it is only possible to transfer an instance of the design and not the knowledge embodied to generate it' (Bermell-Garcia, 2007). Thus new knowledge bases should ensure knowledge sharing across different platforms with neutral usage through open standards. They should be flexible and user friendly as well for effective sharing, re-use and maintenance with semantic clarity of design intent (Verhagen et al., 2012).

Similarly, the source code in a KBES for a particular function for product parameters doesn't reflect the stage of the design process such as conceptual design or detailed design phase. The implementation of the function varies from stages of the design lifecycle such as conceptual and detailed design. Thus, a suitable method for knowledge modelling for DEA using KBE approach should incorporate the relevant aspects of engineering design and development process such as mechanical design with DFM. Also, the neutral formal standard should ideally provide visualisation support for codified domain knowledge for direct consumption

by design engineers (Klein et al., 2015). According to Jubierre and Borrman (Jubierre and Borrmann, 2015), it is crucial to achieve high abstraction of engineering knowledge consisting of technical guidelines and standards for DEA using KBE approach. The knowledge base should have high level of abstraction with a logical modelling approach for development of a neutral formal representation layer for automation with generative modelling capabilities.

In order to address the current limitations such as those by Linkeddesign, Sanya/Shehab, Reijnders, DEE and others, this research aims to bridge these identified gaps by providing a structured method for process based knowledge modelling in concurrency with MBSE approach, its formal representation and its verification with test use-cases as corresponding analysis. This method of schema mapping will also provide transparency and traceability with semantic clarity in the developed process knowledge model with both geometric and non-geometric knowledge for re-use as part of product development. This research will also provide mapping of engineering design aspects with focus on mechanical design and DFM for DEA and re-usable ontology development method with multiple rules and generative modelling capability.

2.10 Summary

This chapter discusses various aspects of DEA with virtual engineering. It also discusses all knowledge entities required to model as part of systems engineering and MBSE with an MDA approach for DEA such as process description, engineering rules, function and behaviour. Through a detailed analysis of existing DEA techniques various gaps were recognised such as a detailed and structured method for development of neutral formal representation of an engineering process model with focus on mechanical design and DFM with both geometric and non-geometric knowledge for traceability, transparency and semantic clarity with contextual relevance as none of the existing KBE methodologies were

successful in achieving DEA from an independent neutral formal representation of a process model for engineering design (Elgh and Johansson, 2014). This is further enhanced by lack of open standard usage, documentation for knowledge modelling and knowledge re-use. ***This research will bridge these gaps by providing a structured and detailed method in the form of a re-usable process model for capturing the activities of the mechanical design process with DFM/DFA and their corresponding neutral formal representation with preserved semantics for DEA with generative modelling***. KBE based approach for DEA will be primarily adopted along with GA in order to develop a knowledgebase with both geometric and non-geometric knowledge for automation with primary focus on the mechanical design process with manufacturing knowledge. The developed process model will be generic, expandable both as informal and formal representation to enable re-usability. This will include design process, rules based on logic, process function and behaviour with product knowledge as F-B-S. In order to develop this model an understanding of existing informal and formal representation standards for knowledge modelling of mechanical design process with activity decomposition and inter-dependencies between knowledge is required which are discussed in the next chapter.

3 Informal and Formal Modelling of Engineering Processes

3.1 Introduction

Chapter 2 provided an overview of design engineering automation (DEA) methods and techniques for mechanical design process as part of product development (PD). Various knowledge types as design decomposition features were described as integral constituents. This chapter will initially discuss existing informal and semiformal modelling standards for knowledge modelling of mechanical design processes with DFM for DEA along with their comparative analysis. The later part will elaborate on the formal representation standards, which will ensure mapping of the concepts of the informal model to the neutral formal representation with preserved semantics.

3.2 Process Modelling for Design Engineering Automation

'Process modelling is an activity set to be followed to create one or more models of a process for a certain purpose, usually the representation, explanation, design, specification, analysis, or control of a given process' (Amigo et al., 2013, Pg 169).According to the National Institute of Standards and Technology (NIST), a 'process model for product realisation is defined as a computer–interpretable representation of human and machine activities and their interactions required for realisation of a product. This may include early concept and configuration design activities, detailed design, prototyping, testing, tooling, fabrication, assembly and other activities within the scope of the realisation process' (Lyons et al., 1995).

There are many methods of capturing and representing knowledge for a DEA or a KBE system. The approach that will be followed as part of this research aligns its concepts to object process methodology(OPM) whose feature is that it breaks down the knowledge into three types of entities: objects, processes, and states with objects and processes being higher level building blocks (Dori, 2002). OPM is also recognized as an International Standards

Organization (ISO) standard in the form of ISO/PRF PAS19450 (Dori, 2002).The OPM methodology keeps systems as the viewpoint and enables merger of object-oriented and process-oriented modelling. The states are indicated by links, which exist as both structural and procedural links representing the static and dynamic behavior of objects in a system. OPM allows for features such as inheritance, and aggregation of objects and their properties. It offers object-process language (OPL) and object-process diagram (OPD) as a means of formal representation of the informal representation (Dori et al., 2003; 2010). The OPL enables java code generation and automatic generation of UML diagrams and natural text output. Pertaining to this research, the formal representation of the entire process model should enable code generation for fulfilling the purpose of process automation.

There are many governing factors for selecting a process modelling technique. Some of the existing purposes are task scheduling, resource allocation, cost-quality-time trade-offs and process improvement in terms of design-to-market lead time (Smith and Morrow, 1999). In order for a process-based model to be interpreted by KBE systems to achieve automation of processes, the process modelling technique should broadly satisfy the following functions–

- **Inter - dependencies between tasks** to enable flow of information such as inputs, outputs, enablers, mechanisms into multiple tasks which will enable dependency backtracking in the formal representation in the system
- **Design process decomposition** to the highest level of abstraction of artefacts, which includes all features such as function, attributes of a process and product with states and behaviour along with resources and requirements. This also includes control mechanisms and enablers for a process for failure modes through existing rules, constraints and logic for successful process adherence and completion. These may be in the form of geometrical tolerances, manufacturing constraints or material selection information for a design process

- **Object-process relationship** by breakdown of the knowledge content primarily in the form of objects and simultaneous representation of governing processes altering the state and behaviour of the object
- **Computational capability** indicating that all aspects of the process model can be mapped to a software system or formally stored in a system with well-defined syntax and axioms which can then be queried and inferred (reasoning) to achieve DEA in terms of process automation

The requirements as functions have been deduced with the help of the following sources (Calkins et al., 2000; Chapman and Pinfold, 1999, 2001; Chapman et al., 2007; COLOMBO et al., 2005; Cooper and LaRocca, 2007; Lohith et al., 2013; Prasad, 2006; Skarka, 2007)

The process modelling techniques discussed will be analysed for various functions as described below -

- Task scheduling and sequential planning
- Cost/time/quality trade-off
- **Inter - dependencies between tasks**
- **Design process decomposition**
- **Object-process relationship**
- **Computational capability**

Thus techniques, which satisfy the stated criteria out of all described functions, will be carried forward for formal representation.

3.3 Informal Modelling Techniques for Engineering Processes

Standards such as Design Structure Matrix (DSM), IDEFX suite, Petrinet, Signposting, Role Activity Diagram (RAD), MBSE based UML/SysML and Business Process Modelling

Notation (BPMN) will be discussed and analysed for capturing engineering design process knowledge to enable design automation in this section.

A process modelling technique based on a matrix structure for sequencing and scheduling is a design structure matrix (DSM) (Eppinger et al., 1994). DSM lays emphasis on activity dependencies and can focus on complicated processes with more than 100 tasks (Smith and Morrow, 1999). It helps in assessment of risks throughout the design process along with failure modes (Amigo et al., 2013). DSM as a technique also helps in implementing concurrent engineering, which is a major advantage when cost is considered an important parameter. It also helps in generating key performance indicators (KPI) to show status of an activity (Amigo et al., 2013). However, one of the limitations of DSM is the lack of ability to manage tasks within an iterative group. Work Transformation Matrix (WTM) is a process modelling method which helps in decomposition of a larger task into small processes (Smith and Eppinger, 1997). It is derived from DSM with a modification that the non-diagonal elements in the matrix are represented by re-work quantity. However, a major shortcoming of WTM modelling is the assumption of computation of re-work as a linear function of work from a previous iteration, which is not true in all cases. Both the techniques including DSM and WTM have strengths in modelling interdependencies of tasks along with process planning and improvement but fail to capture all of the necessary design decomposition features along with lack of focus on object-process relationship.

Modelling techniques such as Petrinet and Event Process Chain Diagram (EPC) fulfill the purpose of measurement of productivity of a process and work flow modelling (Amigo et al., 2013). Petrinet is based on nodes and arcs to represent information (Murata, 1989) and most importantly consists of two kinds of nodes in the form of places and transitions. One of the limitations of Petrinet is its inability to consider time as a process variable (Browning et al., 2006). Petrinet uses tokens as activity inputs to determine the activity's state in order to

execute the activity (Knutilla et al., 1998). Petrinet fails to capture contextual information although it can be used for modelling of interdependencies of tasks (Stacey et al., 2000). To capture contextual information, modifications can be made to Petrinet. For example, NIST researchers used Modified Petrinet (MPN) in an object-oriented methodology to include additional information such as mechanisms and rules for governing failure modes along with resources in the form of people, machines and tools in order to implement computer aided concurrent engineering (CACE) (Lyons et al., 1995). Thus MPN can be used to indicate inter-dependencies within a process along with design decomposition features. An Event Process Chain diagram(EPC) helps in generating tools for benchmarking along with documentation of design data (Amigo et al., 2013; Browning, 2009). Both EPC and Petrinet techniques can be used for simulation of design process, which indicates the behaviour of the process in different scenarios. EPC fails to capture design decomposition features but MPN allows the capturing of design decomposition features along with focus on object-process relationship.

A modelling method, initially for representing manufacturing systems, but which progressed to the design process is Integrated Definition for Functional Modelling (IDEF0) (Colquhoun et al., 1993; FIPS PUBS, 1993). It was derived from Structured Analysis and Design Technique (SADT). An IDEF0 model comprises of a set of activity boxes referred as ICOM (Input, Control, Output, and Mechanism). The top level box is the highest fidelity model and can be represented elaborately in more detail using lower fidelity models (Colquhoun et al., 1993; Gingele et al., 2002). The ICOM activity box for IDEF0 is illustrated with the help of Figure 3-1. Based on MOKA methodology, IDEFO was used with control and resources by developing *Onto-Process* for the production design domain in context to ICARE forms for automation with a KBE perspective (Martínez-Pellitero et al., 2011). PC-PACK was used as a knowledge acquisition tool for knowledge capture of inspection planning process. IDEF0 was

also used by (Gómez et al., 2013) along with UML notation as an information model for conceptual assembly design and its process automation with a KBE perspective.

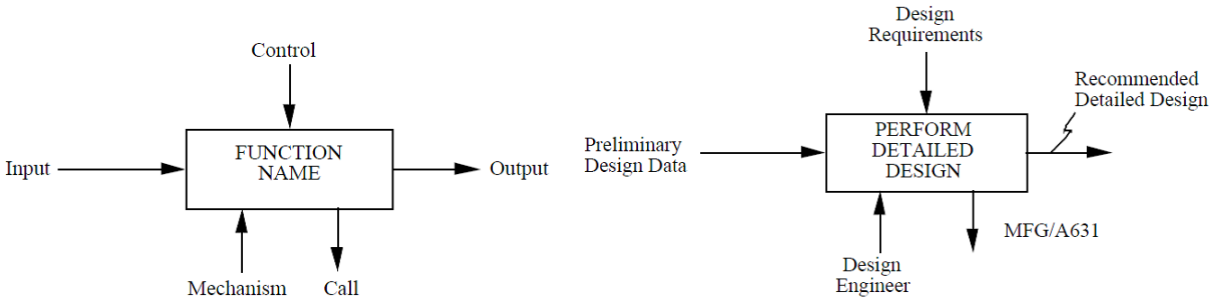


Figure 3-1: IDEF0 higher fidelity activity box with an example (PUBs, F.I.P.S, 1993)

Although IDEF0 was found to be a very detailed graphical representation of the processes (Al-Ahmari and Ridgway, 1999) with all the control parameters, it was considered to be time consuming. A major shortcoming of the IDEF0 approach was its lack of consideration of time as a variable. IDEF1 was introduced after IDEF0 and was based on information modelling instead of IDEF0 functional modelling. It shows the relation between constraints and is based on entity relationships (Lyons et al., 1995; Mayer, 1992). IDEF1 lays emphasis on representing information based on a class of entities with attributes to define their behaviour (Lingzhi et al., 1996). Thus it can be used to model real world objects as well as information required to manage an enterprise. IDEF2 was introduced to address a major shortcoming of earlier IDEFX versions for their lack of inclusion of time. It was supposed to be dynamic but was not successfully implemented in commercial systems (Lyons et al., 1995). IDEF3 shows the relation and logical flow of activities within a process(Mayer et al., 1995). It is referred to as a process description capture method with time-based behaviour of activities. Another advantage of IDEF3 was that it can show two views of the process, one termed Process Flow Network (PFN) which lays emphasis on activity and the other Object State Transition Network (OSTN) which allows an object – centered view (Knutilla et al.,

1998; Plaia and Carrie, 1995). The IDEF3 process description method lays emphasis on the flow of junctions, which embeds the time varying behaviour of activities.

IDEF4 is an object-oriented (O-O) design process description and broadly consists of two models –class and method sub-models with diagrams such as protocol, inheritance and taxonomy diagrams which can be interlinked and sufficiently capture all intricate parts of a process (Mayer et al., 1992). The complete IDEF suite, however, adopts slightly different methods to capture process information, as illustrated. IDEF0 focusses on function modelling, IDEF1 focusses on information modelling, IDEF2 on simulation modelling, IDEF3 on detailed flow of junctions in a process flow, IDEF4 on O-O design and IDEF5 on ontology-based description (Plaia and Carrie, 1995). IDEF4 will be discussed in detail in the next section on 'semi-formal modelling methods and languages' to verify whether it satisfies the requirements for design process automation. IDEF5 will be discussed under 'formal representation methods'.

A Role Activity Diagram (RAD) enables a graphic view of the process with interactions between various processes. It allows an object-oriented (O-O) view of the process with changes in behaviour of the object with activities (Aguilar-Saven and Ruth, 2004). However, one of the limitations of RAD is its inability to decompose the high level processes to lower levels of process with precise details. RAD can be used to model workflows for improvement. RAD can be visualised through MS Visio (Shukla et al., 2014) but it captures high level aspects with activities assigned to roles for a particular system but doesn't capture design decomposition features as stated in the requirements for design automation.

A Data Flow Diagram (DFD) shows the flow of process data and information graphically. It enables decomposition of the process to a lower level of detail (Aguilar-Saven and Ruth, 2004) in contrast to RAD. It allows functional modelling and thus has conceptual similarities to IDEF0. However, it fails to capture all design decomposition features.

Business Process Modelling Notation (BPMN) is an object-oriented (O-O) modelling method and is a recognised standard of the Object Modelling Group (OMG) (Sharma et al., 2014). It includes swim-lanes to show the roles of actors in a system. In this way, it has similarities with the RAD. BPMN can be used to describe activities with the flow of information similar to RAD, Unified Modelling Language (UML) activity diagram and EPC. BPMN can be enhanced to show activities, events, decision nodes, and activity along with actors and roles. BPMN defines 50 constructs and attributes, which can be grouped together in four categories – flow objects, connecting objects, swim lanes and artefacts (Muehlen and Recker, 2008). Flow objects are the most basic constructs and consist of events, activities and gateways. Connecting objects show interdependencies through arrows and links. Swim lanes can be used for categorization of activities. Artefacts can be used to add contextual information to the model. BPMN can be used to model both functional and non-functional requirements (Heidari et al., 2013), improve business processes in terms of lead time to market for products, and in the visualisation of processes. However, it fails to capture all of the design decomposition features to enable design process automation.

The Signposting model is a task-based modelling method. It is based on three core elements – tasks, states and 'signposting parameters', offering three views – task level, process level and the parameter level (Clarkson and Hamilton, 2000). Depending upon the confidence of the parameters, a relationship between tasks is constructed. Thus it enables modelling of the interrelationships between tasks and can also be modelled as a DSM approach. Signposting is very useful for modelling uncertainty in the design process which is a critical feature (O'Donovan et al., 2003). It also offers inclusion of additional text information in its core constructs which can include requirements (Stacey et al., 2000). It allows for the capture of design decomposition features through the addition of contextual information along with interdependencies. It is illustrated with the help of Figure 3-2. Power and rigid body are tasks

to be performed. L, M and H are low, medium and high confidence rating of the parameters such as blade-loads and engine power. After the total confidence of the task is performed based on these parameters, it is used to determine whether the task will be successfully completed. Thus task status is derived from confidence mapping of parameters.

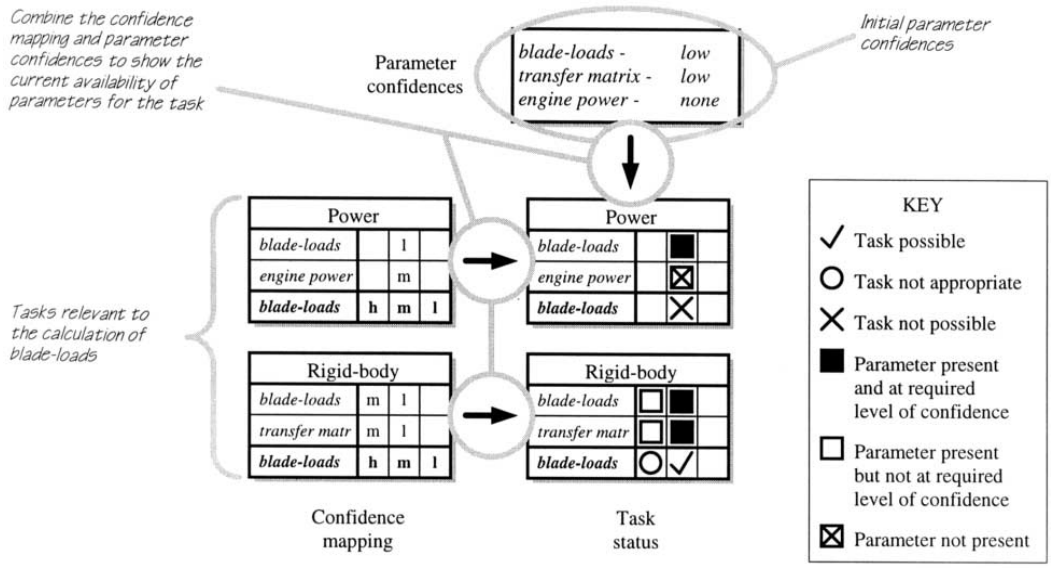


Figure 3-2:Using Signposting to derive task status from confidence mapping of parameters (Clarkson and Hamilton, 2000)

3.4 Semi-formal Modelling Methods and Languages for Engineering Processes (Light weight formalisms)

As per the context of a formal representation of an informal process model to enable DEA, there exists a boundary between informal and formal modelling. All the informal process modelling techniques can be used to capture process-based data in a human readable form or natural text output form. Similarly, languages like UML and SysML can be used both to capture data and represent it formally using tools. Alternatively, any informal method of capturing data can be converted into XML serialisation, which then becomes a formal representation. A formal representation is a low level machine-readable format, which may or may not be easy to understand by humans as against a natural text output, but offers ease of

processing by machines. XML is a data modelling language, which can be used for representing information as tags and exchanging between different applications (Chung and Lee, 2002). XML as a basic language consists of a prolog, elements and an optional epilog (Antoniou and Van Harmelen, 2004). The prolog consists of an XML declaration.

'UML is a language for specifying, visualizing, constructing and documenting the artifacts of software systems, as well as for business modelling and other non-software systems' (Aguilar-Saven and Ruth, 2004).UML version 1.4.2 is considered as an international standard as specified by the OMG in the form of ISO/IEC 19501 (ISO, 2005; Weilkiens, 2007). Various versions of UML exist, starting from OMG recognition of version 1.3 in 2000 to version 1.4 in 2001 to version 2.5 in June 2015 (OMG, 2016). UML version 2 is defined by ISO 19505 (ISO, 2012).

UML is an MBSE approach and utilises object-oriented techniques and nine types of diagrams to model and exhibit information in the form of: class, object, state-chart, activity, sequence, collaboration, use-case, component and deployment diagrams (Aguilar-Saven and Ruth, 2004). UML 2.0 illustrates both structural and behavioural aspects of a system. According to Tim Weilkiens, it illustrates structural aspects through class diagram, component diagram, object diagram, composite structure diagram, deployment diagram and package diagram and behavioural aspects through activity diagram, use case diagram, state machine diagram, sequence diagram, communication diagram, timing diagram and interaction overview diagram (Weilkiens, 2007). There are three main modelling viewpoints in UML – use-case, static and dynamic models (Kim et al., 2003). The use case models define the generic processes that the system should handle. They provide a graphical description and although offer a very brief description, they are similar in principle to IDEF as a means of communication through graphical display. The static view includes class diagrams, which enable a static view in terms of objects and relationships within objects of a

class. The dynamic modelling view enables communication between the system objects. For dynamic modelling UML utilises four types of diagram- state, sequence, collaboration and activity diagrams (Kim et al., 2003). UML can be used as an informal modelling technique and then maps to a formal representation through a final diagrammatic layer known as implementation diagrams.

Systems modelling language (SysML) was derived from UML as part of MBSE for the modelling of complex systems involving real life objects (Weilkiens, 2007). SysML inherits a lot of properties from UML with the addition of two types of diagram – requirement and parametric diagrams. It has minor variations on UML. Blocks in SysML replace UML classes. The class diagram in UML is replaced by a block definition diagram in SysML and the composite structure diagram in UML is replaced by an internal block diagram in SysML (Weilkiens, 2007). A very important point about SysML is that the models can be exchanged via a neutral format in the form of ISO AP233 (discussed later). Both UML and SysML with multiple viewpoints can exhibit and represent design decomposition features along with interdependencies of tasks.

IDEF4 as a derivation of IDEF features but with a focus on object-oriented technique and is similar to UML in terms of layering and process views. Both are object-oriented modelling techniques, which are necessary for capturing processes and representing in a neutral format for process automation. 'IDEF4 is an object-oriented design method for developing component - based client server systems. It has been designed to support smooth transition from the application domain and requirements analysis models to the design and to actual source code generation' (Mayer et al., 1992). IDEF4 provides three layers – system design, application design and low-level design. Thus it decomposes design into higher level of abstraction. Along with the three design models, IDEF4 includes a design rationale component. In IDEF4, symbols such as O, R, L, M, A, E are used to denote objects, relations,

links, methods, attributes and events respectively (Mayer et al., 1992). Thus its concepts become similar to UML by focusing on object-oriented modelling and by providing multiple layers of the design process. However, the design rationale component in IDEF4 is an additional feature and provides the designer with a wider view of the design data. This makes IDEF4 suitable for capturing all of the design decomposition features required for process automation. It also enables inter-dependencies between tasks along with illustrating changes in the state of an object with governing processes propagating throughout the model with object-oriented (O-O) modelling.

3.5 Comparative analysis of informal and semiformal modelling methods and languages for knowledge modelling of an engineering process

As stated earlier, the majority of process modelling techniques for knowledge acquisition or capturing can be visualized or edited with the help of existing tools. Some examples are – use of SIMAN / ARENA tool for simulation of IDEF0 (Al-Ahmari and Ridgway, 1999), ProCAP for IDEF3 (Grüninger, 2009), and CAM for construction and visualisation of Signposting (Wynn et al., 2010). Thus computational capability will be excluded from the criteria in the analysis table as any process-based method of capture can be converted into XML syntax and stored in a system with a formal representation. The other three criteria i.e. inter - dependencies between tasks, design process decomposition and object-process relationship will be the most important functions in evaluating whether a process model can broadly capture enough information which when mapped onto a formal representation can achieve process automation. The analysis is shown in Table 3-1

Table3-1: Analysis of informal and semiformal modelling methods and languages for capturing engineering process knowledge to enable design process automation

Modelling Methods & Languages	**Functions**					References
			Required for mapping to formal representation to enable design process automation			
	Task Scheduling / Sequential Planning	Cost / Time / Quality Trade-off	**Interdependencies between tasks**	**Design Process Decomposition**	**Object-Process Relationship**	
DSM	✓	✓	✓			(Amigo et al., 2013; Browning, 2009; Eppinger et al., 1994; Smith and Morrow, 1999; Wang et al., 2002)
WTM	✓	✓	✓			(Amigo et al., 2013; Smith and Eppinger, 1997; Smith and Morrow, 1999)
Petrinet	✓		✓			(Amigo et al., 2013; Browning et al., 2006; Grüninger and Menzel, 2003; Knutilla et al., 1998; Lyons et al., 1995; Wang et al., 2002)
MPN (e.g. Coloured Petrinet, Timed Petrinet)	✓		✓	✓	✓	(Aguilar-Saven and Ruth, 2004; Amigo et al., 2013; Browning et al., 2006; Knutilla et al., 1998; Lyons et al., 1995)
EPC	✓		✓			(Amigo et al., 2013; Browning, 2009)
IDEF0,1,2,3,4, 5	✓		✓	✓	✓	(Aguilar-Saven and Ruth, 2004; Al-Ahmari and Ridgway, 1999; Amigo et al., 2013; Browning, 2009; Colquhoun et al., 1993; FIPS PUBS, 1993; Gingele et al., 2002; Grüninger and Menzel, 2003; Klein et al., 2014; Knutilla et al., 1998; Lyons et al., 1995; Mayer et al., 1995, 1992; Plaia and Carrie, 1995; Wang et al., 2002)
RAD	✓		✓		✓	(Aguilar-Saven and Ruth, 2004; Badica and Badica, 2011; Badica et al., 2005, 2003; Holt et al., 1983; Shukla et al., 2014)
DFD	✓		✓			(Aguilar-Saven and Ruth, 2004; Al-Ahmari and Ridgway, 1999; Amigo et al., 2013; Colquhoun et al., 1993)

Signposting	✓	✓	✓	✓	✓	(Amigo et al., 2013; Baxter et al., 2007; Browning, 2002; Browning et al., 2006; Clarkson and Hamilton, 2000; O'Donovan et al., 2003; Stacey et al., 2000; Wynn et al., 2010)
UML, SysML	✓		✓	✓	✓	(Badica and Badica, 2011; Booch et al., 1999; Chen and Chen, 2005; Kim et al., 2003; Klein et al., 2014; Nan and Li, 2012; Plateaux et al., 2009; Pooley and King, 1999; Sharma et al., 2014; Vernadat, 2002; Weilkiens, 2007)
BPMN	✓		✓		✓	(Amigo et al., 2013; Badica and Badica, 2011; Heidari et al., 2013; Scheuerlein et al., 2012; Sharma et al., 2014)

3.6 Formal modelling and representation techniques for engineering processes and DEA

In order to perform DEA from the process model, the focus of representation should be on low level machine interpretation instead of natural language (Patil, 2005; Szykman et al., 2000b). This clarifies that the modelling techniques should enable computational reasoning as just opposed to modelling techniques for human aid (Hsu and Woon, 1998). There are many existing formal representations, which can be used for representing engineering process models.

3.6.1 Classification of Formal Representation Standards

Existing process descriptions and process ontologies not based on formal logic provide inadequate semantics for computational support in context to achieving granular DEA at an informal/semiformal layer (Gero and Kannengiesser, 2007a; Patil, 2005). The formal representation standards for process models for DEA can be divided as –

1. Semiformal/Formal and graphical representations (Non logic based) – these can be further subdivided into two categories –
    a. Semiformal/lightweight formalisms that support graphical representation – UML/SysML, OPM.
    b. Formal representations that support graphical representation and support reasoning - frames and semantic networks
2. Logic based and ontology languages – Knowledge Interchange Format (KIF), Common Logic (CL) that are semantically based on formal logic. Ontology based languages as devised or encoded from formal logic also belong to this category. These include ontologies encrypted with both Description Logic (DL) and First Order Logic (FOL) based semantics such as Web Ontology Language (OWL) based on DL, Process Specification Language (PSL) and IDEF5 based on FOL and rule languages such as RuleML, RIF based on horn logic semantics. SWRL is an example of hybrid representation standard as derived from logic-based approach. Ontology language such as Gellish in the form of STEPlib is not based on formal logic. Although not officially from the logic paradigm, production rules can be considered as knowledge representation (KR) where production rule dialects have been devised for both RIF and RuleML.
3. Schema based representations – STEP schemas modelled and represented in EXPRESS language, RDF/RDFS with XML serialisation
4. O-O (Object-oriented) programming languages – examples are LISP, Java and C/C++ as programming languages, which can be used to implement schemas and models for machine interpretation such as UML/SysML models as well. They use classes and methods to represent the behaviour of the objects. The attributes are encoded in the class description. They are also used to embed design automation facilities for e.g.

proprietary CAD enabled automation such as CATIA knowledgeware uses C++ and AML as a proprietary KBE system (TechnoSoft Inc, 2003) is based on a different and much more dynamic language in the form of LISP thus making it generative and demand driven along with enabling dependency backtracking. A lot of other proprietary KBE systems such as GenDL are also based on dialects of LISP originated languages. LISP embeds multi-paradigm programming features on top of O-O programming.

3.6.2 Reasoning: DEA

Reasoning techniques for DEA systems or pertaining to knowledge based engineering representation can be broadly classified as follows – rule based (forward chaining and backward chaining), case based and model based (Van der Velden, 2008). There are other reasoning techniques such as fuzzy logic and neural networks. Reasoning can be classified as monotonic reasoning and non-monotonic reasoning. Monotonic reasoning indicates that a conclusion once inferred from the knowledge base can't be altered if new knowledge entered is related to the conclusion. On the other hand, non-monotonic reasoning allows conclusion once inferred from the knowledge base to be altered if new knowledge entered is related to the query (Ivanov et al., 2015; Olivetti, 2011; Poole and Mackworth, 2010). Thus non-monotonic reasoning adopts closed world assumption (CWA) in the sense unless new information is added, the knowledge base assumes the knowledge base is complete. As and when the new information is added the generated results can be altered. For example, production rules follow CWA. On the other hand, monotonic reasoning follows open world assumption (OWA) in which even after new information is added, the results generated by the reasoning engine don't change. DL support monotonic reasoning and follow OWA. Thus languages such as OWL based on DL support monotonic reasoning following OWA.

There is always a trade-off between reasoning and expressive power of a formal representation standard (Yahia et al., 2012). Thus the relationship between expressiveness

and reasoning is inverse, the more expressive the language its decidability or efficiency of reasoning decreases. Although FOL offers more expressiveness as compared to DL, it does so at the expense of computational efficiency in reasoning.

3.7 Description of formal representation standards

Process models can be shared across multiple domains using different representation formalisms but this may have problems due to syntax, semantics and axioms. The objective of the following section is to discuss and narrow down a few existing neutral formal representation techniques of the informal/semiformal model in terms of these issues that should help integration with multiple platforms and provide interoperability. The explanation will be performed in accordance with the classification of formal representation standards in section 3.6.1.

3.7.1 Object Oriented (O-O) modelling standards – UML and SysML

Both MBSE languages in the form of UML and SysML as O-O modelling languages have been discussed in section 3.4. They can also be referred as lightweight formalisms or semiformal representations. UML uses Object Constraint Language (OCL) in order to define rules and constraints for consistency checking across models (Vaziri and Jackson, 2000). UML data models follow CWA (Hennig et al., 2015). SysML is a language that can be used for capturing and representing of process-based data for a complex system and can be viewed as a formal representation with the help of tools such as visual paradigm. SysML models, once created, can be exchanged via ISO AP 233 of STEP(Weilkiens, 2007). Some of the important APs of STEP for consideration are AP233, AP213, Part49 and AP242 for formal storage of informal process models. However, both UML / SysML can capture process and product semantics in a lightweight formalism approach which needs to be transformed to a formal layer, which ensures common semantics through its axioms (Chungoora et al., 2013a).

Researchers based at NIST have used UML based lightweight neutral representations for product knowledge such as form, function and behaviour along with design rationale for developing Core Product Model (CPM) and product assembly features such as tolerances, kinematics at system level for Open Assembly Model (OAM) (Fenves et al., 2008; J. H. Lee et al., 2010; Rachuri et al., 2006; Sudarsan et al., 2005). There are other concepts related to product structure such as part/assembly and extensible geometrical knowledge such as features, tolerance, material and manufacturing process as well. Along with these product structure and manufacturing concepts, function, behaviour and design rationale have been represented for knowledge sharing using UML class based representation in CPM/OAM for product knowledge in context to PLM systems (Jae H. Lee et al., 2010; Jae Hyun. Lee et al., 2010; Rachuri et al., 2005; Sudarsan et al., 2005).UML and SysML based representation such as class diagram, block diagram, parametric diagram have been used for functional and behavioural representation of mechatronic products (Alvarez Cabrera et al., 2009; Woestenenk et al., 2010). Design rationale has been discussed as the decision making reasons for engineering design and manufacturing activities and has been represented using UML based lightweight notation in context to CAD systems with interaction through an application programming interface (API) (Poorkiany et al., 2016) and generic product models as part of PD (Medeiros et al., 2005; Nomaguchi and Fujita, 2013). Design rationale was successfully captured using Design Rationale Editor (DRed 2.0) utilising UML class diagram with object classes and relationships based on an initial version of DRed with functional analysis in collaboration with Rolls Royce for turbine blades in context to PLM systems (Bracewell et al., 2009a, 2009b, 2004). DRed/DRed 2.0 as graphical representation were developed after the limitations of a previous informal representation for design rationale in the form of Issue Based Information System (IBIS) was realised (Eng et al., 2011).

3.7.2 Object – Process Methodology (OPM)

As discussed earlier, OPM as a methodology enables formal representation in the form of Object Process Diagrams (OPD) and Object Process Language (OPL) (Dori et al., 2010). OPM models can be converted to other modelling languages and notations such as BPMN, UML/SysML as well (Grobshtein and Dori, 2011). However, it uses RDF/XML based representation of its unified object-process viewpoint of a system (Dori, 2004). Following the model based system paradigm (MBSE), Tesperanto language was developed as a next layer to OPL as an enhancement. It is also referred as 'Technical Esperanto'. The main purpose of Tesperanto both as a methodology and language is to improve the quality of technical knowledge in a document following the structure of OPM methodology (Blekhman et al., 2015; Blekhman and Dori, 2013). One of the very important criteria here is that OPL is suitable as a low level language for machine readability and code generation but not very clear and concise for human interpretation. Tesperanto as an enhancement on top of OPL makes it more human readable. Tesperanto enables both model to text generation and text to model generation (Blekhman et al., 2015; Blekhman and Dori, 2013). Thus, in-spite of this strength, this research would be deviating away from Tesperanto as it is more focussed on high level representation of knowledge from a technical document whereas OPL is more focussed on low level machine interpretation.

3.7.3 Frames and Semantic Networks

Frames are a formal method of representing an entity and its associated attributes and values (Minsky et al., 1975). They consist of data structures in the form of slots for allocating the attributes and values for a particular object (La Rocca, 2011; Obitko, 2007a; Prasad, 2006; Robin, 2013). The slots can have both values as attributes as well as encode methods or rules. They can also encode process knowledge or a production rule. Frames provide encapsulation and inheritance of object properties through slots, so in this manner provide similarities with

O-O paradigm. Through inheritance they can show interdependencies between object properties. Frames can exhibit declarative knowledge through attributes and procedural knowledge through methods (Negnevitsky, 2005). Models can be built using frames referred as frame based models or systems (Obitko, 2007a; Wang et al., 2006). These models use inheritance of slot values and attributes for marking interdependencies between various frames. An example of a frame-based model is Open Knowledge Base Connectivity (OKBC). OKBC can use frames properties to create various instances of a class and follows the O-O paradigm. Frames allow reasoning through two methods in the form of when-needed and when-changed (La Rocca, 2011). For 'when-needed' the system executes and generates the value of a slot when demanded by a user. For 'when-changed', often referred as demons, the system executes and generates the value of a slot as soon as the user makes any change.

Semantic networks (Semantic nets) were introduced by Margaret Masterman in 1961 (Sowa, 2008a). Semantic nets, also referred as concept network, is a graphical representation which uses vertices or nodes to illustrate concepts and edges to illustrate relations between the concepts (Obitko, 2007b). Semantic nets are mostly used for representing propositional information (Robin, 2013) and thus are also referred as propositional net. The vertices can represent physical objects or concepts. Semantic nets also support automated systems for reasoning on the knowledge represented (Sowa, 2015).

3.7.4 Ontology Languages

Various ontology languages can be devised from DL and FOL. As stated in section 3.6, PSL, OWL and IDEF5 are ontology-based representations. An ontology-based approach helps formalise the concepts and provides axioms as a formal means of constraining the meaning of the concepts in the language. Ontology is defined as the taxonomy of concepts and their definitions supported by a logical theory. Ontology defines a set of terms, entities and objects, classes and relationships along with formal definitions and axioms to constrain the

meaning of terms (Pouchard et al., 2000). Ontology can also be defined as ‘a requirement for conceptualization and illustrates a set of representation primitives with which a domain of knowledge can be modeled’. It provides machine-readable syntax for a domain knowledge (Mizoguchi, 2003). Using ontology, declarative formalism is used to represent domain knowledge as a set of objects. This set of objects represented is referred as universe of discourse (UoD) (Gruber, 1995). Thus ontology enables interoperability and re-usability of the data using common semantics of modeled information.

All ontology languages don’t offer same expressivity. The level of expressivity of an ontology language is governed by its mathematical foundation in the form of logic (Dartigues et al., 2007). Logic can be defined as a precise and accurate notation for expressing and representing statements that can be judged whether true or false (Sowa, 2007).The use of mathematical logic supports automated reasoning. Some ontology languages are based on DL such as OWL whereas some ontology languages are based on FOL in the form of predicate logic such as PSL, IDEF5. DL can be considered as a subset or a decidable fragment of FOL (Obitko, 2007c). According to NIST, ontology languages can be classified as frame based, description logic, predicate logic and hybrid (Barkmeyer et al., 2003).

Some of the other ontology-based representations not based on formal logic, are Core Plan Representation (CPR), Workflow Process Definition Language (WPDL), and Planning Domain Definition Language (PDDL). The ontologies for WPDL and PDDL do provide common semantics but are unable to provide axioms as a formal means of maintaining the semantics in the language (Gruninger, 2004).CPR (Pease, 1998) was initiated by the Defense Advanced Research Projects Agency (DARPA)-sponsored Object Model Working Group (OMWG). The basic concepts in CPR are action resource, actor, and objective with additional concepts such as plan and time point. However CPR as a language does not enable representation of all design decomposition features through its ontology.

3.7.5 Description Logic Based Languages

Description logic (DL) is a knowledge representation (KR) formalism that evolved from semantic networks and frames but was considered as a subset or fragment of first order predicate logic (FOPL) (Baader et al., 2003; Wang et al., 2004). DL is primarily used for representing formal description of concepts and relations (Obitko, 2007d). A knowledgebase formalised by DL illustrates two components – 'TBox' and an 'ABox' (Baader et al., 2003). TBox exhibits intensional knowledge through terminology that is the concepts and their roles. ABox illustrates extensional knowledge also referred as assertional knowledge, which is relevant to the individuals for a particular domain of discourse. Thus DL based representations represent domain knowledge by first defining relevant concepts of the domain in the form of terminology and then using the concepts to specify the properties of objects and individuals in the domain. Pertaining to this research, the domain is the engineering design process for DEA. Languages based on DL support automated reasoning.

3.7.5.1 Web Ontology Language (OWL)

OWL is a web ontology language based on DL for creating and sharing ontologies on the World Wide Web and is regarded as a W3C recommendation (Bechhofer, 2009). OWL was developed as an extension of the Resource Description Framework (RDF) and is derived from the (DAML + OIL) ontology. OWL has three variants – OWL Lite, OWL DL and OWL Full(Wang et al., 2006, 2004). OWL lite offers ease of implementation but offers the least of the OWL constructs. It is based on description logic SHIF. OWL DL is based on descriptive logic and offers more constructs and, more importantly, reasoning ability. It is based on description logic SHOIN. OWL Full offers the most comprehensive constructs but deviates from reasoning ability and offers less ease of computation compared to OWL DL (Obitko, 2007e). OWL-S, as a semantic markup for web services built on OWL, enables viewing of process with inputs, outputs, parameters, precondition and results (Martin et al., 2004). Thus

selection of a particular OWL Language is critical in order to represent design decomposition features (Bechhofer, 2009; W3C, 2012). OWL is built upon RDF/XML and RDFS supports interoperable ontological representation of concepts over the semantic web and enables automated reasoning (Bechhofer, 2009; Hay, 2006; W3C, 2012). It follows OWA (Hennig et al., 2015). It imposes cardinality upon its classes and properties. OWL adds properties such as relations between classes for e.g. disjointness, cardinality of properties, transitivity as compared to RDF Schema (RDFS)(Zhao and Liu, 2008a).

3.7.5.2 Usage of OWL in Engineering Design, Manufacturing and DEA

OWL ontology models for detailed product models including assembly features such as tolerances, kinematics at system level in OAM along with function and behaviour in CPM/OAM as abstract concepts have also been developed for usage in PLM systems (Fiorentini et al., 2007; Sarigecili et al., 2014). OWL ontology has been demonstrated for manufacturing domain for extensive usage with all machining processes for example MASON and ONTO-PDM (Chang et al., 2010; Lemaignan et al., 2006; Panetto et al., 2012). OWL ontology has been used for modelling and formal representation of design rationale for product knowledge (Li et al., 2014) and also in context to CAD systems (Witherell et al., 2007). Ontology based representation for function and behaviour representation for various products such as gears, shafts and conveyors with focus on knowledge management has been performed with querying on the ontology models (Kitamura, 2006; Kitamura and Mizoguchi, 2004). The advancement of DRed 2.0 for knowledge modelling of design rationale for turbine blades design in the context of PLM systems utilising UML class diagrams was formally represented using OWL/SWRL ontology for computational and system processing of the information (Bracewell et al., 2009a).

Product semantic representation language (PSRL) is another ontology-based language, which is based on (DAML + OIL) and enables open standard usage. It focuses on neutral

representation of product data. Various concepts of non-geometric information such as design rationale, function, behaviour and part dependencies form an integral part of product data (Patil et al., 2005). PSRL based on DL with its syntax based on RDF/XML can be used for product data modelling and computer aided process planning (Liu et al., 2010).

Work has been performed to develop semantic product models with geometric kernels using OWL/SWRL ontology across heterogeneous CAD systems with various product features and shapes such as surfaces, faces, edges, vertices, product parameters, datum planes and axis of rotation (Dartigues et al., 2007; Lu et al., 2016; Noh and Suh, 2008; Qin et al., 2016; Tessier and Wang, 2013; Zhan et al., 2010). Similarly, OWL has been used as neutral formal logic representation language with automated reasoning in context to consistency checking and reducing redundancies during design stage for product models with geometric representations as per heterogeneous CAD and PLM systems (Franke et al., 2011).

The use of OWL ontology with formal data structures for engineering design knowledge management with design process functional requirements, manufacturing processes, material selection for representation along with inference and querying for automation has been performed (Kitamura and Mizoguchi, 2004; Li et al., 2009; Li and Ramani, 2007; Mehrpoor et al., 2013). The role of OWL ontology in the context of DEA with a KBE approach has been adopted and verified (El Kadiri et al., 2015; Furini et al., 2016; Kitamura and Mizoguchi, 2013).

3.7.6 First-Order Logic Based Languages

First order logic (FOL) is commonly used as a basis for KR enabling automated theorem proving and usage across the semantic web (Gruninger et al., 2013). FOL extends the expressiveness of propositional logic by adding quantifiers and variables to the existing propositional connectives of conjunction, disjunction, negation, implication and bi-conditional. A universal quantifier expresses that a relation holds true for all instances of a

variable whereas an existential quantifier expresses that a relation holds true for some specified instances of a variable (Gruninger et al., 2013). DL acts as a subset of FOL.

A graphical representation based on semantic nets and existential graphs is Conceptual Graphs (CG's). CG's provides a logic formalism to illustrate classes, relations, individuals and quantifiers (Obitko, 2007f; Sowa, 2008a). The simple version of CG's is referred as Core CG's and evolved from simple existential graphs developed by Charles Sanders Peirce. Extended CG's provide a superset of the core CG's (Sowa, 2008a). Although the graphical representation of CG's in its linear form (Conceptual Graph Display Form) evolves from semantic nets but the CG's express same semantics as FOL based on predicate calculus also referred as first order predicate logic (FOPL). The instances of concepts are represented in rectangle and relations between concepts as ellipse or circle. Some of the logical operators used by Conceptual Graph Display Form are conjunction and existential quantifier in order to translate the natural language to logic formalism. The formal representation of CG's is referred as Conceptual Graph Interchange Format (CGIF) is a part of Common Logic (CL) in the form of ISO 24707 (Sowa, 2011). CL referred as ISO/IEC 24707 was developed as a framework for a family of logic based languages to allow information sharing and exchange with standardised syntax and semantics (Gruninger et al., 2013; Sowa, 2008b). CL evolved from both CG's and KIF to be built into single ISO project in the form of ISO/IEC 24707 (Sowa, 2008a). CL offers three dialects –

- Common Logic Interchange Format (CLIF)
- Conceptual Graph Interchange Format (CGIF)
- XCL – XML based notation for Common Logic

(Obitko, 2007g; Sowa, 2011, 2008a, 2008b)

CGIF also exists in two forms – core CGIF and extended CGIF (Sowa, 2008a). Core CGIF expresses full semantics of CL. Its dialect maps to Pierce's existential graphs. Core CGIF

uses primitives such as conjunction, negation and existential quantifier. Extended CGIF adds universal quantifier, type labels for restricting the range of quantifiers, Boolean contexts with type labels such as - If, Then, Either, Or, Equivalence, and Iff, and the option of importing external text into any CGIF text (Sowa, 2008a). Thus CL can be used as a logic based formalism for representing knowledge and allowing automated reasoning. It can be used as a neutral representation of knowledge allowing re-usability (Gruninger et al., 2013).

KIF (Genesereth et al., 1992) as a computer-oriented language was developed by the Interlingua Working Group of the DARPA knowledge sharing effort (Knutilla et al., 1998). KIF as a language expresses its semantics in first order predicate logic and is syntactically based on LISP (Hayes and Menzel, 2001; Obitko, 2007h). It has formally defined semantics and breaks down knowledge into the form of objects with related attributes, processes and functions. Thus it aligns its methodology with OPM (Dori, 2002) and solves a major issue of pre-defined formal semantics. As stated earlier, OPM as ISO 19450 forms a part of ISO TC 184 / SC5 (ISO, 2015). ISO TC 184 is managed by the International Standards Organization (ISO) and covers "Standardization in the field of industrial automation and integration concerning discrete part manufacturing and encompassing the applications of multiple technologies, i.e. information systems, machines and equipment and telecommunications" (Pouchard et al., 2005).

3.7.6.1 Process Specification Language (PSL)

To address the shortcoming of formulating common semantics and as a standard for the exchange of process specification, PSL was designed to facilitate correct and complete exchange of process information among manufacturing systems, such as scheduling, process modeling, process planning, production planning, simulation, project management, work flow and business process re-engineering (Grüninger and Menzel, 2003). A major purpose of PSL was to enable interoperability of processes utilising different process models and process

representations (Pouchard et al., 2005). PSL ontology is written in KIF format and forms ISO 18629 as an integral part of ISO TC 184 (Pouchard et al., 2005). PSL ontology is based on FOL(Pouchard et al., 2000). Ontologies based on FOL exhibit more expressiveness compared to DL and can run inference on the modelled information. KIF exists as a predecessor to CLIF (Gruninger et al., 2013). Thus PSL can be considered as a process ontology language based on CLIF (Gruninger et al., 2013; NIST, 2008, 2007). PSL architecture consists of two parts – PSL Core (Foundation theories) and a set of extensions which can be mapped to EXPRESS schemas, UML and XML (Gruninger and Cutting-Decelle, 2000; Pouchard et al., 2005).

PSL ontology is divided into the following four theories – Core theories, Duration and ordering theories, Resource theories and Actor and agent theories (Gruninger, 2004). The PSL core provides four kinds of elements as primitive classes – object, activity, activity occurrence and time point. Within PSL ontology, 'activity' can be stated as 'a repeatable pattern of behaviour' and 'activity occurrence' can be stated as 'concrete instantiation of this pattern' (Grüninger, 2009). A crucial difference between activity occurrence and time point is that activity occurrence have preconditions and effects in the form of postconditions whereas time point just follow linear ordering of time and don't have any preconditions and postconditions. The three relations in the PSL core are – before, occurrence_of and participates_in and the two functions are beginof and endof (NIST, 2004). To represent an activity-based description, PSL uses an activity role declaration (ARD) along with object declarations to describe objects being affected by the activities of the process (Grüninger and Menzel, 2003). The extensions allow for temporal relations between activities. Thus the use of extensions with experimentation may be used for representing design decomposition features other than the core theory. As PSL deals with standardized syntax and semantic sharing of modeled information, it is consistent with ISO 10303, ISO 13584 (PLIB) and ISO

15531 (MANDATE) (Gruninger and Cutting-Decelle, 2000). PSL axioms can represent inputs, outputs and parameters at both activity and the activity occurrence level but mainly focus on process specifications as opposed to process execution at run time (Bock and Gruninger, 2004). An external automated theorem prover is required for execution of PSL specifications as inference (Bock and Gruninger, 2005).

3.7.6.2 Usage of PSL in Engineering - Manufacturing and Production

PSL core through its object and activity description can represent object material and resources as inputs and outputs for product realisation along with activity interdependency in complex manufacturing processes (Qiao et al., 2011). PSL extensions allow for sequencing and ordering of activities including OR, AND relations and inclusion of sub activities thus allowing process logic. PSL extensions can also represent object features and form such as planes, edges and surfaces in correlation to activity flow from a manufacturing point of view for example machining activities such as milling, drilling, reaming, turning, boring and grinding. It represents the knowledge in concise neutral formal semantics for interoperable machine interpretation (Qiao et al., 2011).

For the aerospace industry, process ontologies such as PSL have been used and validated for knowledge sharing and decision-making for PD but mainly for manufacturing and production domain with knowledge sharing across product design such as those developed by (Usman, 2012; Usman et al., 2013) and (Chungoora, 2010; Chungoora et al., 2013a). Work performed by both Usman and Chungoora focussed on machining processes and the knowledge accessibility with engineering design. Min_precedes as a PSL axiom was extensively used to model manufacturing process flow and sequencing by (Usman, 2012). Min_precedes is transitive which can be accessed during inference. However, their applicability has been demonstrated for wide usage in PLM Systems. PSL has been demonstrated for process modelling for paint and dry manufacturing process with focus on activity inputs and outputs

along with object description (Grüninger and Menzel, 2003). PSL has also been used for process specification for cutting process by (Deshayes et al., 2005). PSL was effectively used as a neutral representation of process specification for exchange between heterogeneous manufacturing software applications such as process planning, scheduling and workflow execution (Schlenoff et al., 1999).

IDEF5 is another ontology-based formal representation based on the basic concepts of IDEFX series. It is also written in KIF format and is based on FOL (Benjamin et al., 1994). The IDEF5 ontology language comprises two languages: the IDEF5 Schematic Language and the IDEF5 elaboration language. The schematic language is a graphical language that allows input of information through an automated ontology capture tool. The elaboration language is a structured text language with full expressive power of FOL which allows input of information with detailed context (Benjamin et al., 1994). It enables storage and representation of classes, kinds and first and second order relations as well through the ontology. Both PSL and IDEF5 as ontology representations based on FOL initially evolved from KIF format, which originated in LISP application.

3.7.7 Gellish

Gellish is a neutral ontology called STEPlib, although not based on formal logic. Gellish is extensible and includes concepts from ISO 15926 and ISO 10303 (Van Renssen, 2003, 2005). Gellish is fact oriented instead of being purely O-O and can represent relations between two objects with preserved semantics. Some of the basic concepts in Gellish are – anything, role, relations such as plays role & requires role, individual things, kind of things along with single and multiple things with specialisation of classes. Gellish models can be exchanged by different application domains using XML (Van Renssen, 2003). It can be used for representing both product knowledge as well as design process knowledge including function and behaviour of an artefact.

3.7.8 Rule Languages (Logic based)

Rule Markup Language (RuleML) is a format or a language for representing and sharing of rules on the World Wide Web. It is based upon XML, RDF and OWL (Boley et al., 2005). RuleML also offers 2 modular sublanguages – Derivation RuleML and Production Rule (PR) RuleML (Hirtle et al., 2006). RuleML has 3 parts as different specifications – Deliberation RuleML, Consumer RuleML and Reaction RuleML(Boley et al., 2016a, 2016b, 2016c).

Another language in the form of Rule Interchange Format (RIF) offers a neutral representation language for representing rules, logic and constraints. RIF offers 3 dialects – Core, BLD (Basic Logic Dialect) and PRD (Production Rules Dialect)(Feigenbaum et al., 2013; Kifer and Boley, 2010; Morgenstern et al., 2012). RIF core is the basic language and offers the least constructs or expressiveness. It is also based on XML format similar to RuleML. RIF BLD offers logic functions along with equality and built-ins as per positive horn logic. RIF PRD adds forward chaining of rules to RIF BLD (Feigenbaum et al., 2013). RIF offers a major advantage as it can be expressed in both XML-based syntax and more importantly can be extended to AP242 of STEP(Lützenberger et al., 2012). It can integrate with any platform or a CAD/PDM platform (Colombo et al., 2014).

Semantic Web Rule Language (SWRL) combines OWL DL constructs with Unary/Binary Datalog subset of RuleML (Horrocks et al., 2004; Kuba, 2012) . Thus it allows horn logic rules to be expressed in addition to OWL concepts(Glimm et al., 2009; Zhao and Liu, 2008a). SWRL includes basic functions such as comparison, boolean, strings and math such as multiply, divide, sin, tan, pow (Golbreich, 2004). Semantic Web Services Language (SWSL) as a language consists of two languages – SWSL-FOL as a first order logic based language for defining formal ontology for process models and SWSL-Rules as a rule based language (Battle et al., 2005).

Reasoning on the rule is performed in many ways. Forward reasoning and backward reasoning are some of them. Forward reasoning is referred as data driven or eager approach whereas backward reasoning is referred as goal driven or lazy approach (Negnevitsky, 2005). In forward reasoning, the system matches the statement against and existing rule and generates all results, which match the statement. In backward reasoning, the statement is allocated, as hypothetical goal and the rule will be generated which matches the goal statement. Backward reasoning takes less time as compared to forward reasoning and only provides specific solutions whereas forward reasoning generates all possible solutions and takes more time.

Ontologies have been implemented using OWL for engineering design knowledge primarily including product model and engineering rules using SWRL on top of OWL for DEA (Sanya and Shehab, 2015, 2014). Similarly, engineering rules have also been formalised using RIF-PRD and Content MathML on top of OWL for DEA by (Reijnders, 2012) and RIF for LinkedDesign project by (Colombo et al., 2014; Klein et al., 2014). MathML is also based on XML syntax and provides 2 versions for representation of math based rules – Presentation and Content MathML (Ausbrooks et al., 2014). Presentation MathML provides an inbuilt library of about 30 elements and Content MathML is more exhaustive with an inbuilt library of 120 elements with functions for complex equations such as partial differentiation and matrix on top of basic functions (Bos et al., 2011; W3C, 2016).

3.7.9 Schema based Languages – STEP and VRML

Another important ISO standard for product data exchange is STEP which is also regarded as ISO 10303 (Pratt, 2001; Zha and Du, 2002). STEP is widely used in industry for representing and exchanging CAD data in a neutral format (H. Wenzel et al., 2011). STEP not only covers exchange of geometric information between different CAD formats but includes all product data throughout the lifecycle (Lützenberger et al., 2012; Tang et al., 2001). STEP uses

EXPRESS (ISO, 2004) as a modelling language to represent objects with related attributes and properties and adopts features from O-O modelling approach (Krima et al., 2009; Peak et al., 2004). EXPRESS provides inheritance of objects with data types to represent complex relationships. Although EXPRESS is machine-readable it can represent only static knowledge and cannot be executed in its original form (Dong et al., 1997; Tang et al., 2001). The semantics of the product data in EXPRESS schema is not explicitly specified (Krima et al., 2009; Sarigecili et al., 2014).

STEP allows various formats for product data representation. Some examples are – ISO 10303-21 for text format, ISO 10303-28 for XML serialization, ISO 10303-22 for API, ISO 10303-41 for product identification and product configuration and ISO 10303-46 for visual representation (Weilkiens, 2007). STEP, UML, Parts library (PLIB), PSL, Manufacturing Management Data Exchange (MANDATE) are examples of standardized exchange specifications for sharing of product and process information in industrial data (Chandrasegaran et al., 2013). STEP as ISO 10303, PSL as ISO 18629 along with MANDATE as ISO 15531 all comprise part of ISO TC 184/SC4.

Many conversion mechanisms have been devised from STEP to OWL/SWRL in context to engineering design. Work has been performed to convert STEP EXPRESS schemas to OWL/SWRL models for development of detailed neutral and interoperable product models with geometric knowledge for visual display (Zhao and Liu, 2008a, 2008b). Similar work has been performed to integrate STEP schemas such as Application Protocol (AP) 203 and Part 21 using EXPRESS schemas to OWL/SWRL based ontologies in order to develop interoperable product models with geometric knowledge such as Onto-STEP and ONTO-PDM (Barbau et al., 2012; Krima et al., 2009; Panetto et al., 2012).

Virtual Reality Modeling Language (VRML) is a neutral format for 3D rendering of geometry and allows exchange of product's geometric intent and knowledge (Hartman and

Wernecke, 1996; Qin et al., 2003; Web3D, 2017). It offers ease of sharing over the web as compared to STEP, whichdoesn't support integration over the web for e.g. STEP AP 203. However, VRML doesn't successfully render complete geometric information and retain all intricate features as compared to STEP for efficient product realisation (Cooper and LaRocca, 2007; Szykman et al., 2000a). X3D, which is XML, based for 3D models also offers ease of sharing over the web(Web3D, 2017).It is a successor to VRML and is more comprehensive.

3.7.10 Schema based languages - Semantic Web Base Standards

RDF offers representation of information over the World Wide Web and is regarded as a W3C recommendation (Klyne et al., 2004; Manola et al., 2004). The syntax of RDF describes information by breaking it into a triple form consisting of subject, object and predicate. It also offers a formal graphical syntax in the form of an RDF Graph. The Uniform Resource Identifier (URI) is an id, which locates the address of the information over the web. The most critical aspect of RDF is that it uses XML-based syntax and schema (Klyne et al., 2004).RDF as a data model for objects and relations provide a simple semantics. RDF schema (RDFS) provides generalisation of classes and properties (Dean et al., 2004; Mcguinness and Van Harmelen, 2004). XML can be defined as a universal metalanguage for defining markup and allows interchange of data between various disparate applications (Antoniou and Van Harmelen, 2004). XML provides a formal neutral machine interpretable syntax for data. XML uses a tagging based approach similar to HTML and can be used for various purposes like marking information in design documents, process information and product models. However, a shortcoming of XML based representation or tagged information is that it doesn't provide clear semantics to the data (Antoniou and Van Harmelen, 2004). This indicates that the meaning of the information can't be constrained as sematic clarity and is thus open to interpretation.

An example of a format for requirements for automotive products based on XML schema as open standard is Requirements Interchange Format (ReqIF) (OMG, 2013). Similar to UML, ReqIF is an OMG specification format and provides neutral representation for requirements such as functional requirements between proprietary tools thus enabling open standards usage and providing interoperability.

3.7.11 Object-Oriented (O-O) programming languages

Object-Oriented (O-O) programming languages such as LISP, Java, C/C++, Smalltalk, Python can be considered as formal representation or knowledge representation standards (La Rocca, 2011). O-O techniques vary from modelling methods or standards such as UML, SysML and programming languages, which are executable and dynamic as opposed to UML/SysML, which are static in nature.

As an O-O language, Java can be defined as 'A simple, object-oriented, network-savvy, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dynamic language' (Toussaint and Cheng, 2002, Pg 335). Java is increasingly used for developing client-server applications especially over the web. It allows for calling of information over databases and ontology models as knowledge base enabling automation and offers cross platform usage with its source code for e.g. through an API such as Apache Jena Framework (Toussaint and Cheng, 2002). Work performed for DEA using OWL/SWRL ontologies was converted for visualisation using Java by (Sanya and Shehab, 2015, 2014). Java enables cross-platform usage as it supports network programming, as compared to other programming languages such as C/C++ for which explicit codes need to be written to enable its cross platform usage (Reilly, 2006). Java code also allows for interaction with neutral format product models such as VRML which can be shared over the web (Qin et al., 2003; Zeng et al., 2003). Java can also be used for generating code from O-O modelling methods such as UML (France et al., 2006). Java, C++ and Python are all high level programming

languages (La Rocca, 2011). Even C++ code can be generated from domain models maintained for design automation applications (Bermell-Garcia, 2007). Both python and C++ codes were used to perform DEA in context to OWL ontologies by (Reijnders, 2012).

LISP is also a high level programming language and stands for LISt processing (Foderaro, 1991). LISP supports declarative approach as well along with procedural approach as compared to basic O-O programming languages such as C, which are purely procedural in nature. Thus along with defining LISP allows for change of its own source code thus allowing extensions to its own syntax and create supersets (Lützenberger et al., 2012), which result in languages such as Common LISP. Thus Common LISP follows a multi-paradigm approach by supporting both declarative approach and procedural programming as it evolved primarily from O-O approach (Evenson et al., 2015). KBE applications vary from most O-O languages in the sense that they imbibe declarative nature along with the facility of procedural programming as opposed to purely procedural nature of basic O-O languages (Prasad, 2006). Because of the advantages of LISP as compared to other O-O languages such as Java, C++ in the form of being declarative in nature and allowing extensions in its own syntax thus creating supersets of its own syntax in the form of Common LISP as a superset of LISP, its various dialects are used for creating and building KBE automation applications (Lützenberger et al., 2012; Phillip Sainter et al., 2000).

Some of the existing proprietary KBE applications such as Adaptive Modelling Language (AML) from Technosoft is based on O-O techniques (TechnoSoft Inc, 2003). AML is primarily based on LISP dialect (Preston et al., 2004; Rocca, 2012) but also uses C++ and Fortran codes (TechnoSoft Inc, 2003). AML focuses on automation of product design throughout product lifecycle. From AML's perspective, capturing knowledge in the form of objects and properties is critical. AML performs this by defining class definitions for similar objects and properties in the methods. It also supports class-subclass relation and is dynamic

in nature. It supports constraint mechanism in product alteration by making interdependencies or dependency backtracking in the unified model along with relation to parameters. It also invokes events. ICAD is based on ICAD Design Language (IDL), which being a proprietary KBE application is based on a superset of LISP code in the form of ACL LISP implementation allowing for declarative nature (Bermell-Garcia, 2007; Bermell-García and Fan, 2002; La Rocca et al., 2002). Similarly, General Purpose Declarative Language (GDL) from Genworks as a proprietary KBE language is also based on ANSI standard version of Common Lisp and uses Common Lisp Object System (CLOS) allowing for declarative paradigm (J Kulon et al., 2006; La Rocca, 2011; Rocca, 2012).

Frameworks such as Apache Jena provide interface to the OWL/SWRL representation and support Pellet reasoner for queries and inference results (Chan, 2013; Zhang et al., 2015). Proprietary DEA applications such as AML, ParaPy are based on O-O programming which also forms the basis of representation of geometry kernels such as LISP, Java, C/C++. O-O programming offers few similarities to ontology-based representation in terms of object and class definition with attributes, encapsulation and inheritance.

DEA in context to KBE is driven highly by engineering rules and thus KBES select production rule formalism in conjunction with O-O paradigm as KR for achieving DEA. Table 3-2 illustrates the available formal representation methods for representation of various design decomposition features as discussed in section 2.6.

Table 3-2: Formal representation methods & techniques available for representing design decomposition features to enable design process automation

Design Decomposition Features	Formal Representation Methods & Techniques	References
Process – Inputs, Outputs and Parameters	PSL, IDEF5, OWL DL, OWL-S,	(Bechhofer, 2009; Benjamin et al., 1994; Bock and Gruninger, 2004; Chen and Chen, 2005; Fellmann et al., 2013; Gruninger, 2004; Grüninger, 2009; Gruninger and Cutting-Decelle, 2000; Grüninger and Menzel, 2003; Martin et al., 2004; Pouchard et al., 2000, 2005; Schlenoff et al., 2000b; W3C, 2012)
Engineering Rules, Logic, Constraints, Rationale	RuleML, Rule Interchange Format (RIF), SWRL with OWL DL	(Bechhofer, 2009; Boley et al., 2005; Colombo et al., 2014; Fellmann et al., 2013; Lützenberger et al., 2012; Lützenberger et al., 2012; W3C, 2012)
Functional Requirements	Requirements Interchange Format (ReqIF), SysML Requirements Diagram	(Colombo et al., 2014; Fellmann et al., 2013; Lützenberger et al., 2012; Lützenberger et al., 2012; OMG, 2013; Weilkiens, 2007)

3.8 Analysis of Informal/Semiformal and Formal Process Modelling Standards for DEA

From the observations of comparative analysis of informal/semiformal modelling standards in context to knowledge capture for achieving design automation in Table 3-1, IDEF suite with main emphasis on IDEF0/IDEF4, UML/SysML, Modified Petrinet, and signposting satisfy the criteria as they successfully capture necessary design decomposition features on a higher level. Petrinet is considered to be one of the methods for process modelling and representation techniques. Although in its original form, it does not enable design decomposition to the required level for process automation, MPN can capture design decomposition features. However, it fails to utilise common semantics and uniformity in axioms (Grüninger and Menzel, 2003). This highly inhibits its use in a neutral representation for achieving automation. Also, Petrinets and MPN have their strength in modelling the synchronisation of concurrent processes, cause and effect relationships between events and states along with evaluation of modelled systems based on precedence of activities (Bock and

Gruninger, 2005; Peleg and Dori, 1999; Zhang et al., 2013). Similarly, although Signposting as a modelling method is successfully able to represent product parameters, the confidence mapping of parameters is not a requirement for DEA as observed from Table 3-1. Also, the confidence mapping of parameters is upon the discretion of the engineer and is not standardised. Thus both Petrinet/MPN and Signposting are unable to capture complex interdependencies of a process model with emphasis on flow of information of design decomposition features such as activity inputs, outputs, rules, function and behaviour in context to achieving DEA.

Final selection of an informal model would be suggested after experimentation on formal representation of the informal model, as all of the necessary design decomposition features in the form of parameters, inputs and outputs, rationale, logic, rules, constraints, attributes, and requirements will need to be formally represented. Existing process modelling techniques are able to represent the design process knowledge at a high level granularity instead of low level granularity with detailed attributes and complex interdependencies of knowledge required for DEA (Ding et al., 2009). Integration of all the concepts of engineering design process as design decomposition features with the complete effect of a re-usable and robust process model on the product attributes is required before its implementation to a formal representation framework (Chalupnik et al., 2006).

Engineering rules can be represented in IDEF0 through control component as functional modelling method and represented formally in rule languages such as RuleML, RIF and as production rules. In the work of Skarka, using OWL, rules are represented textually, but not as executable formal representation with link to product attributes which can return values during reasoning and querying. The reasoner doesn't perform reasoning on rdfs:comment in the model (Skarka, 2007). Process flow can be successfully represented in IDEF3 diagram and UML activity diagrams. Product model can be successfully represented through object

diagrams in IDEF4, UML class diagram, OPM object diagram and even in PSL formal ontology object instantiation. Resources can be represented in IDEF0 as mechanisms. Parameters can be successfully represented in signposting and even in PSL formal ontology with extensions. PSL as a language can be used for process and activity description including inputs, outputs and parameters from the process model with extensions along with representing a product description. In order to enable DEA, the formal representation should enable automated reasoning or inference as execution of its axioms. PSL is similar in representation to a low level assembly language and needs a compiler to convert its representation to a high level language such as C or Fortran (Schlenoff et al., 2000a).

The family of IDEFx series has been very successful at systems modelling (Ciocoiu et al., 2001; Reeker, 1994). IDEF4 design rationale component can provide for design rules under the design rationale component as partitions (Mayer et al., 1992) but engineering rules can't be explicitly stated and with contextual relevance to engineering design process. Design rationale can be represented as an integral component in IDEF4 standard and DRed tool as UML class diagram along with formal representation using OWL ontology. UML has been widely adopted as O-O modelling for software systems (Siricharoen, 2007).UML/SysML diagrams have also been very successful at modelling and representation of function, behaviour and structural aspects of engineering systems with complex interaction along with exchange of knowledge to be consumed with KBE applications (Plateaux et al., 2009).

The results indicate that methods and languages such as the IDEF suite and UML/SysML informally capture most design decomposition features such as objects, processes with inputs, outputs along with resources, attributes, requirements, rules, logic, constraints, and rationale for design process automation. The formal representation framework aims to achieve process automation by representing all design decomposition features dynamically in a knowledge model and then running a query and inference as automated reasoning.

Thus design decomposition features are individually supported through existing informal/semiformal and formal modelling standards. However, as none of the existing methods is successfully able to capture and represent the complete functional, behavioural and structural (F-B-S) aspects of engineering design process knowledge, a novel process model utilising strengths of the existing informal/semiformal standards needs to be developed. The schema of the novel process model as developed will provide a method to effectively utilise existing platform independent and neutral formal representation standards for DEA. The basis of the process model will be analysis of functional requirements for generative modelling along with the effect of the process model on product's geometric attributes.

3.9 Summary

This chapter discusses informal/semiformal and formal standards in order to capture and represent all design decomposition features as F-B-S aspects of a mechanical design process with DFM for DEA. The findings have revealed that none of the existing modelling methods are able to capture the complete mechanical design process knowledge with complex interdependencies with product attributes at an informal/semiformal level. ***Thus a hybrid approach will be adopted to develop a highly granular and integrated novel process model based on IDEF0/IDEF4 and UML/SysML for DEA based on the findings***. The platform independent and neutral formal representation framework of the process model enabling DEA with generative modelling has to satisfy the requirements at an implementation level for the axioms and semantic clarity. In order to recommend the method of schema mapping of the proposed hybrid process model to neutral formal representation with preserved semantics in order to fulfill the primary aim of this research, key concepts and relationships of process model as part of the design decomposition features will be identified for development of Meta model along with experimentation aspects with pilot use-cases. Requirements will be

formulated for the implementation of the Meta model in neutral formal (machine or system interpretable) representation with preserved semantics. The identification of these key concepts and relationships for Meta model along with pilot use case investigation along with compilation of requirements and comparative analysis of formal representation standards will be discussed in the next chapter as part of research design.

4 Key Concepts and Relationships of Engineering Processes for Formalisation with Pilot Use Cases

4.1 Introduction

Chapter 3 elaborated on the existing informal and formal modelling standards, which can capture and represent the mechanical design process with DFM knowledge for DEA. The results of the comparative analysis for informal/semiformal modelling standards suggested that none of the existing standards could fully capture the complete domain knowledge of mechanical design process with DFM for DEA in context to KBE. In order to address the research gap identified in chapter 2 and based on the findings of chapter 3, this chapter will identify key concepts and relations of the process model from design decomposition features for DEA with a KBE approach and formulate the requirements for the platform independent formalisation of the Meta model based on these concepts and relationships with neutral semantics. It will also discuss the pilot use-cases for experimentation with existing formal standards. The comparative analysis of existing neutral formal representation standards as per the compiled requirements will yield the implementation method of the schema of the Meta model based on identified concepts and relationships of the process model.

4.2 A Generic Process Model for DEA with Neutral Formal Representation

Process models can be considered as abstractions of a real process with ambiguity depending upon the level of granularity required for different purposes such as the engineering design process for DEA (Eckert et al., 2015; Maier et al., 2017). As per the domain of engineering design process, both process model and product model are modelled separately but also require integration (Maier et al., 2017). Thus the process model developed for DEA as part of this research provides high granularity and integration with the product knowledge with the behavioural effect of the process model on the change in product's state in terms of its

geometric attributes. The model driven approach adopted for identifying the key concepts and relationships of the process model with its neutral formal representation utilises 3 stages is illustrated with the help of Figure 4-1.

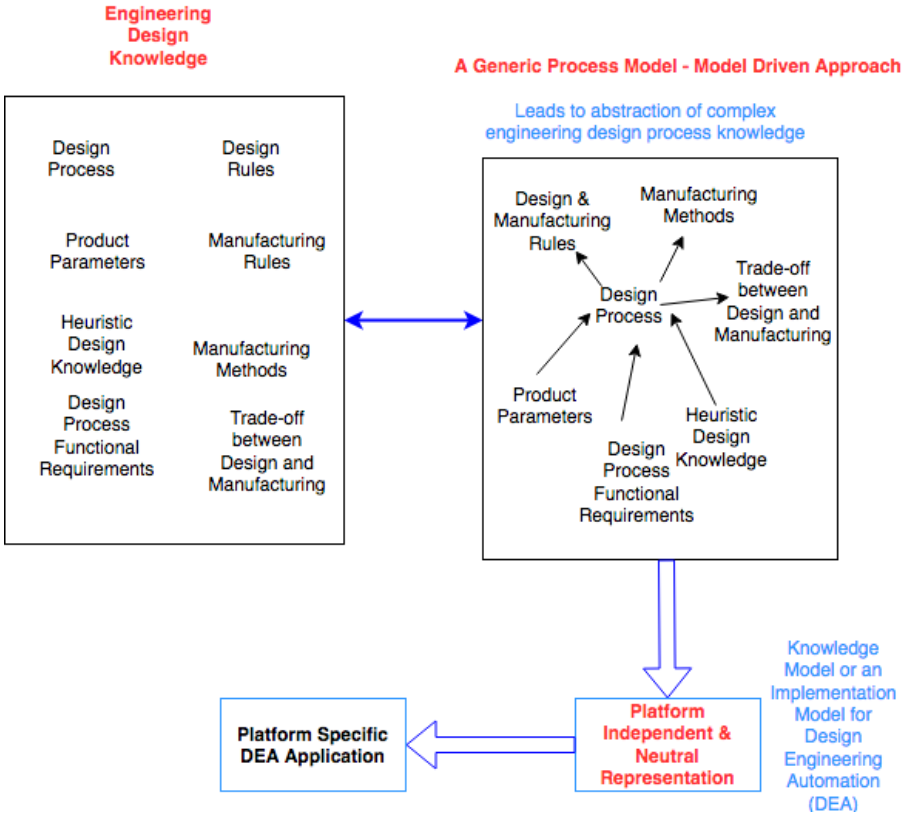


Figure 4-1: Model Driven Approach for Knowledge Modelling and its Equivalent Neutral Formal Representation for DEA

The working of the process model in this research can be divided into 3 steps – Phase 1, Phase 2 and Phase 3. Phase 1 refers to informal/semiformal modelling of engineering design process with focus on mechanical design and DFM for knowledge modelling of all concepts and relationships. This focuses on visual representation using graphical modelling standards. The findings of chapter 3 have revealed that a hybrid approach using existing standards such as IDEF0/IDEF4 and UML/SysML as the basis. Phase 2 refers to equivalent representation of the informal model with platform independent and neutral formal representation as machine or system interpretable axioms using existing standards. This will be continued in the next sections with experimentation with pilot use-cases and requirements analysis for

formalisation for DEA. Phase 3 refers to the automation layer with the help of querying and inference as reasoning mechanism on the formal axioms as part of the verification for the method of schema mapping. The method for developing and testing of the process model is defined in Figure 4-2.

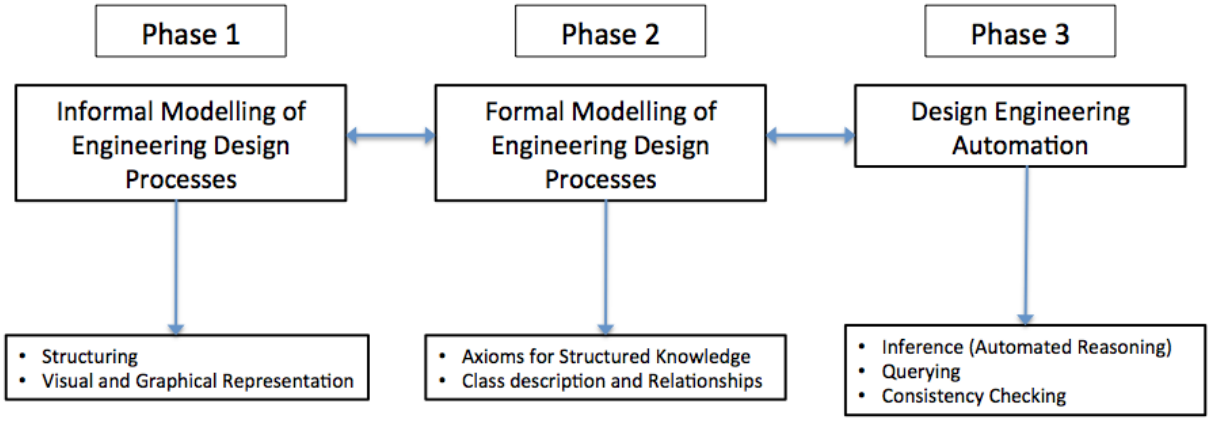


Figure 4-2: Working of the process model for DEA

4.2.1 Phase 1

Firstly domain knowledge of the mechanical product design process with DFM is captured using a model driven approach as a generic process model with concepts and relations with high abstraction. The domain knowledge describes the static information and knowledge objects in an application domain (Schreiber et al., 2000, pg 91). Pertaining to this thesis, the domain knowledge comprises of the mechanical design process and DFM/DFA with activities consisting of inputs, outputs, rules and resources along with process function and behaviour in context to the product attributes.

4.2.2 Phase 2

The process model is finalised in terms of its Meta model based on concepts and relationships before its implementation in platform independent and neutral formal representation standards. The domain schema contains all the concepts and relationships of mechanical design process with DFM/DFA. A domain schema can be defined as ‘a schematic description

of the domain-specific knowledge and information through a number of type definitions' (Schreiber et al., 2000, pg 91). Population of the schema level model or domain schema with instances leads to development of the knowledge base. According to Schreiber, 'a knowledge base contains instances of the types specified in a domain schema' (Schreiber et al., 2000, pg 91). Thus the knowledge base will contain population of the mechanical design with inclusion of DFM aspects as domain schema with instances from all 4 Use cases for experimentation as neutral formal representation standards. Reasoning and querying can be performed as execution of the underlying axioms.

4.2.3 Phase 3

This phase focuses on the accuracy of the reasoning mechanism as inference and querying over the axioms of the knowledgebase along with the completeness of knowledgebase. The reasoning mechanism helps in deduction of new knowledge based on existing axioms; returns answers to the user based on multiple scenarios and provide consistency checking. These are matched to the implementation in a DEA system such as a KBES to verify the correctness of the reasoner in terms of values generated. The values generated will only match correctly if the method of population of schema with instances of the process model to its neutral formal representation is appropriate.

Thus the novel aspect of the solution as part of the research gap is to initially define a core set of mechanical design process Meta model based on concepts and relationships with inclusion of manufacturing knowledge as DFM based on the identified design decomposition features in section 2.6 and findings in section 3.8. These are discussed in the next section 4.3. The other main aspect as the primary aim of this research is the method of schema mapping of the Meta model based on identified concepts and relationships to neutral formal representation with semantic clarity to constrain the meaning of concepts as part of model driven formalisation. For this purpose, the experimentation with pilot use cases for key

concepts is performed in section 4.4 and 4.5. The compilation of the requirements for formalisation is performed in section 4.7 and the comparative analysis for the finalisation of the representation is performed in section 4.8 and 4.9. Another key aspect which is to test the automation (DEA) capability of the formalised model using a series of steps for accuracy of the reasoner and query with the supporting tool. The results of section 4.8 and 4.9 will contribute to the testing mechanism for DEA.

4.3 Key concepts and relationships of the Process Model

Figure 4-3 illustrates all the high level, intermediate and low level concepts as F-B-S aspects of the process model for DEA as a Meta Model developed as part of this research. Inputs and outputs are adopted from the definition - entities consumed and modified during an activity with engineering rules controlling the behaviour as methods with conversion of inputs to outputs along with resources which may be a design tool or a physical resource (Ding et al., 2009). Engineering rules have been modelled for engineering design process knowledge as part of DEA to control the effect of design variations on product parameters (Bermell-García and Fan, 2002; Calkins et al., 2000). Product function and behaviour in context to engineering design process have been modelled as concepts and relationships for interoperable knowledge sharing using Core Product Model 2 (CPM2) and Open assembly Model (OAM) (Fenves, 2001; Fenves et al., 2008; Szykman et al., 2001, 2000a, 2000b). However, very important contributions of process adding semantics to product function and behaviour throughout the product lifecycle as part of engineering design process have been made by Frederic Noel (Noel, 2006) and John Gero (Gero and Kannengiesser, 2007a).

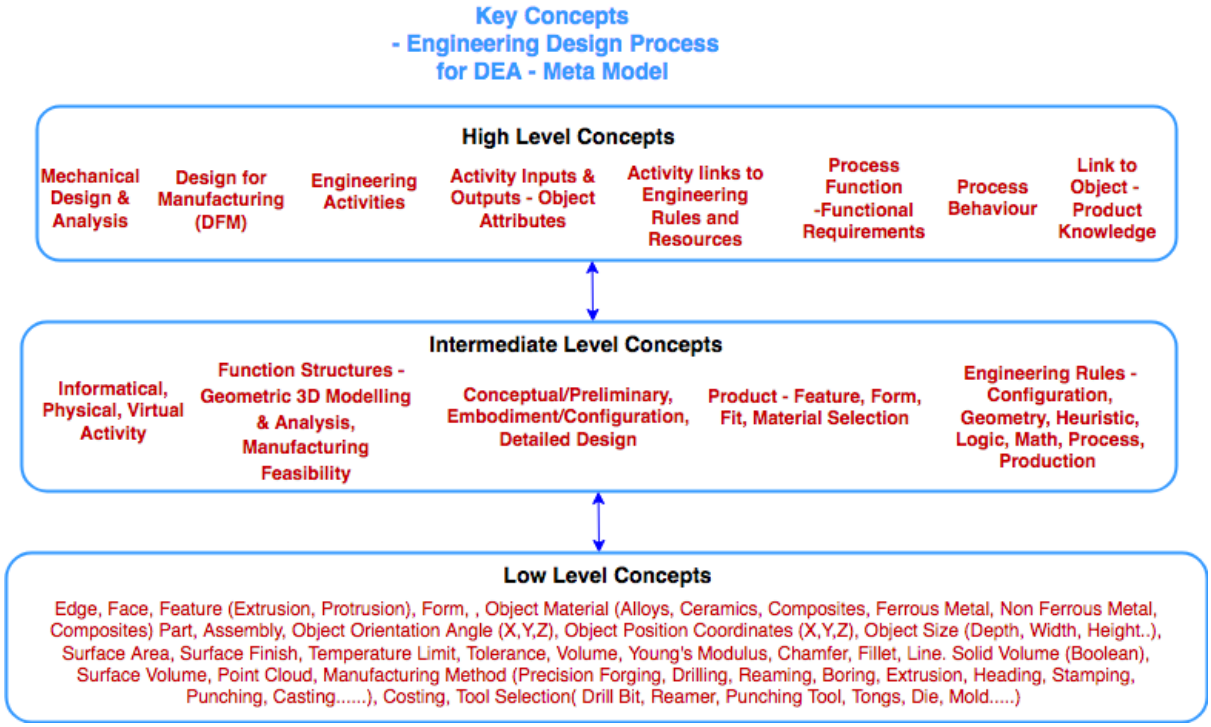


Figure 4-3: Concepts for the required Process Model for DEA – Meta Model

The high level concepts formulated by the author are described as follows –

- ***Process description with activities, inputs, outputs, resources and activity id***
- ***Process inputs & outputs as product geometric attributes***
- ***Engineering rules based on math and logic***
- ***Process functional requirement / function***
- ***Process behaviour***

Thus the following set of research questions arise -

I. How can the mechanical product design process with inclusion of manufacturing knowledge (DFM/DFA) based on various entities such as *activities, rules, logic, function and behaviour* for product realisation as per author's Meta model, be captured in *a generic and re-usable process model as a model driven approach with structured knowledge model for automation* in a virtual engineering environment?

II. How can the developed process model in line with author's Meta model be then *formally represented for machine interpretation in platform independent and neutral representation standards with semantic clarity* (clear meaning of concepts) for Design Engineering Automation (DEA) for mechanical design with DFM/DFA with a KBE approach through open standards?

A 3rd question also arises as a consequence of the 2nd question

III. Can the *formalised process model* enable automation with *generative modelling* from the *functional requirements* generated at the initiation of the design process as the design intent *with queries and reasoning on developed generic functions*?

As observed from the findings in section 3.8, PSL ontology can represent activity with inputs, outputs and object along with resources from the identified core concepts. Similarly, RuleML and RIF can both represent math and logic rules. OWL ontology can represent concepts and binary relationships. All of them have been extensively used in the engineering domain. The next section will discuss the experimentation of the high level concepts of the process model with languages such as PSL, RuleML and OWL to demonstrate the effectiveness of these formal representation languages for DEA in context to KBE.

4.4 Pilot Use Case 1 – Precision Forging of Aero Fan/Compressor Blades as Design for Manufacturing (DFM)

4.4.1 Preliminary Knowledge Analysis

An informal process has been devised for DFM of aero fan blades by the author as illustrated in Fig 4-2. The process is precision forging of compressor blades as method of manufacturing. The activities can be broken down mainly as – '*Extrusion*', '*Heading*' and '*Stamping*'. All the detailed knowledge cannot be shown here due to copyright issues as per the intellectual property rights of the industrial partner.

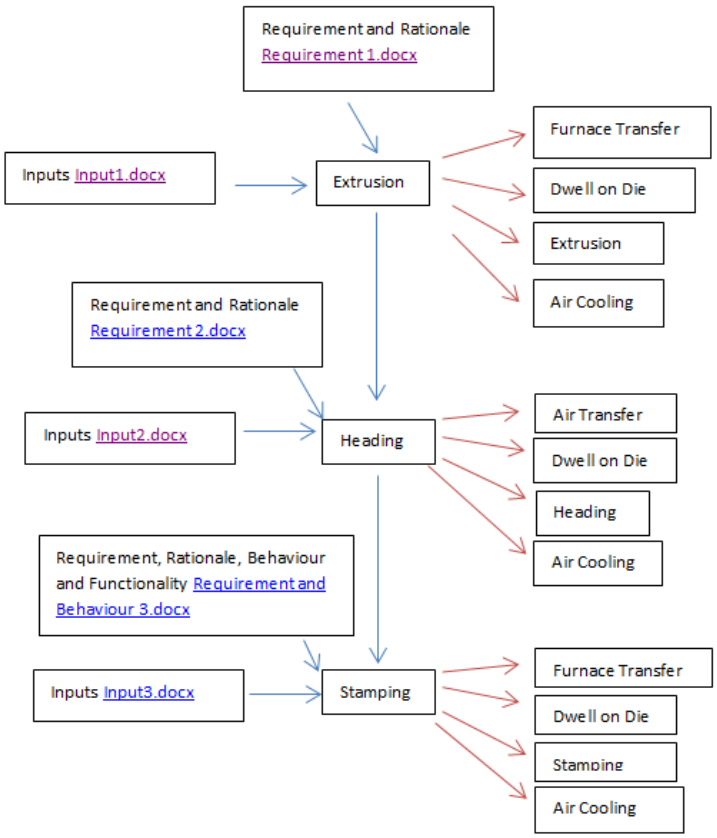


Figure 4-4: Use Case 1 - Example of a precision forging process of a compressor blade

As observed from Figure 4-4, the process map illustrates inputs, functional requirement, behaviour and rules along with sub-processes of major activities in the form of extrusion, heading and stamping. The inputs required for extrusion are material data – *billet & dies, geometries – billet (without glass coating), tongs, dies (Nominal) & punch, temperature to which the workpiece is heated up in the furnace prior to extrusion, furnace transfer duration, duration for which the workpiece rests on the die, die temperature, press characterisation and punch stopping position*.

4.4.2 Mapping of Informal Process Model Concepts to Formal Representation Standards: PSL, RuleML, SysML

The formal representation framework is based on the discussion for representing process information along with other design decomposition features such as rules, logic and requirements along with flow of information in the form of inputs and outputs.

4.4.2.1 Activities with Inputs & Outputs and Objects: Process Specification Language (PSL)

The PSL syntax illustrating the flow of information along with extensions to illustrate inputs and outputs is shown as follows. The inputs and sub-activities are only shown for the extrusion process in the PSL syntax using the core theories and extensions –

(define-parameter
:variable ?cb
:constraints (compressor blade ?cb))
(define-activity-role
:id s1
:name Extrusion
:successors 2
:preconditions (not extruded ?cb (beginof ?occ))
:postconditions (extruded stem of ?cb(endof ?occ)))
(define-activity-role)
:id s2
:name Heading
:successors 3
:preconditions (extruded stem of ?cb(beginof ?occ))
x:postconditions (headed shape of ?cb endof ?occ)))
(define-activity-role)
:id s3
:name Stamping
:successors 4
:preconditions (headed shape of ?cb (beginof ?occ))
:postconditions (stamped ?cb (endof ?occ)))
(forall (?s1 ?m ?g ?t1 ?t2 ?t3 ?t4 ?p1 ?p2)
(implies (= ?s1 extrusion(?m ?g ?t1 ?t2 ?t3 ?t4 ?p1 ?p2))
(and (activity_occurrence ?s1
(Material data – Billet & Dies ?m)
(Geometries – Billet (without glass coating), Tongs, Dies(Nominal) & Punch ?g)
(temperature to which the workpiece is heated up in the furnace prior to extrusion ?t1)
(Furnace Transfer Duration ?t2)
(Duration for which the workpiece rests on the die ?t3)
(Die Temperature ?t4)
(Press Characterisation ?p1)
(Punch stopping position ?p2))))
(forall (?cb ?s1)

(implies (or (occurrence-input ?cb ?s1)
(occurrence-output ?cb ?s1)
(and (object ?cb)
(not (state ?cb))
(activity_occurrence ?s1))))
(forall (?cb ?s1)
(iff (participant ?cb ?s1)
(exists (?t)
(participates_in ?cb ?s1 ?t)))
(forall (?cb ?s1)
(implies (or (occurrence-input ?cb ?s1)
(occurrence-output ?cb ?s1))
(participant ?cb ?s1)))
(exists (?s1 ?m ?g ?t1 ?t2 ?t3 ?t4 ?p1 ?p2)
(and (occurrence_of ?s1 Extrusion(?m ?g ?t1 ?t2 ?t3 ?t4 ?p1 ?p2)
(occurrence-input ?m ?g ?t1 ?t2 ?t3 ?t4 ?p1 ?p2)
(occurrence-output ?m ?g ?t1 ?t2 ?t3 ?t4 ?p1 ?p2)))
(forall (?cb ?s1 ?f)
(implies (or (input-state ?cb ?s1 ?f)
(output-state ?cb ?s1 ?f)
(and (object ?cb)
(not (state ?x)
(activity_occurrence ?s1)
(state ?f))))
(forall (?cb ?s1 ?f)
(implies (input-state ?cb ?s1 ?f)
(and (occurrence-input ?cb ?s1)
(prior ?f ?s1)
(exists_at ?cb (begin_of ?s1)))))
(forall (?cb ?s1 ?f)
(implies (output-state ?cb ?s1 ?f)
(and (occurrence-output ?cb ?s1)
(achieved ?f ?s1)
(exists_at ?cb (end_of ?s1))))
subactivity(Furnace Transfer, Extrusion)
subactivity(Dwell on Die, Extrusion)
subactivity(Extrusion, Extrusion)
subactivity(Air Cooling, Extrusion)

4.4.2.2 Engineering Rules: RuleML

The rule that governs the extrusion process as ExtrusionRule1 is - *A short extruded stem left behind after stamping would cause problems in handling the part while a long stem would result in excessive material use & high costs.*

ExtrusionRule1 is represented in RuleML(Boley et al., 2005) as follows -

<Implies>
<head>
<Atom>
<Rel>stamping</Rel>
<Ind>short extruded stem</Ind>
<Var>problems in handling</Var>
</Atom>
</head>
<body>
<And>
<Atom>
<Rel>stamping</Rel>
<Ind>long stem</Ind>
<Var>excessive material use</Var>
</Atom>
<Atom>
<Rel>stamping</Rel>></Rel>>
<Ind>long stem</Ind>
<Var>high costs</Var>
</Atom>
</And>
</body>
</Implies>

4.4.2.3 Functional Requirements: SysML Requirement Diagram

The functional requirement of the overall precision forging process is – '*achieve an accuracy of +/-2mm in shape prediction, as shape prediction accounts for a bulk of the manufacture objectives*'.The functional requirement for extrusion is - '*the objective of extrusion modelling is to ensure that the extruded stem is long enough for the part to be handled in subsequent operations. The base of the extruded part also needs to have enough material for subsequent*

heading, which can be estimated using its length'.The functional requirement of each activity is captured and represented in SysML requirement diagram as shown below. The underlying schema of the requirement as illustrated in the Figure 4-5 is the text of the requirement, identifier, source, kind, method, risk and status. The model can be exchanged via AP233 as well as discussed in the earlier section.

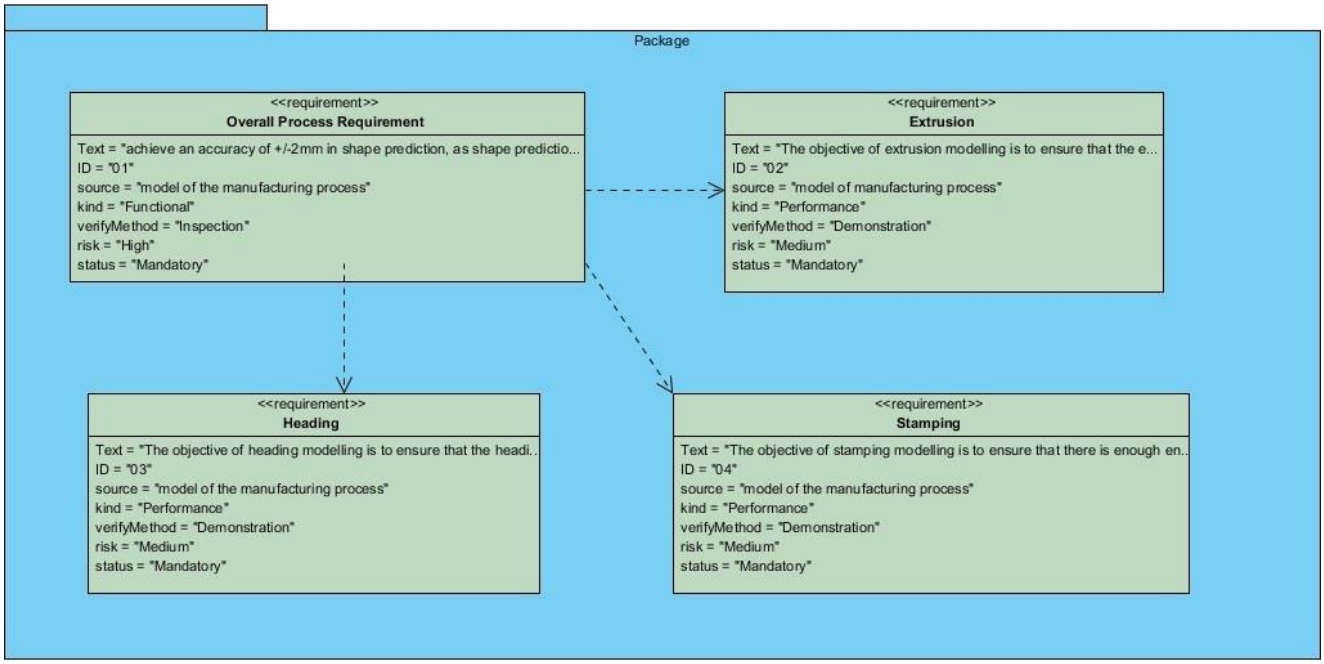


Figure 4-5: Use Case 1 - SysML Requirement Diagram for capturing functional and performance based requirements of the precision forging manufacturing method

4.5 Pilot Use Case 2 – Conceptual Design of Aero Fan Blades

4.5.1 Preliminary Knowledge Analysis

The case study discusses design aspects of fan blades (Amoo, 2013). An informal process capturing design aspects of fan blades has been derived and compiled by the author in a process map illustrating inputs, enablers, parameters, requirements, rules, logic, behavior, and attributes along with the object primarily defined as a blade (Amoo, 2013). The blade can be a fan blade, compressor blade or a turbine blade, which makes the process model generic for reusability. The process map captures all aspects of a process which IDEF4 captures but does

not demonstrate the IDEF4 syntax in the form of a static, dynamic, and behavioural models in the present shape. The activities or events are broken down into three basic steps:*blade geometryoptimisation*, *dovetail attachment* and *material selection* as shown in Fig. 4-6 along with other design decomposition features. Fig. 4-7 illustrates the object box for the blade.

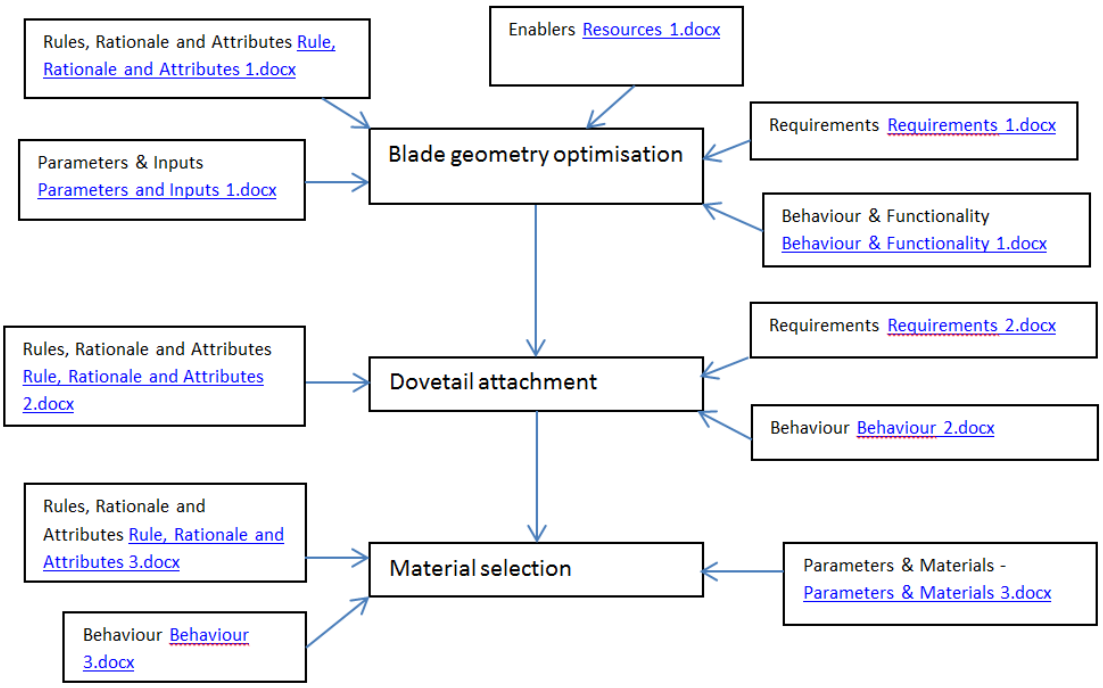


Figure 4-6: Use Case 2 - An informal process capturing design aspects of a fan blade

Blade	
Object	{O} fan blade
Attributes	{A} leading edge
	{A} trailing edge
Methods	{M} BEM Theory usage
Event	{E} geometry optimization
	{E} dovetail attachment
Relation	{R} compressor blade
Link	{L} design features

Figure 4-7: Use Case 2 - The object box as per IDEF4 methodology

4.5.2 Mapping of Informal Process Model Concepts to Formal Representation Standards: PSL, RuleML, SysML

The formal representation framework is based on the discussion for representing process information along with other design decomposition features such as rules, logic and requirements along with flow of information in the form of inputs and outputs.

4.5.2.1 Activities and Objects: Process Specification Language (PSL)

PSL activity role declaration (ARD) and object declaration syntax is explained as follows:

(define-activity-role
:id <*number*>*
:name <*string*>
:successors <*number*>*
:preconditions <PSL sentence>*
:postconditions <PSL sentence>*)
(define-object
:name <*KIF constant*>
:constraints <*PSL sentence*>*)
(define-parameter
:variable <*KIF variable*>
:constraints <*PSL sentence*>*) (Grüninger and Menzel, 2003)

The object declaration can be a constant as shown in the first object declaration or a variable as shown in the next object declaration. The PSL syntax illustrating the flow of information along with extensions to illustrate parameters along with inputs and outputs is shown as follows, but only for blade geometry optimisation.

(define-parameter
:variable ?fb
:constraints (fan blade ?fb))
(define-activity-role
:id s1
:name Blade geometry optimisation
:successors 2
:preconditions (existing design of ?fb(beginof ?occ))
:postconditions (preliminary optimal geometric design features of ?fb(endof ?occ)))

(define-activity-role)
:id s2
:name Dovetail attachment
:successors 3
:preconditions (preliminary optimal geometric design features of ?fb(beginof ?occ))
:postconditions (attached dovetail design features to the design of ?fb endof ?occ)))
(define-activity-role)
:id s3
:name Material selection
:successors 4
:preconditions (attached dovetail design features to the design of ?fb (beginof ?occ))
:postconditions (material allocated to the preliminary design of ?fb (endof ?occ)))

4.5.2.2 Activity Inputs and Outputs: PSL

The parameters and inputs for blade geometry optimisation are broken down informally as:

Parameters: incremental lift created by each blade, ideal power and proper airfoil section, twist, chord, and pitch angle for optimal thrust distribution. Inputs: aerodynamic forces acting on a local airfoil and global changes in momentum along with rate of air intake (Amoo, 2013).

The formal syntax in PSL incorporating extensions is as follows:

(forall (?s1 ?l ?p ?t ?fm ?r)
(implies (= ?s1 Blade geometry optimization(?l ?p ?t ?fm ?r))
(and (activity_occurrence ?s1
(Incremental Lift created by each blade ?l)
(Ideal power ?p)
(Proper airfoil section, twist, chord, and pitch angle for optimal thrust distribution ?t)
(Aerodynamic forces acting on a local airfoil and global changes in momentum ?fm)
(Rate of air intake ?r))))
(forall (?fb ?s1)
(implies (or (occurrence-input ?fb ?s1)
(occurrence-output ?fb ?s1)
(and (object ?fb)
(not (state ?fb))
(activity_occurrence ?s1)))))
(forall (?fb ?s1)
(iff (participant ?fb ?s1)
(exists (?t)

(participates_in ?fb ?s1 ?t))))
(forall (?fb ?s1)
(implies (or (occurrence-input ?fb ?s1)
(occurrence-output ?fb ?s1))
(participant ?fb ?s1)))
(exists (?s1 ?l ?p ?t ?fm ?r)
(and (occurrence_of ?s1 Blade geometry optimisation(?l ?p ?t ?fm ?r)
(occurrence-input ?fm ?r ?s1)
(occurrence-output ?fm ?r ?s1)))
(forall (?fb ?s1 ?f)
(implies (or (input-state ?fb ?s1 ?f)
(output-state ?fb ?s1 ?f)
(and (object ?fb)
(not (state ?fb)
(activity_occurrence ?s1)
(state ?f))))))
(forall (?fb ?s1 ?f)
(implies (input-state ?fb ?s1 ?f)
(and (occurrence-input ?fb ?s1)
(prior ?f ?s1)
(exists_at ?fb (begin_of ?s1)))))
(forall (?fb ?s1 ?f)
(implies (output-state ?fb ?s1 ?f)
(and (occurrence-output ?fb ?s1)
(achieved ?f ?s1)
(exists_at ?fb (end_of ?s1)))))

4.5.2.3 Engineering Rules: RuleML

A few examples of the design rules to be followed during the blade geometry optimisation process are represented in RuleML (Boley *et al*., 2005) as follows:

BladeGeometryOptimisationRule1:*a 30% hollowing in a hollow fan blade results in about a 13%–16% decrease in torsional rigidity compared to a solid blade design* (Amoo, 2013).

rule ml declaration (implication)

<Implies>
<head>
<Atom>
<Rel>hollowing</Rel>
<Ind>30%</Ind>

<Var>hollow fan blade</Var>
<Rel>compared to a solid blade design</Rel>
</Atom>
</head>
<body>
<Atom>
<Rel>decrease</Rel>
<Ind>13-16%</Ind>
<Var>torsional rigidity</Var>
</Atom>
</body>
</Implies>

BladeGeometryOptimisationRule2: *The rate of air intake varies and is dictated by factors such as airfoil geometry, angle of attack, air density, and the speed at which the airfoil moves through the air* (Amoo, 2013).

rule ml declaration (statement)

<Atom>
<Var>rate of air intake</Var>
<Rel>dictated by factors such as</Rel>
<Var>airfoil geometry</Var>
<Var>angle of attack</Var>
<Var>air density</Var>
<Var>speed at which the airfoil moves through the air</Var>
</Atom>

4.5.2.4 Functional Requirements: SysML Requirement Diagram

The functional requirement as derived from the process for blade geometry optimisation is that– '*the fan blades spin to accelerate a mass of air into the engine to generate thrust that propels the aircraft forward. Approximately 80% of the thrust produced by a modern jet engine is delivered by the fan'*. '*Fan blades also function to reduce total engine damage from the ingestion of various foreign objects such as birds by radially deflecting outward such objects rather than passing them through to the core parts of the engine*' (Amoo, 2013). The functional requirements of the process are captured and represented in a SysML requirement diagram as shown in Fig. 4-8. The underlying schema of the requirement as illustrated in the

figure is the textual requirement, identifier, source, kind, method, risk, and status. The model can be exchanged via AP233 of STEP.

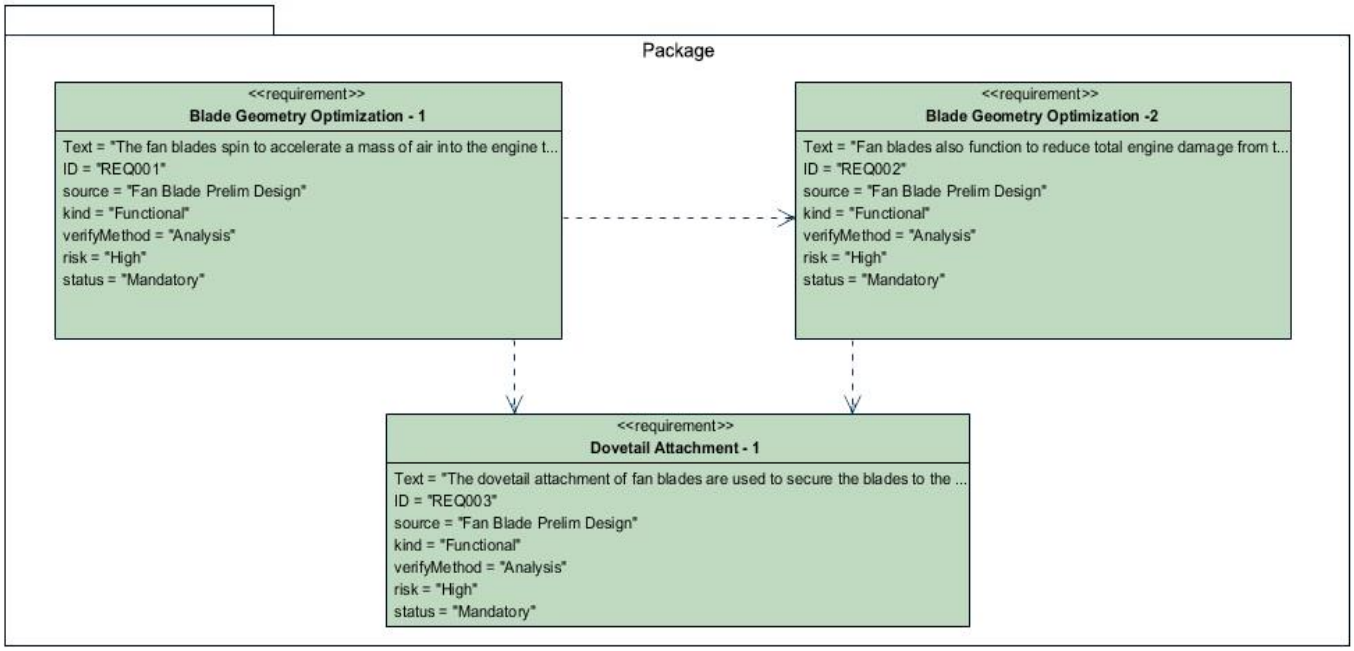


Figure 4-8: Use Case 2 - SysML requirement diagram for representing functional requirements of the design aspects of the fan blades process

4.5.3 Mapping of Informal Process Model Concepts to Formal Representation Standards: OWL

As part of this research, for the initial process model affecting the product at part level, text based instances were created for all classes using Use-case 1, 2 & 3 by the author. Initial naming convention for all instances follows the pattern – ProcessModel:ActivityNameClassNameNo. For example, the function of the fan blade exhibited during the activity BladeGeometryOptimisation – 'The fan blades spin to accelerate a mass of air into the engine to generate thrust that propels the aircraft forward. Approximately 80% of the thrust produced by a modern jet engine is delivered by the fan' is named as ProcessModel:BladeGeometryOptimisationFunctionalRequirement1.

BladeGeometryOptimisation is the activity name; FunctionalRequirement is the class name and no. is 1. The other function of the fan blade exhibited during the activity

BladeGeometryOptimisation – ‘Fan blades also function to reduce total engine damage from the ingestion of various foreign objects such as birds by radially deflecting outward such objects rather than passing them through to the core parts of the engine’ is named as ProcessModel:BladeGeometryOptimisationFunctionalRequirement2.

BladeGeometryOptimisation is the activity name; FunctionalRequirement is the class name and no. is 2.All instances will satisfy the class description and properties. SPARQL query is a method of querying the RDF graph (Composer, 2011). In Topbraid, in the SPARQL tab queries are run over the asserted triples in the ontology. In order to run the query over both the asserted and the inferred triples, inference needs to be executed on the model. For this SPIN rules through OWL 2 RL need to be activated. Only then, the inference window produces the inferred triples from the ontology model. Performing query in the SPARQL tab will now perform query over both the asserted triples and inferred triples in the standard edition of Topbraid. The syntax of SPARQL query is illustrated as follows –

```
SELECT *
WHERE {
        ?subject rdfs:subClassOf ?object .
}
```

SELECT * selects the complete ontology model, subject and object correspond to classes and individuals, rdfs:subClassOf is the predicate. Using the properties created in the model and putting them as predicate between classes and instances of the model, objects and subjects can be retrieved in the SPARQL query tab. Thus although concepts like rationale, function and behaviour of the process have presently been implemented as literals (datatype properties) in context to the design process, running a SPARQL query through the property returns the function and name of rules associated with the activity. For example,

```
SELECT *
WHERE {
        ProcessModel:BladeGeometryOptimisation
```

ProcessModel:exhibitsFunctionalRequirement ?object . }

Running this query yields the results as ProcessModel:BladeGeometryOptimisationFunctionalRequirement1 and ProcessModel:BladeGeometryOptimisationFunctionalRequirement2 as the objects. Clicking on these specified instances yields other linked properties such as ProcessModel:hasmethodasBehaviour and the stated functional requirement as a literal with the help of datatype property in the form of ProcessModel:isdescribedby. The domain of the property ProcessModel:isdescribedby is defined as FunctionalRequirement class and the range as a string. The illustration for this query is shown in the Figure 4-9.

Another example of SPARQL query -

```
SELECT *
WHERE {
      ProcessModel:BladeGeometryOptimisation ProcessModel:followsRule ?object .
}
```

ProcessModel:BladeGeometryOptimisation is the subject in the query, ProcessModel:followsRule is the property or the predicate in the query and object needs to be returned. Running the above query yields the results as ProcessModel:BladeGeometryOptimisationRule1 and ProcessModel:BladeGeometryOptimisationRule2 as objects through the property ProcessModel:followsRule linking the domain and the range.

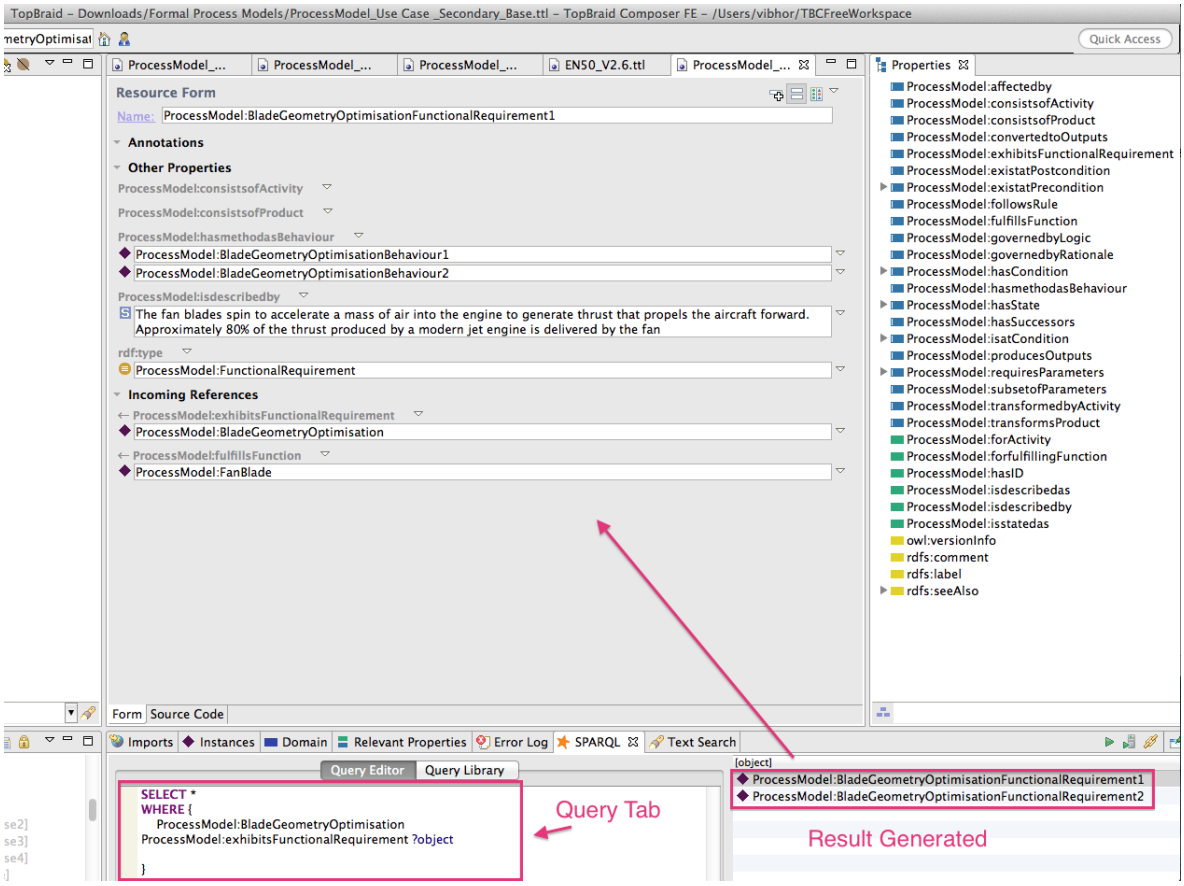


Figure 4-9: SPARQL Query Illustration: Activity and Functional Requirement

As illustrated, the limitation of SPARQL query is that it only infers the name of the rules as text. To formally represent rules, they need to be embedded in the model with rule language such as RuleML, RIF or SWRL formalism on top of OWL2 as shown in section 4.4.1.2.3. For this reason along with the requirement of the process model to affect product attributes at the highest level of granularity for DEA, use case 4 and use case 5 have been used to refine and validate the process model. Use cases 4 and 5 will be discussed in Chapter 5 in detail. Use case 4 involves making a hole in the block with drilling as the manufacturing process. Use case 5 includes design of bookshelf implemented in AML as a KBE tool and Siemens NX, CATIA Knowledgeware as parametric CAD systems from MOKA ICARE forms.

4.6 Findings and Analysis – Pilot Use Case Experimentation

With the help of Use Case 1 & 2, it has been illustrated that PSL enables process representation with parameters, inputs, and outputs using core theory and extensions in the ontology(Bock and Gruninger, 2004). It needs more experimentation to illustrate representation of other design decomposition features such as constraints and attributes. RuleML (Boley *et al*., 2005) can be implemented to exhibit for textual rules. Similarly SysML can exhibit requirements. The formal representation framework will need integration for simultaneous application. PSL can be directly mapped to UML and hence to the SysML requirement diagram. RuleML and PSL can be integrated and shared via XML schemas. Similarly, all formats and languages to be experimented for representing other design decomposition features will need integration.

4.7 Requirements for a process model for implementation in neutral formal representation enabling design engineering automation (DEA)

The author has compiled the requirements for a generic process model enabling DEA through neutral formal representation. The requirements are an amalgamation of 2 sets –1) requirements for a process model to capture mechanical design domain concepts and relationships in a unified and integrated model and 2) requirements of the knowledge representation (KR) or knowledgebase (formal representation of process model) for DEA.

Some of the requirements for KBE methodologies enabling automation can be classified as flexibility, extensibility, scalability and integration (Colledani et al., 2008). This means that the process model must be generic and widely applicable to various product design systems, must be extensible to add both product and process knowledge, and provide all relationships as interdependencies. Thus some of the key characteristics, which can be deduced as requirements for process model for mapping to formal representation model, are

encapsulation of concepts and relationships with high level of knowledge abstraction for re-use. Even as per KNOMAD methodology enabling DEA, the knowledge model needs to follows these steps although the implementation can vary. These steps as summarised by Pinto and Martins (Pinto and Martins, 2004) can be stated as

I. Specification – identification of scope of knowledge model
II. Conceptualisation – identification of domain concepts and relationships
III. Formalisation – organising domain concepts in class hierarchies and completion of axioms to formally model relationships
IV. Implementation – codification of class hierarchies and axioms in a suitable formal knowledge representation language
V. Maintenance - updating and maintenance of the implemented knowledge model

These steps are critical even though the representations may vary. Some representations have been implemented as ontologies by Noy & McGuiness as seven-step method (Noy and McGuinness, 2001), METHONTOLOGY (Fernández-López et al., 1997) and six-stage methodology (Ahmed et al., 2007).

4.7.1 Requirements for an unified / integrated process model ready for implementation as formal representation to enable DEA in context of KBE

Thus, some of the key requirements have been deduced by the author for the process model ready for implementation as formal representation to enable DEA in context of KBE.

I. ***Modelling of the Meta model based on domain concepts and relationships of the process model*** - This includes design process activities, activity inputs & outputs as product geometric attributes, resources, engineering rules based on logic as well as mathematics in relation to the design process as well as product geometric attributes for change of product's state. This also includes interdependencies between sub-functions

corresponding to activities, products and the design process functional requirements and process behaviour. It should also be able to represent an interface between the process model and aspects of product model such as assembly & part structures, feature, form and fit

II. ***The modularity of engineering design intent with concise classification of concepts and relationships and the ability to instantiate*** - This will enable high re-usability of the developed process model as high abstraction

III. ***Suitable axioms for constraining the domain concepts and relationships in a suitable formal representation language for execution in the form of reasoning and querying*** - It should be ensured that there is optimal syntactic and semantic mapping of the informal/semiformal model to formal model. This was stated as computational capability in section 3.2

The requirements have been jointly formulated and compiled as per these sources (Colledani et al., 2008; Danjou et al., 2008; Frank et al., 2014; Klein et al., 2015, 2014; J Kulon et al., 2006; J. Kulon et al., 2006; Lützenberger et al., 2012; Nomaguchi et al., 2002; Pinto and Martins, 2004; Rezayat, 2000; Ríos et al., 2005; Tomiyama et al., 2002).

Thus the next steps are to formulate requirements for axiom selection and formal representation enabling DEA.

4.7.2 Requirements for a knowledge representation system (knowledge base) enabling DEA

A knowledge representation system (KRS) generally consists of a knowledge representation (KR) formalism with well-defined syntax and additionally if possible, semantics preserved, as symbolically encoded knowledge (Shehab and Abdalla, 2002, 2006). The symbolically encoded knowledge is crucial in making the knowledge representation layer machine readable or computer readable (Patil, 2005). The symbolically encoded knowledge of the

engineering design process model domain concepts and relationships refers to formal representation in the context of this thesis. It also contains an inference engine, which means reasoning mechanism closely built in with the representation layer or language for deducing queries and consistency checking ensuring the representation is dynamic in nature as opposed to just a static representation (Davis et al., 1993; Johansson, 2011; Tomiyama et al., 2002). It may also contain a front-end environment for visualisation and possibly knowledge editing and debugging (Bullinaria, 2005; Clark, 1996; La Rocca, 2011). The front-end environment for visualisation is out of scope for this research.

The compiled requirements by the author for KRS enabling DEA in context of KBE can be classified as follows -

I. ***Expressiveness*** – it means the expressive capability of the language to exhibit domain knowledge of all classifications. In this case, it means representation of engineering design knowledge as a unified process model with class-subclass relationship, properties, logical rules, mathematical rules and functional knowledge

II. ***Inference adequacy and efficiency as execution of its code*** – the formal representation system or KRS should be able to perform inference as reasoning and queries as execution of its code with minimum degree of incompleteness. The system should enable maximum time and memory efficiency while performing the inference or execution of its code so that it returns the answer to the user in reasonable amount of time along with correctness of the answer. This is the layer that adds dynamic nature to the static representation

III. ***Explanation for inference*** – ideally, along with an inference, a system representation should also be able to tell the reason for selecting an answer through inference

IV. ***Semantic clarity*** – additionally if possible, the language should offer well defined semantics or meaning of terms through its axioms

V. ***Acquisition efficiency*** - the efficiency and naturalness of input of domain knowledge by the knowledge engineer. This indicates syntactic friendliness of the representation or knowledge base along with graphical display and convenience. It supports structured and modular knowledgebase

VI. ***Feedback during knowledge input*** - the system should not be static. It should be dynamic and warn of inconsistencies if the axioms entered are incorrect. This is tangible to consistency checking paradigm

VII. ***Extensibility and scalability*** - the system should offer ease of adding new information to the existing knowledge base. As the size of the knowledge base increases, the system performance should still function within a reasonable time and performance shouldn't degrade quickly

VIII. ***System Interface to external applications*** – the system should atleast provide mechanism to link to other database or application (DEA application including KBE application in this case). The linkage to an external system won't be addresses as part of this thesis

IX. ***Robustness, portability and ease of integration*** – the system should offer least bugs as possible or no bugs at all in an ideal situation. The system should not be too difficult while transferring its representation to other platforms. In this case, the neutral representation should enable open standard usage.

The requirements have been jointly formulated and compiled as per these sources (Bullinaria, 2005; Clark, 1996; Colledani et al., 2008; Davis et al., 1993; Frank et al., 2014; Johansson, 2015, 2011, 2008; La Rocca, 2011; Ríos et al., 2005; Rocca, 2012; TechnoSoft Inc, 2003; Tomiyama et al., 2002; Tomiyama and Hew, 2000; Tor et al., 2008; Van Der Velden et al., 2012; Van der Velden, 2008)

4.7.3 Compiled requirements for a process model for implementation in neutral formal representation enabling DEA in context of KBE

Combining both the sets of requirements in a concise manner yields the requirements for a process model to enable DEA through neutral formal representation in context of KBE -

I. ***Expressive capability of language to exhibit domain knowledge for the mechanical design process in the form of Meta model based on domain concepts and relationships of process model***- This should include all entities such as design process activities with inputs & outputs as product's geometric attributes, resources, engineering rules based on logic and math, function, behaviour and interface with product model as identified in section 4.3 in a unified and integrated approach. It should support the class-subclass relationship between the concepts and represent all relationships. Product parameters as object inputs are crucial in product design process and thus form a critical part of the design process knowledge at the detailed design stage.

II. ***Inference (reasoning) and querying with optimum adequacy and efficiency as execution of its code*** – the system should allow for deduction of new information from static domain knowledge through inference making the system dynamic in nature. It should perform reasoning or execution of its code with optimum performance between time and memory efficiency and degree of completeness. If possible, the representation should support consistency checking.

III. ***Semantic clarity*** – additionally if possible, the axioms of the language should constrain the interpretation of domain concepts and relationships

IV. ***Modularity in the knowledge representation system (KRS) with precise axioms for domain concepts and relationships*** - This will ensure structuring of the knowledgebase along with the ability to instantiate enabling high re-usability

V. ***Extensibility and scalability*** - optimum system performance in accordance with addition of new domain knowledge in the form of concepts, relations and instances of the engineering design process

VI. ***Neutral representation*** - adopted from robustness and portability. Pertaining to this research, the formal representation should enable open standard usage.

4.8 Basic Comparison of Formal Representation Standards

A brief comparison of formal representation standards is explained before detailed comparison for implementation of all aspects of the process model in neutral formal representation.

4.8.1 STEP vs. Ontology Based Approach

STEP based on EXPRESS (ISO 10303-11) (ISO, 2004)provides a schema for product data model throughout its lifecycle (Zhao and Liu, 2008a). However, it differs from ontology-based approach in various ways. All ontological languages formalised over various logic which may be OWL based on DL or PSL based on FOL support automated reasoning (Hay, 2006; NIST, 2008; W3C, 2012). They can deduce new knowledge from the existing knowledgebase with the help of an inference engine, making the representation dynamic in nature. STEP based on EXPRESS schema is static in nature as it can't execute (Dong et al., 1997; Tang et al., 2001) and doesn't possess reasoning capability (Qin et al., 2017). However, the procedural knowledge contained inside class descriptions of EXPRESS can be extended and merged with external systems and even programming languages such as Java/C++ to enable execution of the statements (Zha and Du, 2002; Zhao and Liu, 2008a). The execution of the procedural knowledge inside the EXPRESS schema with the help of externally integrated systems will make the representation dynamic in nature.

4.8.2 UML/SysML vs. OPM

UML and SysML allow visual representation of an engineering system through multiple diagrams whereas OPM allows visual representation of an engineering system through an integrated approach in the form of OPD and OPL which is an added advantage (Reinhartz-Berger and Dori, 2004). Thus OPM is more favourable for modelling engineering systems at a higher level such as the conceptual or class and schema level as it doesn't model the individual activities of a process. Although, OPM goes to various levels of abstraction to represent the complete F-B-S of a system, it provides very less relation between the individual activities of a process and its implementation as formal representation (Subahi, 2015). If the abstraction of knowledge is required at product attribute level, UML class diagram or SysML Block diagram (Graves, 2009) are more comprehensive in expressing product model with all its geometric attributes. OPCAT as a tool for OPM allows direct export of XML information from the OPD and OPL, which form a machine-readable formal representation. However, a shortcoming of XML based representation is that it only covers the syntax level and doesn't impose any constraints on the semantics, hence is open to interpretation, and also doesn't provide support for reasoning (Antoniou and Van Harmelen, 2004; Ray and Jones, 2006; Yahia et al., 2012). In order to overcome this shortcoming and address DEA, formal representation framework beneath the graphical representation will need to preserve semantic clarity and allow reasoning or inference capability.

4.8.3 Ontology vs. Systems modelling approach as UML/SysML and OPM

Ontology is different from object oriented (O-O) modelling such as UML and object process methodology (OPM) in various ways. One of the most crucial differences is ontology modelling is based on logic (Siricharoen, 2007) and allows for automated reasoning or inference resulting in generation of new knowledge which are not supported by either languages such as UML/SysML and OPM. According to (Graves, 2009; Graves and

Horrocks, 2008), current systems modelling approaches such as OPM, UML/SysML are unable to provide formal semantics to the knowledge expressed and represented. UML is good for graphical display of the ontology but without the logic layer (Zhu et al., 2009). This is the reason UML can be considered as semiformal representation or light weight formalism (Chungoora et al., 2013a). Another difference is the built in of properties in ontologies which are marked at the same level as classes which means object properties can be defined between classes. O-O modelling limits the relationship between classes to superclass-subclass relationship (Siricharoen, 2007). Ontology modelling also adds relationships to properties in the form of symmetric, inverse and transitive, which can be accessed in the reasoning as against O-O modelling which doesn't support these features. Ontology modelling supports multiple inheritance exhibiting complex relationships whereas O-O modelling such as UML/SysML only allow for single inheritance. Ontology modelling also provides restrictions for class definition in the form of allvaluesfrom, somevaluesfrom (Zhu et al., 2009). Thus in spite of various differences in the underlying philosophy of UML/SysML and OPM with OPM focussing on object and process as kinds instead of UML/SysML on objects/blocks, both don't support logic for ontology and relations.

As OWL ontology provides formal semantics to the knowledge represented, it can act as semantic integration standard (Graves and Horrocks, 2008). Ontologies are thus good for defining metadata and providing semantic clarity and can be used as a basis of knowledge representation (KR) or defining metadata for building software and system engineering applications. Pertaining to this research, ontology encoded in OWL2 can be used as a backbone (semantic metadata) for DEA applications. As compared to O-O modelling techniques such as UML/SysML and even OPM for modelling systems, ontology modelling provides better support for exchange of knowledge across heterogeneous multiple platforms

by offering additional semantic clarity through reasoning mechanisms (Zhu et al., 2009). Figure 4-10 illustrates the classification of formal logic for knowledge representation.

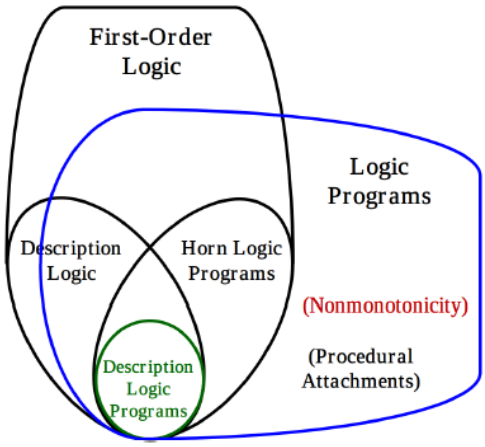


Figure 4-10: Formal Logic for Knowledge Representation (Grosof et al., 2010)

4.9 Comparative Analysis of Formal Representation Standards

The comparative analysis of the above mentioned formal representation standards as per the requirements for a process model enabling DEA, performed as part of this research is shown in Table 4-1.

Table4-1: Comparative Analysis of Formal Representation Standards

Classification of formal representation standards and languages	Requirements for a generic process model for DEA in context of KBE						References
	Expressive capability to represent all concepts and relationships	**Inference (automated reasoning), querying as execution of its code**	**Semantic clarity**	**Modularity with instantiation for high re-usability**	**Extensibility and Scalability**	**Neutral Representation (Open standards)**	
Semi-formal and graphical (non-logic based) without reasoning e.g. UML/SysML, OPM	Can represent most of the concepts and relationships including product and process knowledge with inputs, outputs and resources efficiently but not complete domain knowledge as a unified and granular process model	Don't support inference. However, the models can be executed with the help of programming languages such as Java/C++	Yes	Yes	Yes	Available open source tools such as visual paradigm, OPCAT provide neutral representation	(Blekhman and Dori, 2013; Dori, 2004, 2002; Foufou et al., 2005; Graves, 2009; Grobshtein and Dori, 2011; Hart, 2015; Mordecai et al., 2016; Sircharoen, 2007; Vanderperren et al., 2008; Weilkiens, 2007)
Formal and graphical (non-logic based) with reasoning e.g. frames, semantic networks	Represent some concepts and relationships but not all concepts and relationships as complete domain knowledge of the process model	Support inference	Not explicit	Yes, but lack of contextual relevance	Yes	Only open source tools may provide neutral representation	(Davis et al., 1993; La Rocca, 2011; Minsky et al., 1975; Obitko, 2007a, 2007b; Prasad, 2006; Robin, 2013; Sowa, 2015, 2008a; Wang et al., 2006)
Schema based representation in the form of STEP (EXPRESS Schema)	Represent some concepts and relationships such as product model with extremely high efficiency but not all concepts and relationships as complete domain knowledge of the process model	Don't support inference. However, the EXPRESS schema can be integrated with external systems and programming languages such as Java/C++ for execution of statements as inference	Not explicit	Yes, but lack of contextual relevance	Yes	EXPRESS schemas are available as neutral representation	(Barbau et al., 2012; Chandrasegaran et al., 2013; Dong et al., 1997; Krima et al., 2009; Lu et al., 2016; Lützenberger et al., 2012; Peak et al., 2004; Pratt, 2001; Qin et al., 2017; Sarigecili et al., 2014; Tang et al., 2001; Zhao and Liu, 2008b)
Schema based representation in the form of RDF/RDFS	Represent some concepts and relationships but not complete	Support inference and query in the form of SPARQL	Yes	Yes, but lack of contextual relevance	Yes	Open source tools such as Protégé, Topbraid provide neutral representation	(Beckett and McBride, 2004; Bruijn and Welty, 2013; Dean et al., 2004; Hay,

	domain knowledge of the process model						2006; Klyne et al., 2004; Manola et al., 2004; Mcguinness and Van Harmelen, 2004)
Formal logic based languages in the form of KIF, CG's, CLIF	Represent most concepts and relationships but not complete domain knowledge of a unified and granular process model	Support inference with logic based theorem provers	Yes	Yes	Yes	Only open source tools supporting logic paradigm may provide neutral representation	(Genesereth et al., 1992; Gruninger et al., 2013; Hayes and Menzel, 2001; Knutilla et al., 1998; Obitko, 2007f, 2007g, 2007h; Schlenoff et al., 2000a; Sowa, 2008b, 2011, 2008a)
Ontology languages based on formal logic such as OWL, process ontology as PSL and non-formal logic based such as Gellish	Individual language such as OWL2 and Gellish can represent most of the concepts but not rules, PSL can represent process specification with inputs, outputs, parameters but not complete domain knowledge as a unified and granular process model	Support inference or execution and querying with logic based reasoners for OWL2 and PSL	Yes	Yes	Yes	Open source tools supporting logic paradigm provide neutral representation for e.g. Protégé /Topbraid for OWL2	(Bechhofer, 2009; Bock and Gruninger, 2005, 2004; Chungoora et al., 2013a; Grüninger, 2009; Grüninger and Menzel, 2003; Hay, 2006; Hennig et al., 2016; Mcguinness and Van Harmelen, 2004; NIST, 2008; Obitko, 2007e; Pereira et al., 2011; Pouchard et al., 2000, 2005; Siricharoen, 2007; Van Renssen, 2003, 2005, Wang et al., 2006, 2004)
Rule languages based on formal logic such as RuleML, RIF and production rules	Rule languages can represent logical rules and basic mathematical rules but need to be linked to other logic based representations for complete domain knowledge as a unified and granular process model	Support inference and querying with logic based reasoners	Yes	Yes	Yes	Open source tools supporting logic paradigm provide neutral representation	(Boley et al., 2016a, 2016b, 2005; Davis et al., 1993; Feigenbaum et al., 2013; Hirtle et al., 2006; Kifer and Boley, 2010; La Rocca, 2011; Morgenstern et al., 2012; Pugliese and Colombo, 2014)
Object Oriented and multi-	Can represent all concepts and	Support dynamic inference or	Yes, but proper and	Yes	Yes	Original scripts available as languages, not	(Bermell-Garcia, 2007; Bermell-García

paradigm dynamic programming language in the form of LISP and LISP dialects such as Common Lisp (KBE systems are based on proprietary LISP dialects)	relationships of the domain knowledge as a unified and granular process model	execution by adding new code, object definitions at runtime. Support querying through methods	efficient execution required for precise semantics			as neutral representation standards. Automation applications developed are not available as neutral representation	and Fan, 2002; Cooper and LaRocca, 2007; Evenson et al., 2015; Foderaro, 1991; Kaufmann and Moore, 1997; La Rocca, 2011; La Rocca and Van Tooren, 2010; Lassila, 1990; Lützenberger et al., 2012; Preston et al., 2004; Rocca, 2012; P Sainter et al., 2000)
Object Oriented programming based languages in the form of Java, C/C++, Smalltalk, Ruby, Python, Fortran	Can represent all concepts and relationships of the domain knowledge as a unified and granular process model	Support inference or execution but not as dynamic as LISP, as they don't add new code, object definitions at runtime. Support querying through methods	Not explicit	Yes	Yes	Java scripts are available as cross-platform language, not as neutral representation standards; for C/C++ explicit codes need to be specified to enable cross-platform usage but still not as neutral representation standard. Automation applications developed are not available as neutral representation	(Barkmeyer et al., 2003; Bermell-Garcia, 2007; Goldberg and Robson, 1983; La Rocca, 2011; La Rocca and van Tooren, 2007; Reilly, 2006; Schlenoff et al., 2000a; TechnoSoft Inc, 2003; Toussaint and Cheng, 2002; Zeng et al., 2003; Zhao and Liu, 2008a)

4.9.1 Results and Discussion

As per the results of the Table 4-1, logic based languages seem to be the appropriate standards. O-O languages specially LISP oriented and combined with other O-O languages satisfy every criterion for DEA but not in open standards. In the context of this thesis, open standards enable neutral representation with semantics preserved (Peak et al., 2004; Usman et al., 2011).

As an open standard logic based language, CLIF, as ISO 24707 is extremely powerful knowledge representation paradigm with automated theorem prover. However several errors

were identified in 1st edition of CLIF as ISO 24707 (Gruninger et al., 2013) where the authors recommend the development of 2nd edition of CLIF as ISO 24707. The 2nd edition of CLIF (ISO, 2017) is under development in the present stage as compared to the 1st edition (ISO, 2007) published in 2007.Thus pertaining to open standard logic based language framework, integration of multiple languages is required from the observations in Table 4-1.

OWL is an extremely powerful semantic mediator for integration of concepts of the domain knowledge with contextual reference to be represented formally (Danjou et al., 2008; Graves and Horrocks, 2008). The formal DL logic as basis of OWL provides open standard usage enabling interoperability as compared to bespoke platform specific automation (Alexandrou et al., 2013).

Similar to OWL, the limitation of Gellish in context to the needs of DEA is representation and codification of engineering rules as multiple ary predicates. For inclusion of engineering rules, executable languages such as RuleML and RIF have been experimented with use-case examples.

PSL is the most comprehensive language for representing manufacturing knowledge with preserved semantics (Cochrane et al., 2009; Zhan et al., 2010). The execution of PSL can be achieved by either implementing them as methods in an O-O language such as Java/C++ (Cochrane et al., 2009) or with a theorem prover as inference or reasoning (Bock, 2006; Bock and Gruninger, 2005; Das et al., 2007). As experimented with Use Case 1, 2 and 3, PSL is very capable to represent process specifications with activity inputs, outputs as parameters (Bock and Gruninger, 2004). It is limited in representing product's geometric attributes with duration along with the actual state of the object in activity occurrence. However, it has been illustrated that along with binary change of state, change of product's geometric attributes as input and output states in activity occurrences can be achieved in PSL. However, it was identified to incorporate additional non-PSL based axioms in order to fully represent

relationship of product geometric attributes with specific integer and float values as units in context to DFM for design systems (Cochrane et al., 2009).It was also identified that a more detailed knowledge based system validation methodology is still required for development of design support systems with inclusion of manufacturing knowledge as DFM (Cochrane et al., 2009). The inclusion of change of product's geometric attributes as states linked to all design activities including the boolean operations on solid product profiles is extremely crucial for representing a process model for DEA specifically at detailed design stages.

In order to address the needs of DEA, formal representation of process function and behaviour are also very crucial as they form an integral part of the engineering design process specially the early stages such as preliminary and conceptual design. It was found out that the inclusion of engineering process function, behaviour and rationale in context to product model attributes has not been integrated in PSL with extensions (Zhan et al., 2010).PSL is still limited to define objects and concepts needed for finer details for DEA (Niles and Pease, 2001; Schlenoff et al., 2000b). Although it can represent some aspects object model knowledge such as form and features in terms of activity flow, it is restricted in representation of inputs and outputs of the process in terms of detailed object model knowledge such as form, fit and features (Young et al., 2007).

An important factor for DEA is the consideration of the equivalent representation of both virtual process for design and the corresponding physical process of manufacturing. The representation of the semantics of virtual process is very crucial for representing the design process for example; representation of removal of material in the form of hole is an boolean subtraction activity (extrusion or pocket) as virtual process and different forms of manufacturing methods such as drilling, reaming, boring as physical process. Similarly, representation of addition of material is a boolean addition activity (protrusion) as virtual process and different forms of manufacturing methods such as welding, joining or advanced

methods such as additive manufacturing as physical process. PSL has high representation capability of a physical process of design in terms of neutral formal representation of extremely complex manufacturing process with preserved semantics through neutral standards (Qiao et al., 2011). However, in its present state, PSL does not fully allow the representation of the equivalent virtual process with preserved semantics through its axioms.

4.9.2 Comparison of Neutral Formal Representation Standards for Mapping of Key Concepts and Relationships

Table 4-2, compiled as part of this research, will yield the complete framework of individual neutral representation standards that will represent the syntactic and semantic mapping of the identified key concepts and relationships as F-B-S aspects of an informal/semiformal process model as a formal model that intends to achieve DEA by performing execution of its code as inference and querying on axioms, similar to a DEA system or a KBES (KBE system).

Table4-2: Mapping of Identified Concepts and Relationships to Neutral Formal Representation Standards

Neutral formal representation standards and languages	**Concepts and relationships for automation in context to DEA system functionality as executable representation**						References
	Process description with activities, inputs, outputs, resources and activity id	**Process inputs & outputs as product geometric attributes**	**Engineering rules based on math**	**Engineering rules based on logic**	**Process functional requirement / function**	**Process behaviour**	
PSL	Can represent activity inputs, outputs and resources as parameters as well as activity id	Can represent activity occurrence inputs and outputs as product's geometric attributes to some extent. Need extensions for full representation and validation	Can represent manufacturing flow and sequencing operations as process rules. However, cannot comprehensiv ely represent rules with process and product knowledge with variable geometric attributes and nesting of math conditions	Can represent manufacturing flow and sequencing operations as process rules. However, cannot comprehensiv ely represent rules with process and product knowledge with variable geometric attributes and nesting of logic conditions	No, cannot represent design process function with respect to product	Can represent behaviour of manufacturing process models. However, cannot represent complete design process behaviour with respect to product attributes	(Bock, 2006; Bock and Gruninger, 2005, 2004; Chungoora and Young, 2011; Cochrane et al., 2009; Das et al., 2007; Usman et al., 2013; Young et al., 2007; Zhan et al., 2010)
OWL	Can represent activity with inputs, outputs, resources and activity id if a structured methodology is provided	Can represent activity inputs and outputs as product's geometric attributes if a structured methodology is provided	No, cannot represent engineering rules such as design and manufacturing rules based on math	No, cannot represent engineering rules such as design and manufacturing rules based on logic	Can represent design process function with respect to product if structured methodology is provided	Can represent design process behaviour with respect to product if structured methodology is provided	(Golbreich et al., 2012; Graves and Horrocks, 2008; Horridge and Patel-Schneider, 2012; Mcguinness and Van Harmelen, 2004; Motik et al., 2012; Siricharoen, 2007; W3C, 2012; Wang et al., 2006, 2004)
Rule ML	No, cannot represent taxonomic relations with activity inputs, outputs, resources and activity id	No, cannot represent activity inputs, outputs with product's geometric attributes	Yes, can represent engineering rules such as design and manufacturing rules with basic math built-ins	Yes, can represent engineering rules such as design and manufacturing rules with horn logic	No, cannot represent process function with respect to product	No, cannot represent process behaviour with respect to product	(Ball et al., 2005; Boley et al., 2016a, 2016b, 2016c, 2005; Golbreich, 2004; Hirtle et al., 2006)
RIF	No, cannot represent taxonomic	No, cannot represent activity	Yes, can represent engineering	Yes, can represent engineering	No, cannot represent process	No, cannot represent process	(Boley and Kifer, 2013; Feigenbaum

	relations with activity inputs, outputs, resources and activity id	inputs, outputs with product's geometric attributes	rules such as design and manufacturing rules with basic math built-ins	rules such as design and manufacturing rules with horn logic	function with respect to product	behaviour with respect to product	et al., 2013; Kifer and Boley, 2010; Morgenstern et al., 2012)
OWL / SWRL (OWL DL+ Unary /Binary Datalog RuleML)	Can represent activity with inputs, outputs, resources and activity id if a structured methodology is provided	Can represent activity inputs and outputs as product's geometric attributes if a structured methodology is provided	Yes, can represent engineering rules such as design and manufacturing rules with basic math built-ins	Yes, can represent engineering rules such as design and manufacturing rules with horn logic	Can represent design process function with respect to product if structured methodology is provided	Can represent design process behaviour with respect to product if structured methodology is provided	(Glimm et al., 2009; Golbreich and Imai, 2004; Horrocks et al., 2004; Kuba, 2012; Noh and Suh, 2008; Qin et al., 2016; Sarigecili et al., 2014; Tessier and Wang, 2013)
Gellish	Can represent activity with inputs, outputs, resources and activity id if a structured methodology is provided	Can represent activity inputs and outputs as product's geometric attributes if a structured methodology is provided	No, cannot represent engineering rules such as design and manufacturing rules based on math	No, cannot represent engineering rules such as design and manufacturing rules based on logic	Can represent design process function with respect to product if structured methodology is provided	Can represent design process behaviour with respect to product if structured methodology is provided	(Braaksma et al., 2011; Frisch, 2007; Hennig et al., 2016, 2015; Pereira et al., 2011; Van Renssen, 2003, 2005)

4.10 Analysis of Findings

Thus, from the results of Table 4-2, OWL/SWRL is a good candidate for representing all identified key concepts and relationships as unary and binary predicates in the form of classes and properties of PM-DEA for achieving DEA. Due to the limitation of ontologies based on OWL2 in representing engineering rules, another formal standard needs to be incorporated to represent engineering rules with n-ary relationships, which can be based on logic as well as maths. Similar to OWL, the limitation of Gellish in context to the needs of DEA is representation and codification of engineering rules as multiple ary predicates.

Engineering rules can be represented either in RuleML or RIF. Both RuleML and RIF are based on horn logic semantics and have various versions. For example, some aspects of Datalog RuleML can be mapped to RIF Core Dialect, Derivation RuleML to RIF Basic Logic Dialect (RIF BLD) and production rule sublanguage of reaction RuleML to RIF Production

Rule Dialect (RIF PRD) (Feigenbaum et al., 2013). Both RuleML and RIF have data types and built-ins in the form of logical operators for comparison such as greater than, less than and basic mathematical built-ins such as multiply, divide along with logical operators for strings and boolean value operations (Horrocks et al., 2004; Polleres et al., 2013). Datalog RuleML as integrated with OWL becomes SWRL (OWL DL & Unary/Binary Datalog RuleML). Thus SWRL is a purposeful extension to OWL and covers most of the features of RIF BLD. Although RIF was although originally designed for exchange of knowledge between rule languages such as RuleML and SWRL, it can also be considered as a rule language. However, there are a few differences between RIF and SWRL. As compared to SWRL, RIF BLD offers a few advantages such as provision of multiple-ary predicates as properties as compared to unary/binary predicates as properties in SWRL. Also, RIF BLD has more built in functions as compared to SWRL (Feigenbaum et al., 2013). However, this problem can be avoided by incorporating additional predicates as properties in SWRL. The major advantage of SWRL is ease of integration with OWL2 formalism along with the reasoners which support both DL reasoning and Horn Logic reasoning with separate reasoners such as Pellet, Racer for DL reasoning and Jess, Drools for Horn Logic reasoning, within a single integrated development environment (IDE) such as Protégé (Golbreich and Imai, 2004). Protégé editor also provides both Sematic Query-Enhanced Web Rule Language (SQWRL) and SPARQL Protocol and RDF Query Language (SPARQL) for querying SWRL rules and OWL knowledgebase with RDF/XML representation respectively.As OWL/SWRL is one of the candidates for investigation for formal representation of knowledge in context to DEA systems (Lützenberger et al., 2012) and in spite of a few limitations such as unary/binary predicates, SWRL provides ease of integration with OWL.

OWL/SWRL or OWL/RIF as a combined representation come close to the expressivity of PSL as a single language, although the method of using OWL/SWRL should be precise. Thus

although PSL is more expressive than OWL/SWRL, due to the limitations of PSL for knowledge representation of design systems and in compliance with the research design requirement of availability of supporting tool for experimental verification of formal axioms, OWL/SWRL formalism has been selected within protégé environment with Pellet for DL reasoning and Drools for Horn Logic reasoning. However, a careful consideration for implementation of SWRL on top of OWL is that the variables, relations and individuals in Datalog RuleML as SWRL should consist of OWL ontology elements in the form of classes, properties and instances. Thus, modelling of the OWL ontology needs to be accurate in order for SWRL rules to include ontology elements and reasoning to provide accurate results.

4.11 Summary

This chapter has identified the Meta model based on key concepts and relationships as F-B-S aspects of the process model for DEA as part of the novel aspect of this research. The experimentation of these concepts was performed using existing platform independent and neutral formal representation standards such as PSL, RuleML, OWL and SysML. Requirements were compiled for the formalised representation to enable DEA for all the specified concepts and relationships. The results of the comparative analysis along with the research design for a supporting tool to test the axioms for a KR language revealed OWL/SWRL to be a suitable candidate. Both these tasks will address the other primary novel aspect of this research by providing a method for ontology development of identified concepts and relationships in OWL/SWRL as neutral formal representation with inference and reasoning and semantic clarity. The next chapter will address these aspects by utilising the high level, intermediate and low level concepts as the Meta model for development of a generic process model for DEA. The model schema will provide the method for populating OWL/SWRL as a suitable ontology for DEA with neutral formal semantics.

5 Development and Implementation of Process Model for Design Engineering Automation: Ontology Based Approach

5.1 Introduction

Chapter 4 identified the key concepts and relationships as Meta model with F-B-S aspects for development of a generic process model for DEA for mechanical design with DFM. It also discussed experimentation of the Meta model aspects with neutral formal representation standards and their comparative analysis as per compiled requirements. This chapter will initially discuss the development of a generic process model, which is named in the thesis as "Generative Process Model for Design Engineering Automation (GPM-DEA)" based on the high level; intermediate and low level concepts as the Author's Meta model. The second half will elaborate on the method of the schema mapping of GPM-DEA knowledge model for mechanical design process with DFM to OWL/SWRL ontology. The ontology development methodology is in line with the approach discussed in research design in section 1.4.2. It is claimed that the ontology representation will achieve the requirements of DEA for mechanical design with DFM process with generative modelling capability as per KBE perspective based on functional requirements, and with the effect of unified/integrated process model on product's geometric attributes. For this purpose, Use case 3 and 4 will be discussed for system development and experimental verification of the claim in the next chapters. This will be performed with the assistance of inference results on neutral formal representation as the use of ontologies, with the help of reasoning and query mechanism on author developed set of predefined generic functions with semantic clarity.

5.2 Initial Process Model for Design Engineering Automation

An initial process model was developed by the author, with the literature review findings on knowledge entities of mechanical design process with DFM for formulation of key concepts

and relationships as Meta model, strengths and weaknesses of existing informal/semiformal and formal modelling standards for their knowledge modelling and experimentation of the Author's Meta model concepts and relationships with neutral formal representation standards based on pilot use-cases. This is illustrated with the help of Figure 5-1.

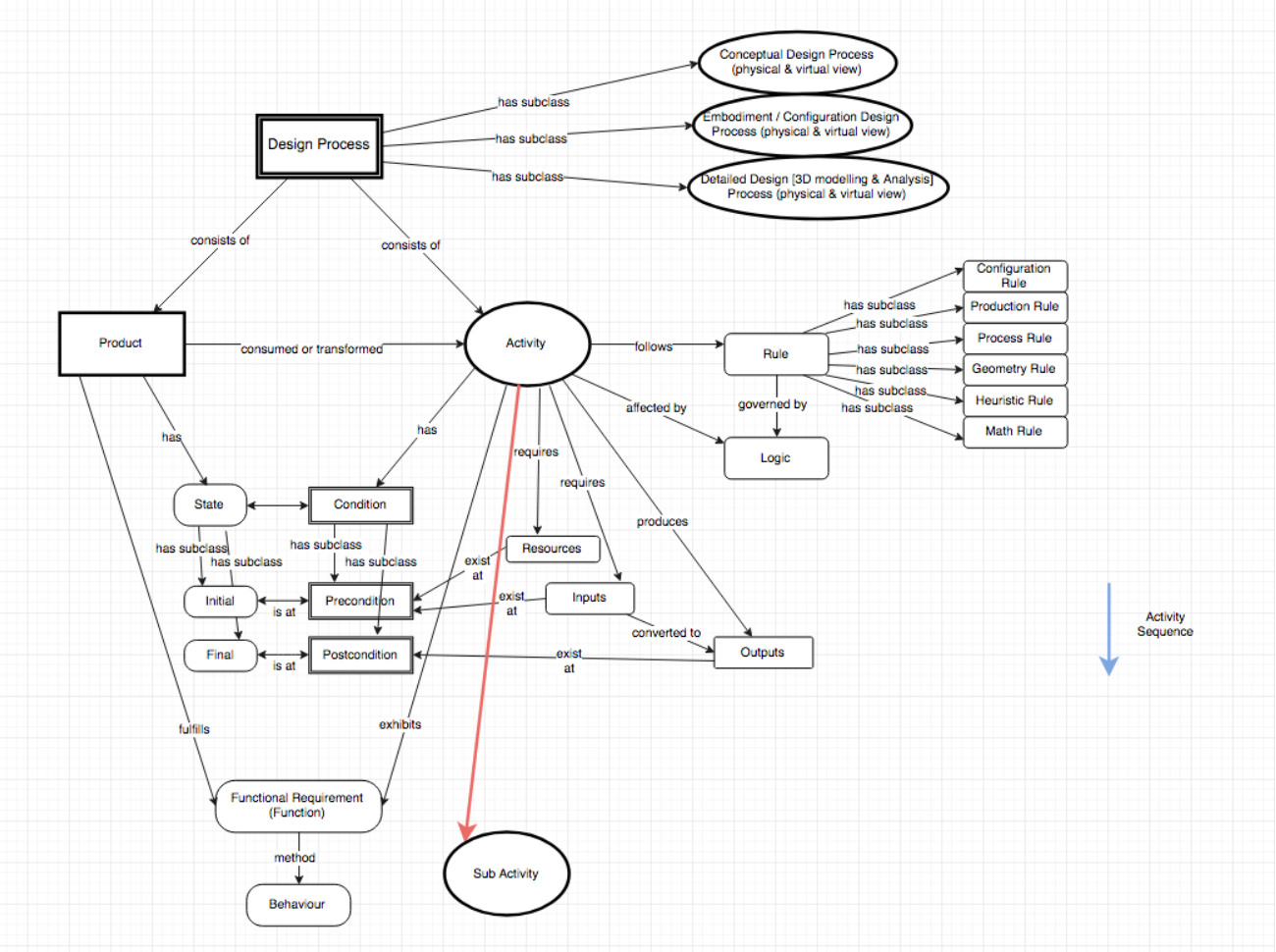


Figure 5-1: Initial Process Model for DEA as Informal / Semiformal Representation

As it is observed, that although existing modelling standards such as IDEF0, IDEF4, UML, SysML can capture most high level identified concepts and relationships of the author's metamodel of the process model informally but to capture all the aspects requires merging of existing standards utilising a hybrid approach and modifications for amendments. Thus the initial version of an instance of the process model consists of these concepts (Meta model) – design process, activity, product, rule, logic, inputs, outputs, resources, functional

requirements-function, behaviour, state and condition. However, in compliance with the requirements formulated from a process model for implementation in neutral formal representation for DEA in context to KBE in section 4.7.1, condition and state classes are not required as the design process can take multiple routes within a process and just needs to reflect its output in terms of product's geometric attributes. These geometric attributes can be used across different bespoke DEA systems in the form of parametric CAD systems such as Siemens NX Fusion, CATIA Knowledgeware enabling GA, CAM systems and KBEs such as AML and ParaPy. In order to further refine the process model and its ontology system development, Use case 4 and 5 will be instantiated in order to refine the process model and verify the effect of GPM –DEA on the product's attributes. A revised version is illustrated in Figure 5-2.

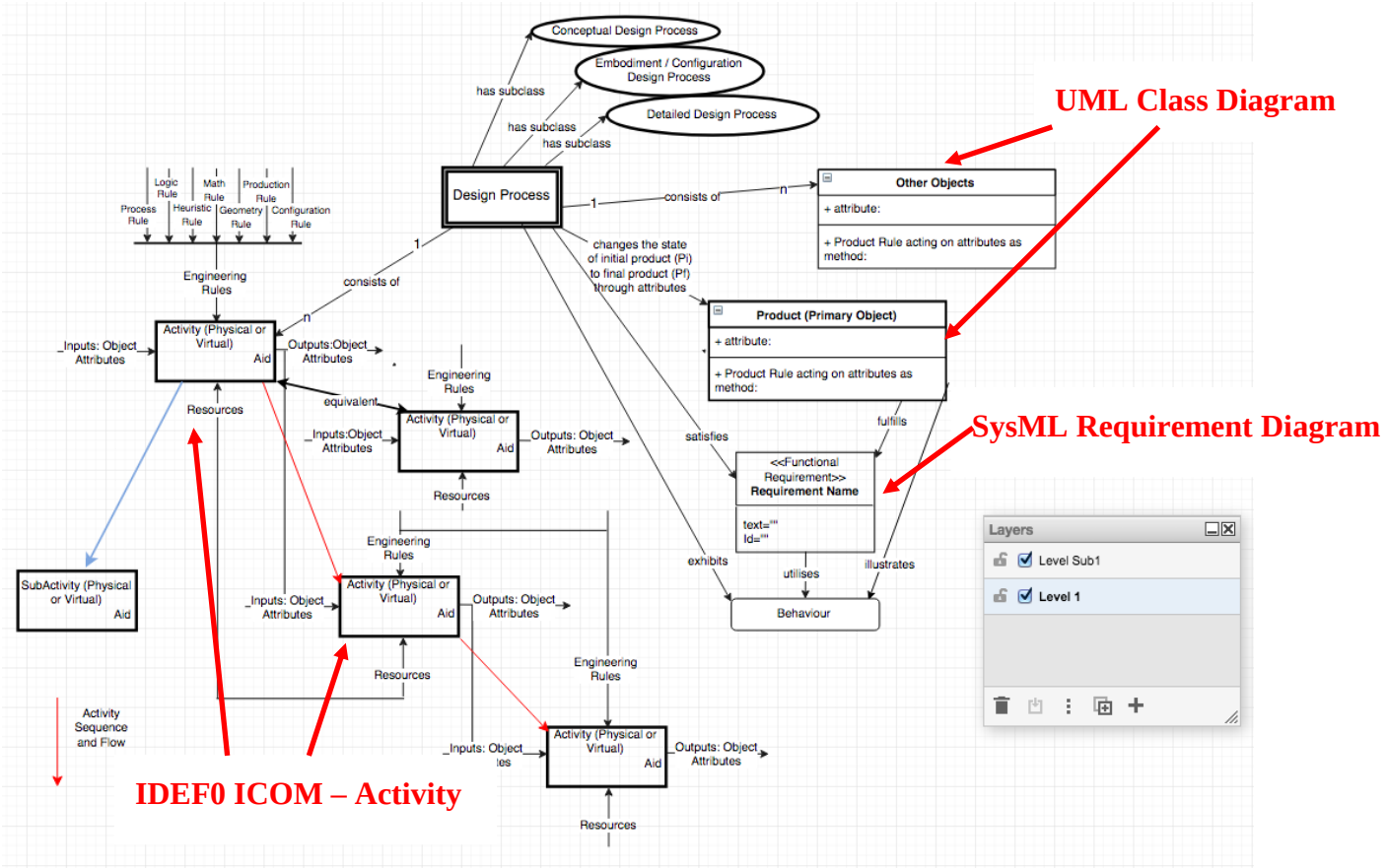


Figure 5-2: Revised Process Model for DEA as Informal / Semiformal Representation

However, as per the combination of critical analysis of literature review and experimentation with pilot use-cases, the process model based on the F-B-S aspects of the Meta model was revised with alterations in order to fully address the needs of DEA system. The formulation of final version of the process model developed by this research is illustrated as follows.

5.3 Development of final version of GPM-DEA – Relationships of MetaModel

The purpose of the product design process is to satisfy a set of functional requirements (Chen et al., 2008). IDEF0 with its syntax is used to describe the structural (S) effect of a process with functional modelling approach (Chang et al., 2008). This is due to the fact that IDEF0 enables functional modelling.

As developed and refined by author, the one to many relationships for the activity as the primary concept of the process model with F-B-S aspects for DEA is described as follows –

1. ***Activity satisfies a function*** which is a sub-function of the ***design process functional requirements*** **(*SysML Requirement Diagram*)**
2. ***Activity requires inputs for conversion to outputs*** which are described in terms of ***product specific attributes or parameters*** ***(UML Class Diagram)*** as well as independent re-usable objects
3. ***Activity is controlled by engineering rule***, which may be a design or a manufacturing rule, which control its completion. There are various types of rules such as process, logic, heuristic, geometry, math, production and configuration rule. These also include the trade-offs between design and manufacturing constraints
4. ***Activity requires resources*** which may physical elements such as fixture, jig for manufacturing process and virtual elements such as CAx tool for the design process

5. ***Activity is described by an integer id, Activity has sub-activity***
6. ***Activity is followed by (successor) an activity***

Detailed analysis by author has revealed the following key observations of existing modelling standards which satisfy few aspects of the Metamodel developed by author to target DEA with focus on mechanical design with DFM process –

- **Strength of *IDEF0*** is capturing of all points except 1, which shows the simultaneous function as a sub-function of the design process functional requirements and 6, which allows for process logic. Thus, these 2 relationships are added to IDEF0 for activity completion in context to the needs of activity knowledge capture for DEA, which is elaborated in section 5.3.
- **Strength of MBSE language diagrams** such as ***UML class diagram*** or **S*ysML block definition diagram*** is they are able to capture static aspects of product attributes. Similarly, ***SysML requirement diagram*** is able to capture functions of a process in context to the product for DEA (Finance, 2010). Thus UML class diagram is used to represent the object attributes and the SysML requirement diagram for functional requirements. It is important to notice that although UML and SysML activity diagram are also successful in capturing activities with inputs and outputs (Weilkiens, 2007), thus fulfilling points 2 and 5 of activity relationships but are not able to fulfill points 3 and 4 as they can't incorporate resources and rule in the same diagram. This is the reason for selecting IDEF0 for activities and UML class diagram and SysML requirement diagram for product model and functional requirements respectively in context to the needs of DEA.
- **Strength of UML class diagram** is it can represent **engineering rules as methods** to convert activity inputs to activity outputs in terms of product attributes. This eliminates the need to use the SysML parametric diagram separately for

representation of rules. Both **IDEF0** and **UML** have been used as knowledge objects for modelling of conceptual design of aerospace assembly processes (Mas et al., 2013).

The author has added ***UML condition link on top of IDEF0*** to successfully model process rules for controlling the sequence of individual IDEF0 activities as a red link. Similarly, blue link is added to represent the sub-activities of individual activities. The author has also added the behaviour concept separately as a knowledge object to the activity, object and function thus completing the function-behaviour and structural (F-B-S) aspects of the process model for DEA. The author has also added a ***relationship (as an arrow) between IDEF0 ICOM activity box and its functional mapping to SysML requirement diagram*** in green borderline as individual sub-functions with a pink link. An instance of this is illustrated in Figure 5-3.

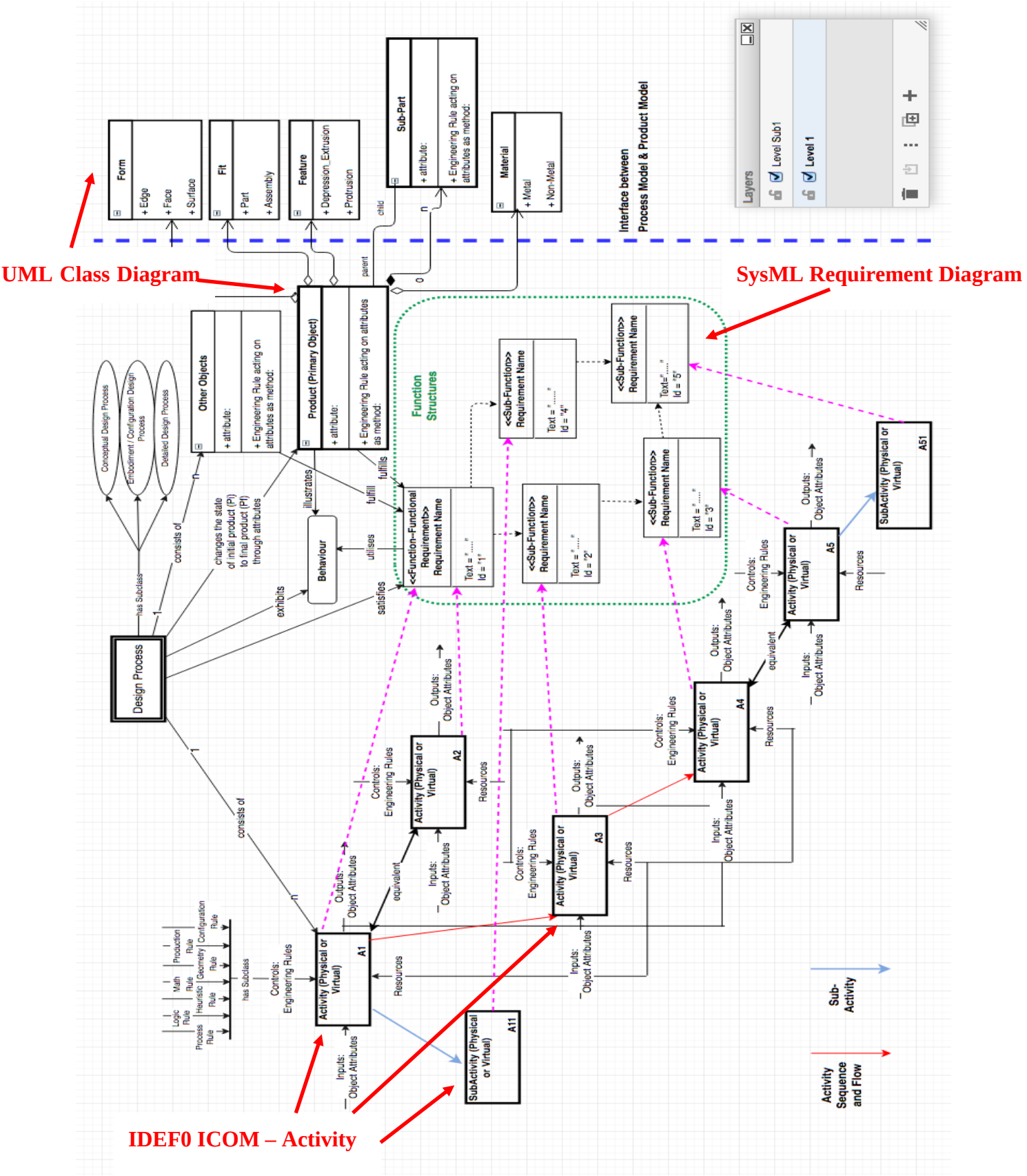


Figure 5-3: Instance of Generative Process Model for Design Engineering Automation (GPM-DEA) as Informal / Semiformal Representation – Developed by Author

5.4 Functioning of GPM-DEA – Coherent Process Knowledge Model

This research has developed a generative process model for design engineering automation (GPM-DEA), which is dynamic in nature through its ontological neutral formal representation. It is explained in detail in this section.

5.4.1 Workability

The working of GPM-DEA as developed in this research is shown in Figure 5-4. The functional requirement of the design process is broken down into sub functions, which are represented using ***SysML requirement diagram***. SysML requirement diagram is used for illustrating functional requirements of engineering design process in context to the product as the primary object (Weilkiens, 2007). In order to generate activities and objects as *generative modelling capabilities* developed in this research, the sub functions are matched to activities and objects, which fulfill the same functions. However, this can only be achieved during representation of GPM-DEA in formal standards.

The product in initial state is assessed and then its geometric attributes are marked as activity inputs and outputs. Activity description is captured using *an IDEF0 notation*. **IDEF0** has inputs, controls, outputs, and mechanisms (ICOM) as described in context to engineering design processes (Pugliese and Colombo, 2014). Controls can be entities or laws guiding the process, which in this case become the engineering rules based on logic and maths. Mechanisms are synonymous to resources, which are used but not consumed or transformed directly during an activity.

Thus in the developed process model, IDEF0 illustrates design process activities with inputs, outputs, rules as controls and resources as mechanisms. There are various subclasses of rules - process rules, logic rules, heuristic rules, math rules, geometry rules, production rules and configuration rules.

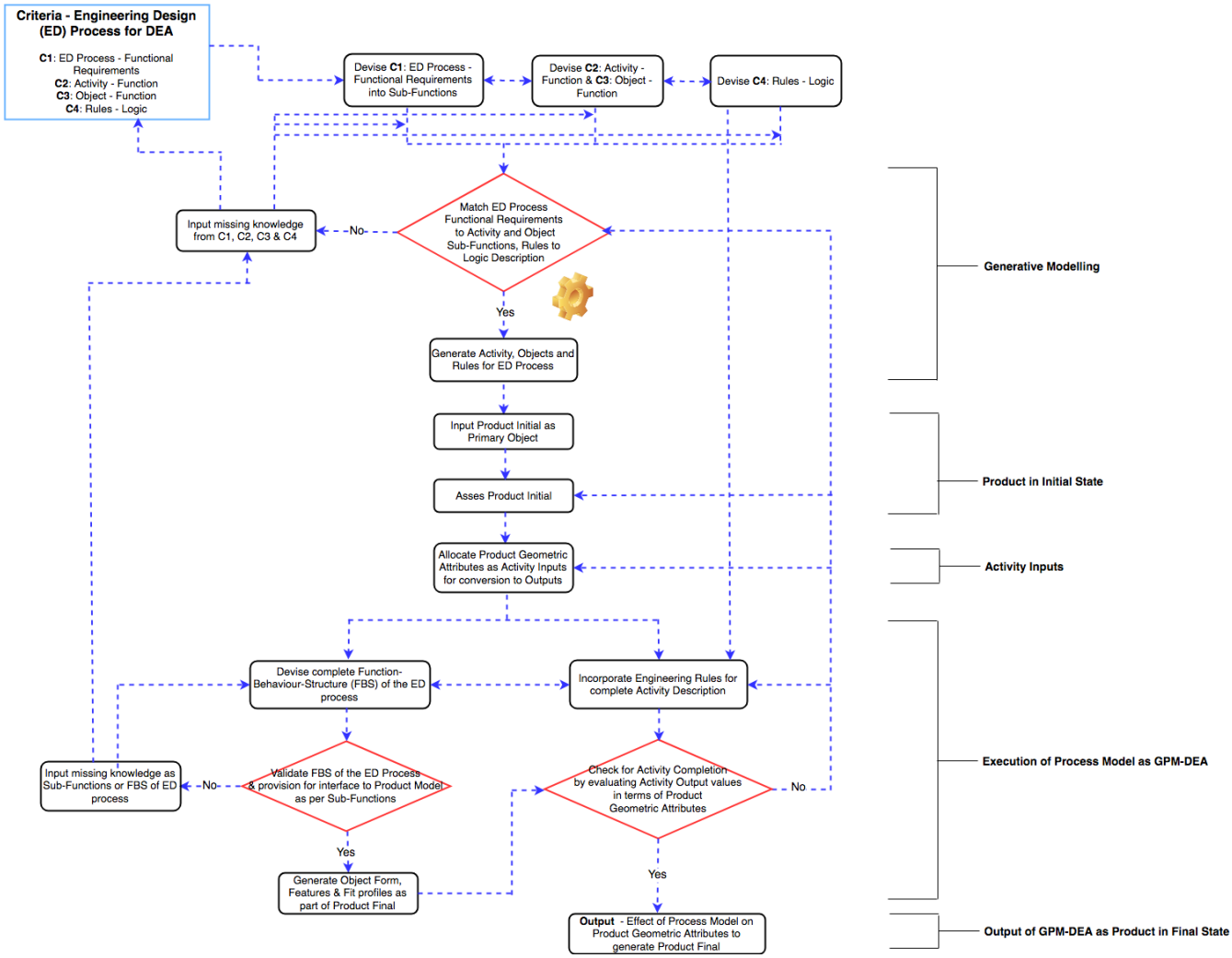


Figure 5-4: Working of Generative Process Model for Design Engineering Automation (GPM-DEA) – Developed by Author

As IDEF0 corresponds to functional modelling, all design activities satisfy a function. GPM-DEA, based on IDEF0 for activities, is based on dependency modelling for analytical purposes in the form of DEA (Wynn and Clarkson, 2017). Process rules for sequencing and optimisation are represented with UML condition links.

Process adding semantics to product function and behaviour has been imperative in incorporating both function (functional requirement) and behaviour in respect to process modelling approach of this research. Similarly, a crucial point is to capture the relationship of the process function and behaviour in context to the change of state of product through its

attributes as one of the key artefacts especially during the conceptual and preliminary design phase. Thus, in GPM-DEA process has function & behaviour and structure in terms of an ICOM box with all aspects in relation to product geometric attributes.

As stated earlier in the thesis, the process model developed by the author also adopts basic principles of OPM as ISO/PRF PAS 19450 with the change of state of the product from initial state to final state in context to process execution. However, it was mentioned in section 4.8.2 that although, OPM goes to various levels of abstraction to represent the complete F-B-S of a system, it doesn't fully model the individual activities of a process model and provides very less relation between the activities and its implementation as formal representation. This is the reason that OPM notation has not been utilised for informal/semiformal representation for the activity and related concepts of the process model developed by the author.

The change of state of product from initial state to final state upon acted upon by a process is reflected by change in its attributes as also adopted from IDEF3 and IDEF4 methodology. In order to reflect the effect of process model on product geometric attributes in this work, ***UML class diagram*** is used for product model with attributes and engineering rules as methods.

Interface of the process model with product model is illustrated in Figure 5-4 where, UML class diagram can represent parts and assembly relations with composition links and also parent child relations for the product.

Thus GPM-DEA is built upon existing standards such as IDEF0, UML and SysML and **incorporates additional constructs** such as sequencing and flow of activities based on process rules, automatic generation of activities and objects based on function matching for complete Function-Behaviour-Structure (FBS) representation in order to address the needs of DEA. As GPM-DEA model has various sub-levels, an instance of GPM-DEA is illustrated in Figure 5-3 at its highest level of abstraction.

5.4.2 Pilot Use Cases - Function Structure Matching: Basis of Generating Activities and Objects of GPM-DEA

The instance of GPM-DEA as shown in Figure 5-4 is a graphical representation corresponding to the knowledgebase consisting of all concepts for FBS representation of the engineering design process. Figure 5-5 shows the knowledgebase compiled in this work, where various design processes exist with their functional requirement, which is broken down into sub functions.

Engineering Design Process	Engineering Design Process Functional Requirement	Sub-Functions
Precision Forging1	<Achieve an accuracy of +/-1mm in shape prediction>, as shape prediction accounts for a bulk of the manufacture objectives.	Achieve an accuracy of +/-1mm + shape prediction
Conceptual Design1 Fan Blades	The fan blades <spin to accelerate a mass of air> into the engine to <generate thrust> that propels the aircraft forward. Fan blades also function to <reduce total engine damage> from the <ingestion of various foreign objects> such as birds by radially deflecting outward such objects rather than passing them through to the core parts of the engine. The <dovetail attachment> of fan blades are used to <secure the blades to the hub or disk>. It is important to allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength	Generate thrust + reduce total engine damage from the ingestion of various foreign objects such as birds + secure the blades to the hub or disk + allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength

Figure 5-3: Example of Engineering Design Process with corresponding Functional Requirement and Sub-Functions: Knowledgebase

An example of both design processes and activities from Use Case 1 & 3, which includes physical, informatical and virtual activities with their corresponding sub-functions as functional requirements, is illustrated with Figure 5-5 and Figure 5-6 respectively.

Engineering Design Process Activity (Physical/Informatical)	Functional Requirement --[Sub-Functions]
Extrusion	Extruded stem long enough for the part to be handled in subsequent operations + base of the extruded part needs to have enough material for subsequent heading + shape prediction
Stamping	Enough energy in the top die to achieve the movement required to reduce the gap between the dies to the sum of the gap between the flash lines and the minimum running thickness + shape prediction
Heading	The heading cavity is filled out as much as possible even if the cavity does not get filled in its entirety + shape prediction
Blade Geometry Optimisation	Fan blades spin to accelerate a mass of air into the engine + generate thrust + 80% of the thrust delivered by fan + reduce total engine damage from the ingestion of various foreign objects such as birds
Dovetail Attachment	Secure the blades to the hub or disk
Material Selection	Allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength

Figure 5-4: Example of Engineering Design Process Activities with corresponding Functional Requirement as Sub-Functions: Knowledgebase

Thus, from Figure 5-5, the engineering design process – Conceptual Design1 Fan blades has 4 sub-functions – 'generate thrust', 'reduce total engine damage from the ingestion of various foreign objects such as birds', 'secure the blades to the hub or disk' and 'allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength'. From Figure 5-6, activity 'Blade Geometry Optimisation' satisfies the 2 of these sub-functions - 'generate thrust', 'reduce total engine damage from the ingestion of various foreign objects such as birds'. Similarly, the activity 'Dovetail Attachment' satisfies the function – 'secure the blades to the hub or disk' and the activity 'Material Selection' satisfies – 'allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength'. Thus, the design process - conceptual design1 fan blades should consist of these 3 activities - 'Blade Geometry Optimisation', 'Dovetail Attachment' and 'Material Selection'.

Similarly, ‘Precision forging1’ as a DFM process satisfies 2 sub-functions – ‘Achieve an accuracy of +/-2mm’ and ‘shape prediction’. All the existing activities in the knowledgebase, which fulfill a subset of these functions are - ‘Extrusion’, ‘Heading’ and ‘Stamping’. Thus ‘Precision forging1’ should consist of these 3 activities as ‘Extrusion’, ‘Heading’ and ‘Stamping’. All the activities with their inputs, outputs, controls as rules and mechanisms as resources along with participating objects as inputs is shown in Figure 5-7. Their corresponding graphical representation is an ICOM box of IDEF0 standard in GPM-DEA.

Engineering Design Process Activity (Physical/Informatical/ Virtual)	Objects - Product as primary object	Activity Inputs	Activity Outputs	Controls: Engineering Rules	Resources
Extrusion	Product - Fan Blade or Compressor Blade or Turbine Blade, Object - Punch	Material data – Billet & Dies, temperature to which the workpiece is heated up in the furnace prior to extrusion, Furnace Transfer Duration, Duration for which the workpiece rests on the die, Die Temperature, Press Characterisation, Punch stopping position	Extruded part	Extrusion Rule1	Billet (without glass coating), Tongs, Dies (Nominal)
Heading	Product - Fan Blade or Compressor Blade or Turbine Blade, Object - Punch	Material data – dies, Air Transfer duration, Duration for which workpiece rests on die, Die temperature, Press characterisation, Punch stopping position	Headed part		Tongs, Dies(nominal)
Stamping	Product - Fan Blade or Compressor Blade or Turbine Blade	Material data – dies, Temperature to which workpiece is heated up in the furnace prior to stamping, Furnace transfer duration, Duration for which workpiece rests on die, Die temperature, Press characterisation, Gap between flash lines & running thickness	Stamped part		Tongs & dies(nominal)
Blade Geometry Optimisation	Product - Fan Blade, Compressor Blade, Turbine Blade	Aerodynamic forces acting on a local airfoil and global changes in momentum, Rate of air intake, Proper airfoil section, twist, chord, and pitch angle for optimal thrust distribution	Incremental Lift created by each blade, Ideal power	BladeGeometryOptimisationRule1, BladeGeometryOptimisationRule2	BEM Theory
Dovetail Attachment	Product - Fan Blade, Compressor Blade, Turbine Blade	Proper airfoil section, twist, chord, and pitch angle for optimal thrust distribution	Incremental Lift created by each blade, Ideal power	DovetailAttachmentRule1, DovetailAttachmentRule2	
Material Selection	Product - Fan Blade, Compressor Blade, Turbine Blade	Cost, weight, aerodynamic efficiency	Material Alloted		

Figure 5-5: Example of Engineering Design Process Activities with Inputs, Outputs, Rules and Resources with Objects: Knowledgebase

A snapshot of the rule description controlling the activities is shown in Figure 5-8.

Controls: Engineering Rules	Engineering Rule Description	Type of Rule
BladeGeometryOptimisationRule1	The rate of air intake varies and is dictated by factors such as airfoil geometry, angle of attack, air density and the speed at which the airfoil moves through the air	Configuration Rule
BladeGeometryOptimisationRule2	A 30% hollowing in a hollow fan blade results in about a 13–16% decrease in torsional rigidity compared to a solid blade design	Configuration Rule
Extrusion Rule1	A short extruded stem left behind after stamping would cause problems in handling the part while a long stem would result in excessive material use & high costs.	Configuration Rule

Figure 5-6: Example of Engineering Rules controlling the Design Process Activities: Knowledgebase

5.4.3 Types of Engineering Design Process with Variable Concepts: Function and Objects

All the sub-classes of the design process cannot be illustrated here in the Figure, as we need to go to sublevels. For example, the design process for this thesis has to cover conceptual / preliminary design, embodiment / configuration design, detailed design and other crucial aspects such as DFM, DFA as part of embodiment design. The detailed hierarchy of engineering design process, which can be implemented in DEA systems, needs to cover all aspects of the design process with high level, intermediate and low level concepts identified in this thesis such as design for assembly (DFA), design for manufacturing (DFM), fluid flow analysis, structural analysis, thermal analysis, stress analysis, detailed design process aspects such as form, features and fit with 3D modelling, computer aided engineering (CAE) analysis process such as computational fluid dynamic (CFD) analysis, finite element analysis (FEA) analysis, pre-processing, post-processing, computer aided manufacturing (CAM) process such as casting, joining, machining and so on. In order for *function matching* to work, which will be illustrated later, sub functions are classified in this work as – geometric 3d shaping / sizing, manufacturing feasibility such as attach / connect & positioning, output electrical magnetic performance such as capacitance, electric field, voltage, energy, power, work, output mechanical performance such as acceleration, fatigue, force, hardness, momentum, stiffness, strain, strength, torque, velocity and output thermodynamic performance such as compression, expansion, flow, foreign object damage, heat, pressure and vibration. The complete list of both design process and function subclasses are illustrated later.

Pertaining to this research, as the process model has the effect on object attributes used across DEA systems applications, the object model including the product knowledge needs to cover basic aspects such as feature, form, fit and material. There is an interface between the process model and product model as observed from Figure 5-3. The product model can be expanded

to include detailed product knowledge. As part of this research, the following aspects are included which can be further extended as integration to the product model. Features include depression / extrusion features such as hole, notch, pocket, slot and protrusion features such as block, shaft. Fit includes part and assembly relationships. Form is broken down as edge, face, surface and volume. Edge is further broken down as chamfer, fillet and line. Similarly, face is broken down as circle, ellipse, hyperbola, parabola and polygon with variable sides. Surface is broken down as Bézier and Non-uniform rational basis spline (NURBS). Volume is broken down as box, cone, cylinder, ellipsoid, hyperboloid, paraboloid, polygon volume and sphere. Material is further classified as alloys, ceramics, composites, ferrous metal, non-ferrous metal and polymer. Alloys are classified as brass, bronze, duralumin, inconel, nimonic and manganin. Ceramics are broken down as boron carbide, boron oxide, silicon carbide and silicon nitride. Composites are broken down as glass fiber, carbon fiber and so on, ferrous metal as carbon steel, cast iron, mild steel and so on, non-ferrous metal as aluminium, copper, lead, nickel, tin, titanium, zinc and so on. Similarly, polymers are further classified as neoprene, plastic, polyethylene, polypropylene and so on.

Product has been divided into two main classes – product_initial and product_final. The product_initial indicates the state of the product at the beginning of the design process; product_final indicates the state of the product at the end of the design process.

5.5 Synthesis of GPM-DEA

In order to address the needs of DEA, integration of various engineering design concepts and relationships with focus on mechanical product design process with DFM knowledge has been achieved by developing GPM-DEA in this research. GPM-DEA provides a coherent method to build structured knowledge model and enables automation with generative modelling by automatic generation of activities and objects by matching the functions as functional requirements of the design process with corresponding functions of activities and

objects. GPM-DEA includes all concepts of the Author's Meta model in context to process modelling for DEA with focus on mechanical design with DFM knowledge, and preserved semantics based on knowledge entities such as activity, function, behaviour, object and its attributes as structure being affected by rules and logic in a coherent and structured manner. It provides categorisation for sub-functions and object knowledge model with geometric attributes along with integration facilities to the product model. This allows for an unified/integrated and highly granular process model ready for implementation in a neutral (open standards) formal representation framework for DEA ensuring correct syntactic and semantic mapping of the informal/semiformal model to the formal model.

5.5.1 GPM-DEA – Hybrid Representation of Existing Modelling Standards

Thus, in order to develop a coherent and structured process based knowledge model, the author has exploited the strengths of the existing modelling standards and added the constructs on top of the integration. The working of the developed process model, GPM-DEA as informal/semiformal representation for visual display by the author can be summed up as –

1. ***IDEFO ICOM box* for activity description** with inputs and outputs in terms of product attributes along with links to rules as controls and resources
2. **UML class diagram for product knowledge** with engineering rules as methods
3. ***SysML requirement diagram*** for functional requirements
4. ***UML condition link*** for process rules and flow
5. ***Bi-directional relations*** between function, process links, objects, activity description behaviour for complete F-B-S of a process model

Thus the author has combined the strengths of IDEF0, UML and SysML and added constructs on top to develop a hybrid representation of GPM-DEA. Some of the few critical aspects of a process model for DEA using a KBE approach is ***generative modelling*** capabilities which means that the individual activities should not be static and must be

generated from the initial specification or the design intent in the form of functional requirement classification. In context of this research, as the design process satisfies a functional requirement, all the activities, which fulfill functions, as part of the design process should be automatically generated. Thus in compliance with function structure, the functional requirements of the design process as captured in SysML requirement diagram are broken down into sub-functions. All the activities, which match the individual sub-function instances, should be automatically generated for DEA.

5.5.2 GPM-DEA - Generative Modelling Aspects

The following are the crucial aspects of *generative modelling* of GPM-DEA, which have been embedded by the author in formal OWL ontology representation with the help of predefined set of generic functions in context to DEA with a KBE approach -

1. ***Generation of activities based on sub-functions*** *as functional requirements*
2. ***Generation of objects based on sub-functions*** *as functional requirements*
3. ***Generation of engineering rules for activities based on logic*** *as the basis of rules*
4. ***Assessment of initial product to generate the initial activity*** *of the process model*
5. ***Virtual and physical activity functional equivalence***

These will be elaborated in the next section 5.6 which explains OWL ontology development based on the schema of GPM-DEA. GPM-DEA has been developed with assistance of pilot use cases and the requirements formulated for DEA. It has been further refined with the usage of test use-cases, discussed in the next chapter for refinement of Meta model concepts and relationships to incorporate product's geometric attributes and further system development. The results of comparative analysis of available formal standards as per the formulated requirements for a KRS to enable DEA as discussed and analysed in section 4.9, has

recommended OWL/SWRL as a suitable ontological neutral formal representation framework of GPM-DEA with semantic clarity.

5.6 Implementation of GPM-DEA in OWL/SWRL Ontology and Rule Representation: Neutral Formal Representation

This research thesis has developed an ontology for the mechanical design process with design for manufacturing (DFM)/ design for assembly (DFA) based on the schema of the structured GPM-DEA knowledge model. The ontology has been developed using Topbraid Composer FE (Composer, 2011) and Protégé (Horridge et al., 2011) with formal representation standard as OWL2 (Golbreich et al., 2012; Hitzler et al., 2012; Horridge and Patel-Schneider, 2012; Motik et al., 2012) as the basis for axioms. OWL2 is based on formal logic SROIQ (Krötzsch et al., 2012). The main focus of this work is to develop ontology of the mechanical product design process for DEA with the effect of the process model on the change of state of the product in terms of its geometric attributes.

As explained in the earlier sections, engineering rules form a very integral and crucial part of an engineering design process for DEA and have been extensively formalised. However, a limitation of binding engineering rules to a process based approach has been a major limitation as engineering rules have been purely associated with product geometry and features in DEA systems. It was also observed that function, behaviour have been individually modelled in context in product modelling and implemented in ontology encoded in OWL.

The ontology model as OWL/SWRL developed as part of this research constrains the interpretation of the knowledge base through its axioms and allows for subsumption relation validation (class-subclass relationship) and reasoning.

5.6.1 Ontology Development in OWL: Classes, Properties and Restrictions

For OWL/SWRL as ontology implementation of GPM-DEA, the master class under Thing as described in OWL2 is the design process with subclasses as activity and product as one of the underlying main classes. Under the Design Process with activity and product being the main classes of focus, all the other concepts including rule, logic, resources, function or functional requirements and behaviour have been assigned as classes in the developed GPM-DEA. Design process function and behaviour are very crucial to GPM-DEA with function class allowing for generative modelling capabilities using SWRL. Subclasses have been clearly assigned to master classes for example; product_initial and product_final as initial and final state respectively are subclasses of the product class.Similarly, the rule class has different types of engineering design rules classified as production rules in the form of 'If-Then' and 'If-Then-Else' construct, process rules, logic rules, math rules, geometry rules, configuration rules and heuristic rules as its subclasses. Many rules can be classified under multiple subclasses as various classes share common characteristics. However, the heuristic rules are disjoint with logic rules as a member of one class cannot be a member of the other class. Figure 5-9 illustrates the OWL implementation of GPM-DEA with classes and properties.

The activity description concepts have been adopted from (Ding et al., 2009; Zhang et al., 2013) including inputs, outputs, resources, activity id and description along with methods as transformation of inputs to outputs. The methods become synonymous to engineering rules and logic in the engineering design process. They have been implemented with the help of all use cases examples. Inputs, outputs of activity and other specified relationships as arrows between classes in GPM-DEA have been clearly assigned as properties in OWL2 formalism. Properties have been created between concepts as classes and classified as either object or datatype properties.

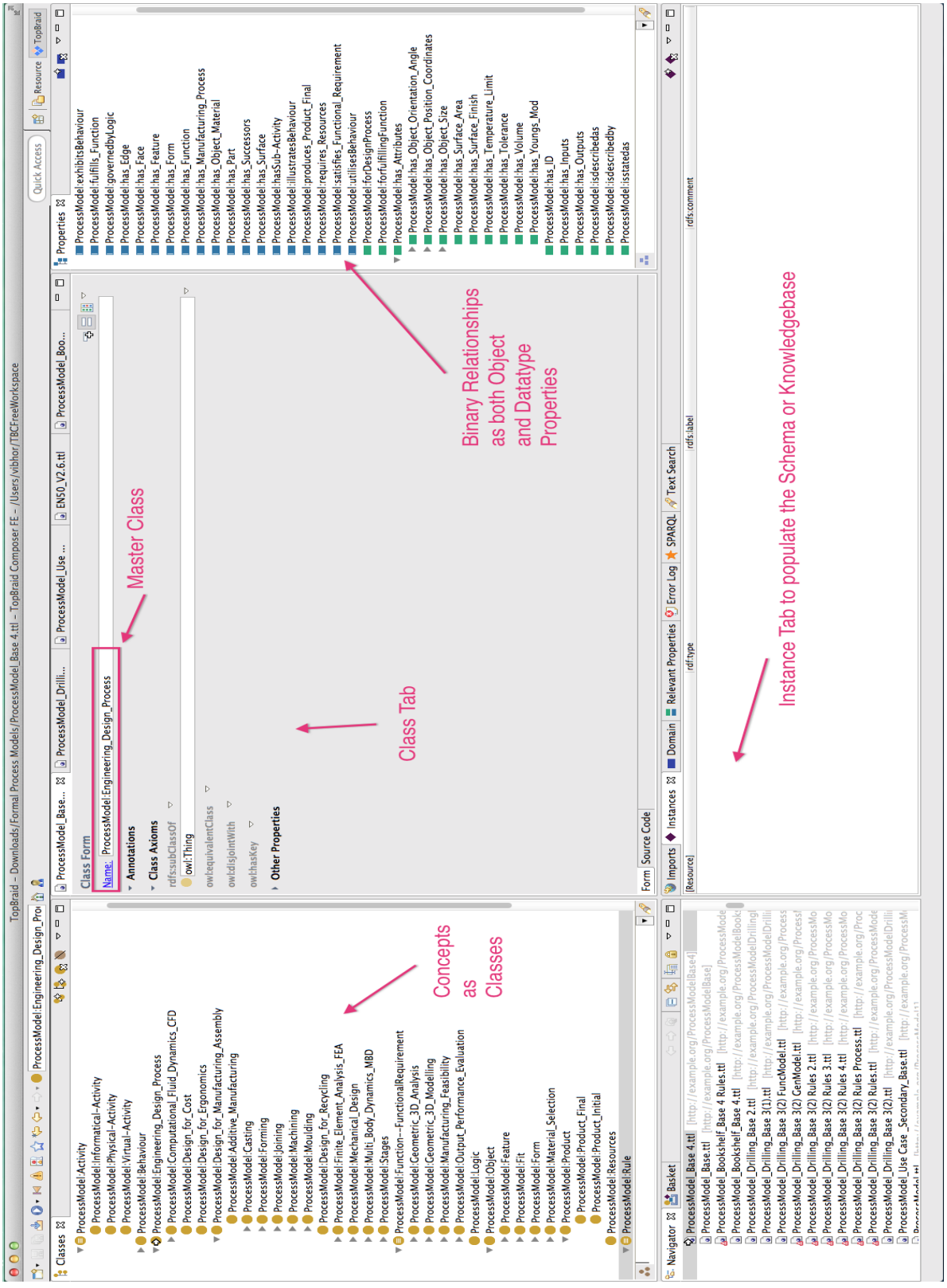


Figure 5-7: OWL implementation of GPM-DEA developed by this research: Classes and Properties

OWL2 allows object properties between individuals of classes and datatype properties between individuals and values such as string, integer, and float. Properties link individuals from domain to range. Thus relationships such as design process satisfies functional requirement (ProcessModel:satisfiesFunctionalRequirement), activity controlled by rule (ProcessModel:controlledbyRule), activity requires resources (ProcessModel:requiresResources) have been implemented as object properties. The rdfs:domain of the property becomes the initial class and the rdfs:range of the property becomes the second class.For example, (ProcessModel:satisfiesFunctionalRequirement) property has been created for which rdfs:domain is the Design_Process class and the rdfs:range becomes the FunctionalRequirement class.Similarly, (ProcessModel:controlledbyRule) property has rdfs:domain as Activity class and rdfs:range as Rule class. This has been illustrated with the help of query in earlier versions of GPM-DEA with Use Case 3 in Section 4.5.3 in Chapter 4. To model the sequencing and optimisation of activities an object property called (ProcessModel:has_Sucessors) has been created with both rdfs:domain and rdfs:range set as Activity class.

Datatype properties have been created such as to model activity has inputs and outputs in terms of object attributes(ProcessModel:has_Inputs), (ProcessModel:has_Outputs); activity has id (ProcessModel:has_ID). Both (ProcessModel:has_Inputs) and (ProcessModel:has_Outputs) have domain as Activity class and range as xsd:float. For example (ProcessModel:has_Object_Size) has been created as a sub property of (ProcessModel:has_Attributes) in GPM-DEA.(ProcessModel:has_Attributes) has domain as Product and Object class and range as xsd:float. (ProcessModel:has_Object_Size) can be marked as a sub property of (ProcessModel:has_Inputs)under which dimensions of objects can be assigned values as sub properties of activity inputs. Similarly, (ProcessModel:has_Object_Position_Coordinates) has been created as a sub property of

(ProcessModel:has_Attributes) to allocate positioning of the parts and assemblies with all 3 co-ordinates as X, Y and Z. (ProcessModel:has_Object_Position_Coordinates) can also be marked as a sub property of (ProcessModel:has_Inputs) under which position coordinates of objects can be assigned values as sub properties of activity inputs.Similarly, the datatype property (ProcessModel:has_Object_Orientation_Angle) created as a sub property of (ProcessModel:has_Attributes) allows allocation of orientation angle of all parts and assemblies with respect to X, Y and Z co-ordinates. (ProcessModel:has_Object_Orientation_Angle) can also be marked as a sub property of (ProcessModel:has_Inputs) under which orientation angle of objectscan be assigned values as sub properties of activity inputs.The datatype property (ProcessModel:has_ID) with domain as Activity class and range as xsd:integer means each activity has an integer id.

OWL2 supports the following types of properties – asymmetric property, symmetric property, functional property, inverse functional property, reflexive property, irreflexive property and transitive property. Functional property can be both datatype and object property whereas inverse functional can only be an object property. Functional property means that the individual from a class can only be associated with one value. Thus (ProcessModel:has_ID) property created in the model is a functional property as it can only be associated with one integer as a datatype property. An inverse functional property means that the inverse of a property is functional and can only be associated with one value but is always an object property.

All the other properties are classified under object properties as well.Reflexive property allows an individual from a class to relate to itself using the property. Any property, which doesn’t allow individual from a class to relate to itself, becomes an irreflexive property. Symmetric property means that if the property relates individuals from class A to class B then the individuals from class B are related to the individuals from class A with the same

property. Property, which doesn't relate back individuals from different classes with the same property, is referred as asymmetric property. Transitive property indicates that if a property relates individuals from class A to class B and also individuals from class B to class C then the property holds true for individuals from class A to class C. All individuals created as instances of these classes will follow these properties as relationships.

Restrictions are axioms that constrain class descriptions in OWL. Following restrictions are supported by OWL2 – quantifier restrictions in the form of existential and universal restriction, cardinality restrictions in the form of minimum, maximum and exact cardinality and hasValue restriction. *Existential restriction* or *existential quantifier* is referred as *someValuesFrom (some)* and may also be denoted as ∃. *Universal restriction* is referred as *allValuesFrom (only)* and may also be denoted as ∀.Existential restriction means that the individuals from a class must hold the property with atleast one individual from the filler class or datatype.

For example, in GPM-DEA ontology model developed by this work, activity class has been created with an existential restriction in the form of (ProcessModel:has_Successors some ProcessModel:Activity), (ProcessModel:has_Inputs some xsd:float). These axiom in the form of *existential restriction (some)* means that all individuals from Activity class will need to hold (ProcessModel:has_Successors) object property with rdfs:domain set as Activity and rdfs:range set as Activity with atleast one individual from the filler class Activity. In natural language, it indicates that all instances of activity will need successor activities in order to describe them for a DEA system. Similarly, the existential restriction in the form of (ProcessModel:has_Inputs some xsd:float) constrains that all individuals of the Activity class must hold (ProcessModel:has_Inputs) datatype property with rdfs:domain set as Activity and rdfs:range set as xsd:float with atleast one individual from the filler datatype float.In natural

language, it indicates that all instances of activity will need inputs as object attribute float values in order to describe them for a DEA system.Similarly, an existential restriction has been created on the Activity class with another datatype property in the form of (ProcessModel:has_ID). The restriction is stated as (ProcessModel:has_ID some xsd:integer) which indicates that all instances of Activity class will hold (ProcessModel:has_ID) property with the filler as an integer datatype. In natural language it indicates that all activities will hold an ID in order to describe them for a DEA system.The existential restrictions on activity class along with subclasses of Rule are shown in the Figure 5-10.

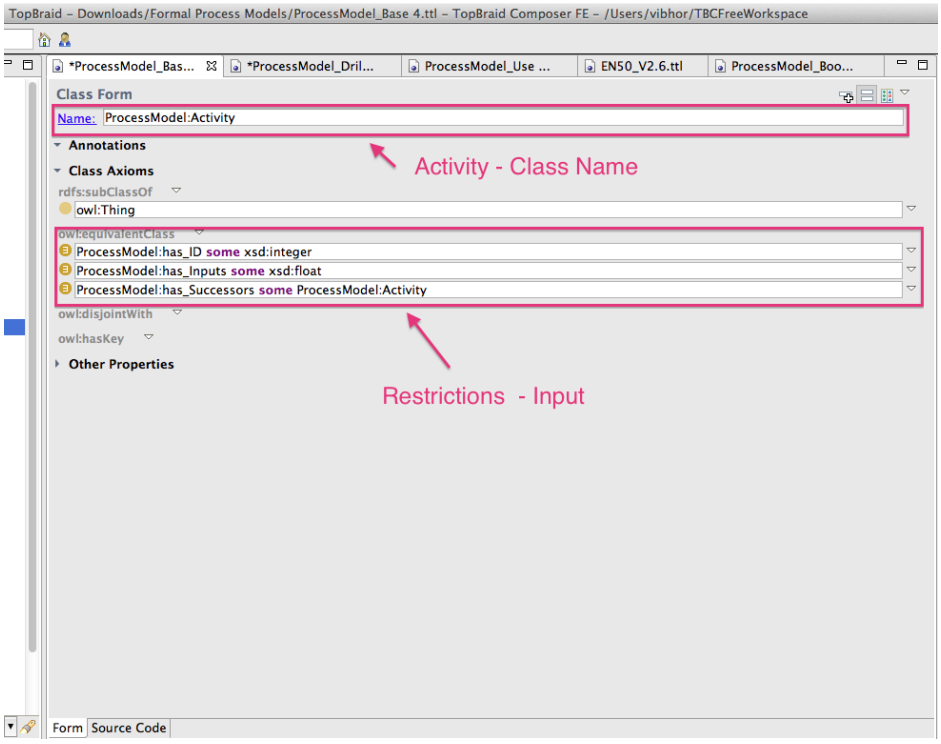


Figure 5-8: Axioms for Restrictions on Activity Class

SPARQL query will generate the classes and relationships based on the defined process model as GPM-DEA.

5.6.2 Function Structures, Design Process and Objects: Class Specification

The engineering design process covers a wide lifecycle from conceptual design to the detailed design stage as discussed in literature review in Chapter 2 and development of GPM-DEA. As elaborated in section 5.3.3, the various types of engineering design process with

their stages in class hierarchy in OWL representation are illustrated with the help of Figure 5-11.

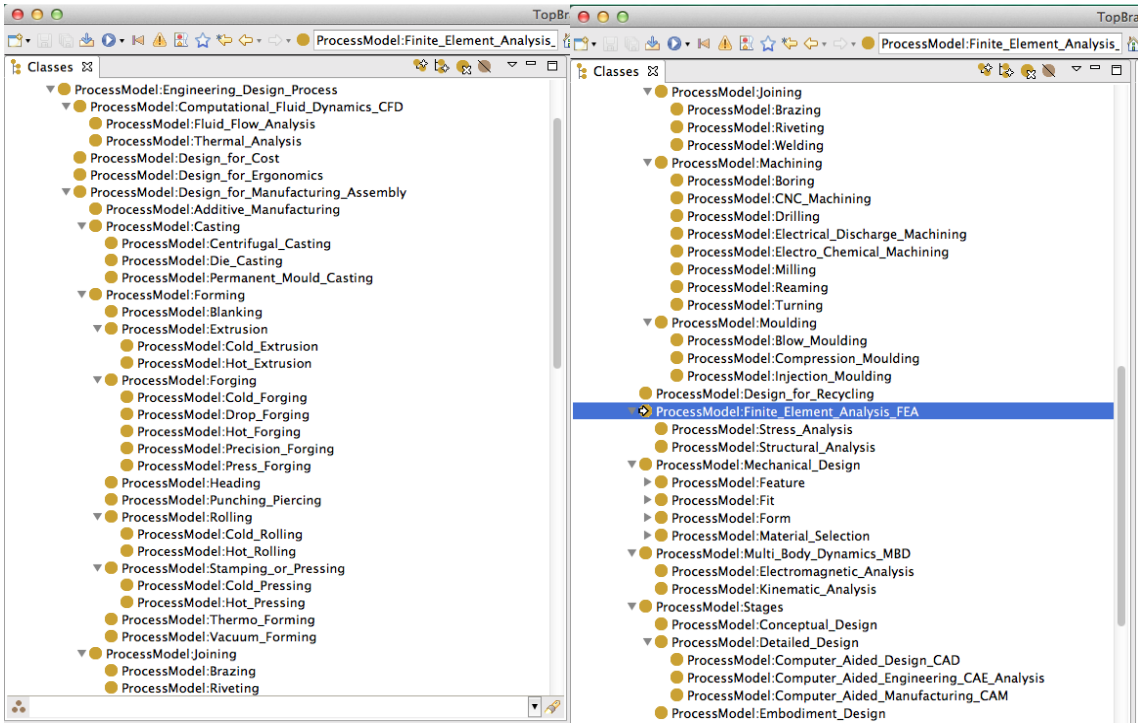


Figure 5-9: Types of Design Processes: Class Hierarchy

Similarly, for function structure, the highest level of class-subclass relationship of functional requirements for engineering design process, activities and objects has been broken down in this thesis for representation in OWL2. It is represented with the help of Figure 5-12 and 5-13.

Similarly, the object knowledge is represented as an interface to the process model with limited aspects, which can be extended further. The complete object model is illustrated with the help of Figure 5-14 and Figure 5-15.

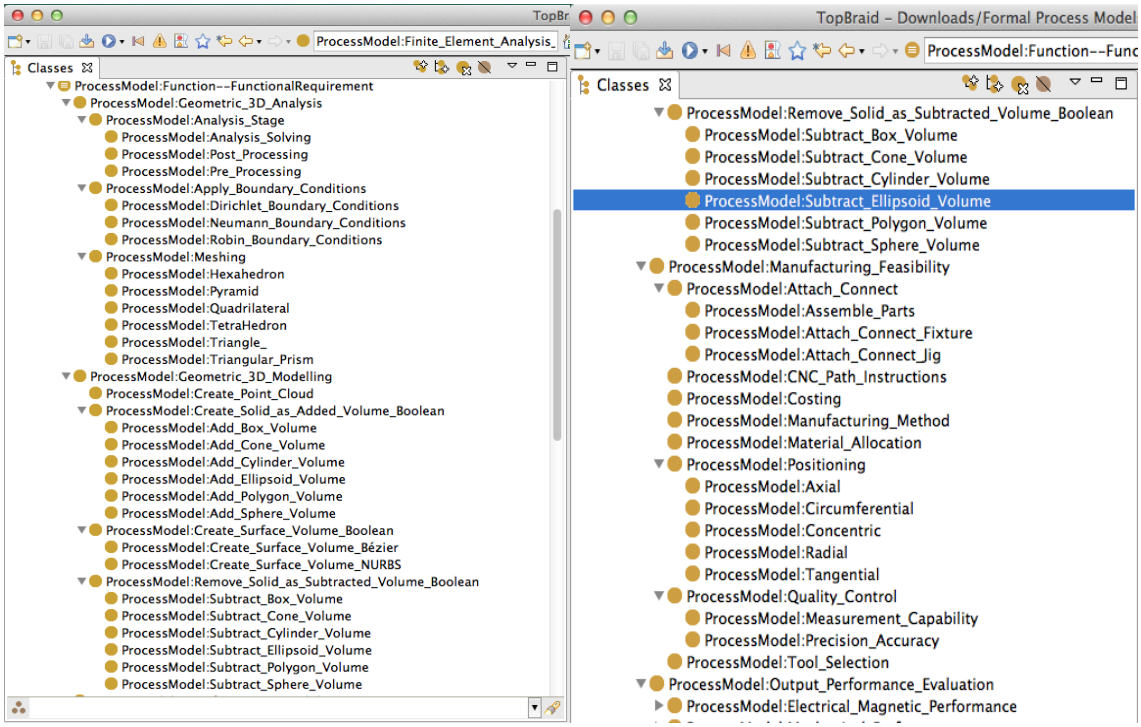


Figure 5-10: Function Structure Classification: Class Hierarchy

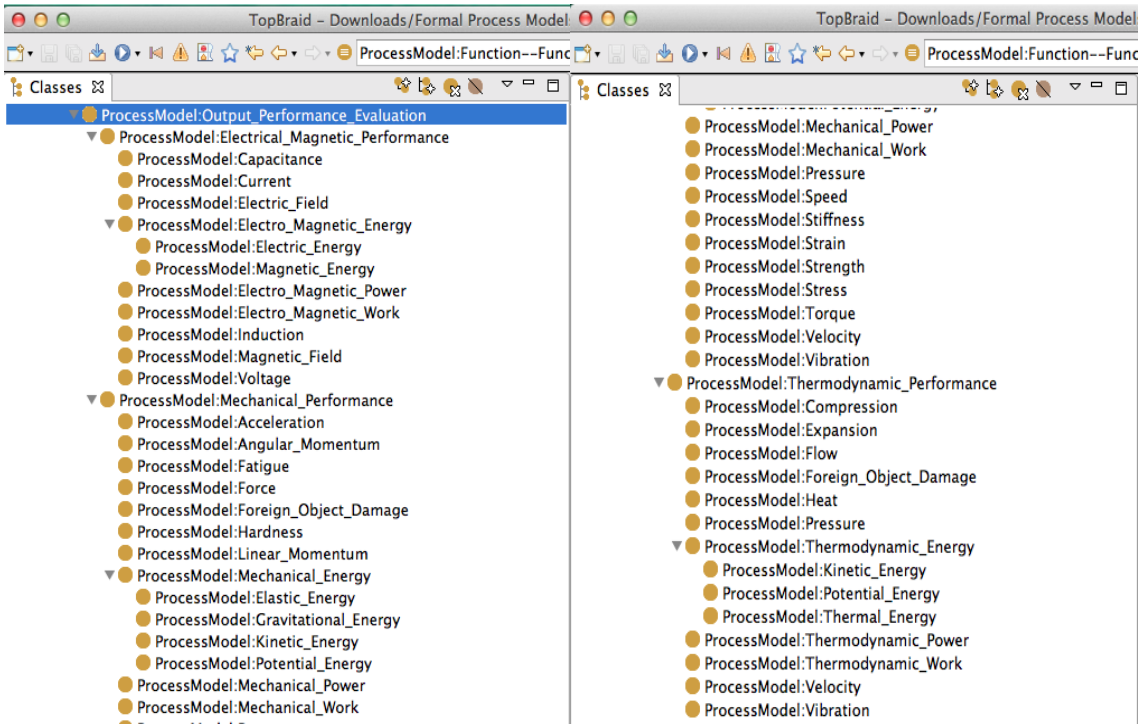


Figure 5-11: Function Structure Classification: Class Hierarchy Continued

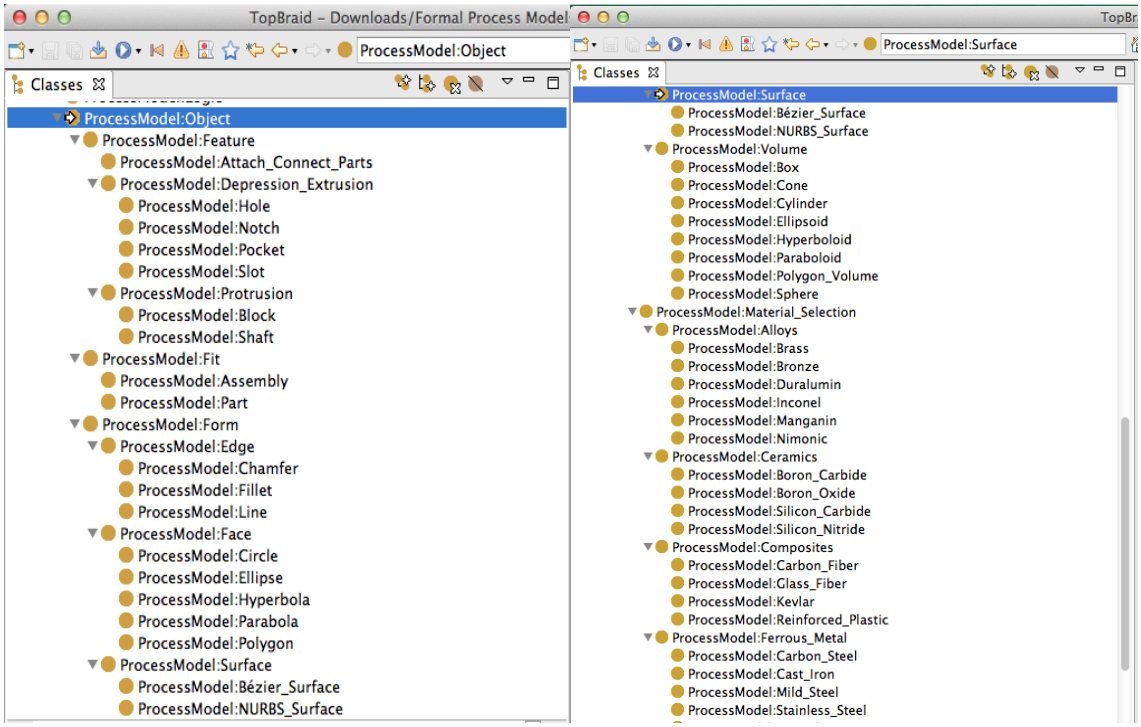


Figure 5-12: Object Model Classification: Class Hierarchy

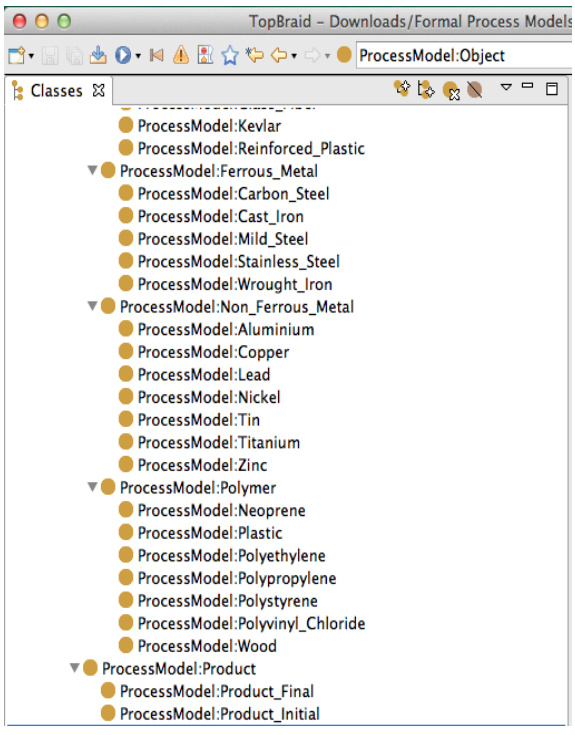


Figure 5-13: Object Model Classification: Class Hierarchy Continued

As explained earlier in section 5.3.3, in the present state of this work, the object knowledge consists of high-level classes such as features, form, fit and material selection with further sub classification as shown in Figure 5-14 and Figure 5-15.

5.6.3 Generative Modelling: Function Structure Matching using SWRL – Based on Function Structures, Design Process and Objects

In order to satisfy the ***generative modelling capability*** of DEA as summarised in section 5.5.2, following functions have been added as part of this research using SWRL for formalisation.

1. ***Generation of activities based on sub-functions*** *as functional requirements*
2. ***Generation of objects based on sub-functions*** *as functional requirements*
3. ***Generation of engineering rules for activities based on logic*** *as the basis of rules*
4. ***Assessment of initial product to generate the initial activity*** *of the process model*
5. ***Virtual and physical activity functional equivalence***

The following functions represented in SWRL on top of OWL, fulfil these 5 predefined set of generic functions to generate query and reasoning results on various instances for DEA for mechanical design with DFM process with semantic clarity and generative modelling.

4. ***Assessment of initial product to generate the initial activity*** *of the process model*

Function1: Generating 1st Activity (Physical): SWRL

Design_Process(?dp) ^ consumes_Product_Initial(?dp, ?pi) ^ Physical-Activity(?pa) ^ has_Function(?pa, ?f) ^ Assess_Product_Initial(?f) ^ Assesses(?f, ?pi) ->Starts_with_Activity(?dp, ?pa)

Function2: Generating 1st Activity (Informatical): SWRL

Design_Process(?dp) ^ consumes_Product_Initial(?dp, ?pi) ^ Informatical-Activity(?ia) ^ has_Function(?ia, ?f) ^ Assess_Product_Initial(?f) ^ Assesses(?f, ?pi) ->Starts_with_Activity(?dp, ?ia)

Function3: ***Generating 1st Activity (Virtual): SWRL***

Design_Process(?dp) ^ consumes_Product_Initial(?dp, ?pi) ^ Virtual-Activity(?va) ^ has_Function(?va, ?f) ^ Assess_Product_Initial(?f) ^ Assesses(?f, ?pi) ->Starts_with_Activity(?dp, ?va)

1. ***Generation of activities based on sub-functions*** *as functional requirements*

Function 4: Generating other Physical Activities: SWRL

Design_Process(?dp) ^ satisfies_Functional_Requirement(?dp, ?f) ^ Physical-Activity(?pa) ^ has_Function(?pa, ?f) -> consists_of_Activity(?dp, ?pa)

Function 5: Generating other Informatical Activities: SWRL

Design_Process(?dp) ^ satisfies_Functional_Requirement(?dp, ?f) ^ Informatical-Activity(?ia) ^ has_Function(?ia, ?f) -> consists_of_Activity(?dp, ?ia

Function 6: Generating other Virtual Activities: SWRL

Design_Process(?dp) ^ satisfies_Functional_Requirement(?dp, ?f) ^ Virtual-Activity(?va) ^ has_Function(?va, ?f) -> consists_of_Activity(?dp, ?va)

5. ***Virtual and physical activity functional equivalence***

Function 7: Physical and Virtual Activities Equivalent Function: SWRL

Physical-Activity(?pa) ^ has_Function(?pa, ?f) ^ Virtual-Activity(?va) ^ equivalent_to(?pa, ?va) -> has_Function(?va, ?f)

2. ***Generation of objects based on sub-functions*** *as functional requirements*

Function 8: Generating Objects: SWRL

Design_Process(?dp) ^ satisfies_Functional_Requirement(?dp, ?f) ^ Object(?o) ^ fulfills_Function(?o, ?f) -> consists_of_Object(?dp, ?o)

Function 9: Generating Object Features: SWRL

Design_Process(?dp) ^ satisfies_Functional_Requirement(?dp, ?f) ^ Feature(?fe) ^ fulfills_Function(?fe, ?f) ^ Product_Initial(?pi) ^ consumes_Product_Initial(?dp, ?pi) ^ Product_Final(?pf) -> has_Feature(?pf, ?fe) ^ produces_Product_Final(?dp, ?pf)

Function 10: Generating Object Form: SWRL

Design_Process(?dp) ^ satisfies_Functional_Requirement(?dp, ?f) ^ Form(?fo) ^ fulfills_Function(?fo, ?f) ^ Product_Initial(?pi) ^ consumes_Product_Initial(?dp, ?pi) ^ Product_Final(?pf) -> has_Form(?pf, ?fo) ^ produces_Product_Final(?dp, ?pf)

Function 11: Generating Object Fit: SWRL

Design_Process(?dp) ^ satisfies_Functional_Requirement(?dp, ?f) ^ Fit(?fi) ^ fulfills_Function(?fi, ?f) ^ Product_Initial(?pi) ^ consumes_Product_Initial(?dp, ?pi) ^ Product_Final(?pf) -> has_Fit(?pf, ?fi) ^ produces_Product_Final(?dp, ?pf)

3. ***Generation of engineering rules for activities based on logic*** *as the basis of rules*

Function 12: Generating Rules controlling Physical Activities: SWRL

Physical-Activity(?pa) ^ affectedbyLogic(?pa, ?l) ^ Rule(?r) ^ governedbyLogic(?r, ?l) -> controlled_by_Rule(?pa, ?r)

Function 13: Generating Rules controlling Informatical Activities: SWRL

Informatical-Activity(?ia) ^ affectedbyLogic(?ia, ?l) ^ Rule(?r) ^ governedbyLogic(?r, ?l) -> controlled_by_Rule(?ia, ?r)

Function 14: Generating Rules controlling Virtual Activities: SWRL

Virtual-Activity(?va) ^ affectedbyLogic(?va, ?l) ^ Rule(?r) ^ governedbyLogic(?r, ?l) -> controlled_by_Rule(?va, ?r)

Function 15: Physical and Virtual Activities Logic Equivalence: SWRL

Physical-Activity(?pa) ^ affectedbyLogic(?pa, ?l) ^ Virtual-Activity(?va) ^ equivalent_to(?pa, ?va) -> affectedbyLogic(?va, ?l)

5.7 Summary

This chapter discusses the development of ***Generative Process Model for Design Engineering Automation (GPM-DEA) as a hybrid approach of IDEF0, UML, SysML individual diagrams and addition of constructs*** as in informal/semiformal process model for DEA. The complete working of the model ***incorporates generative modelling to generate the activities, objects based on functional requirements and engineering rules based on logic for a KBE perspective***. This leads to the formalisation of GPM-DEA in OWL/SWRL ontology based on formal logic based on the method as schema mapping thus providing a method to use ontologies as neutral formal representation for DEA for mechanical design and DFM/DFA with preserved semantics. The usage of OWL/SWRL syntax and semantics constrains the meaning of its concepts and relationships through the axioms. GPM-DEA provides mechanical product design ontology with inclusion of manufacturing knowledge for DEA with a KBE approach through open standards based on the Meta model developed by the author. It can be further extended to incorporate other phases of PD such as operations and maintenance and wider aspects of DEA such as thermal design, structural design. The next chapter will discuss the test use-cases to further enhance the ontology system development for experimental verification in chapter 7.

6 Development of Knowledge Representation System with Test UseCases

6.1 Introduction

This chapter elaborates on the system development and test use-cases in the form of creating a hole in a block with the drilling process and bookshelf design process collated from literature. The use cases have been devised to provide a proof of concept working of GPM-DEA and its formal ontology implementation in OWL/SWRL as described in the previous chapters. They have been formulated around the research hypothesis as described in chapter 1 to target the DEA needs with a KBE approach. Both the use-cases have been implemented in a proprietary DEA system such as AML, ParaPy and GA based CATIA Knowledgeware and Siemens NX KF. The instantiation of GPM-DEA with its implementation in OWL/SWRL ontology for both these use-cases will be discussed in this chapter.

6.2 Overview of Use Case 3 & 4

In this thesis, concepts from Pilot Use Case 1 and 2 as Meta model partially led to the development of GPM-DEA and its system development in OWL/SWRL ontology for platform independent and neutral formal representation to enable DEA with semantic clarity for mechanical design with DFM/DFA. The automation capability includes a set of geometric and non-geometric knowledge as F-B-S aspects of mechanical design process with DFM/DFA for automation. GPM-DEA as a coherent and structured process based knowledge model provides a schema or a method as a Meta model for ontology development as neutral formal representation for DEA.

The test use-cases compiled and analysed in this work are targeted to refine the implementation of GPM-DEA in OWL/SWRL ontology for incorporation of product's geometric attributes with numeric values. The working of GPM-DEA with functional

requirements as the basis for generative modelling has been discussed in section 5.5.2 and its method of implementation in ontologies in section 5.6.3. The automation capability varies from sub-function structures at conceptual design stage to generation of activities, objects and engineering rules to show the effect of the process model on the product's geometric attributes at the detailed design stage.

As the pilot use cases with experimentation for formalisation as discussed in section 4.4 and 4.5 catered primarily to the conceptual and configuration / embodiment design stage with DFM, the test use-cases in the next section have been developed to target the detailed design stage with inclusion of DFM as manufacturing knowledge with datatype float numeric values for product's attributes. Both the test use-cases have been devised and implemented in OWL/SWRL ontology with the method of schema mapping as developed in section 5.5 in this work. The allocation of use-case is illustrated in Figure 6-1.

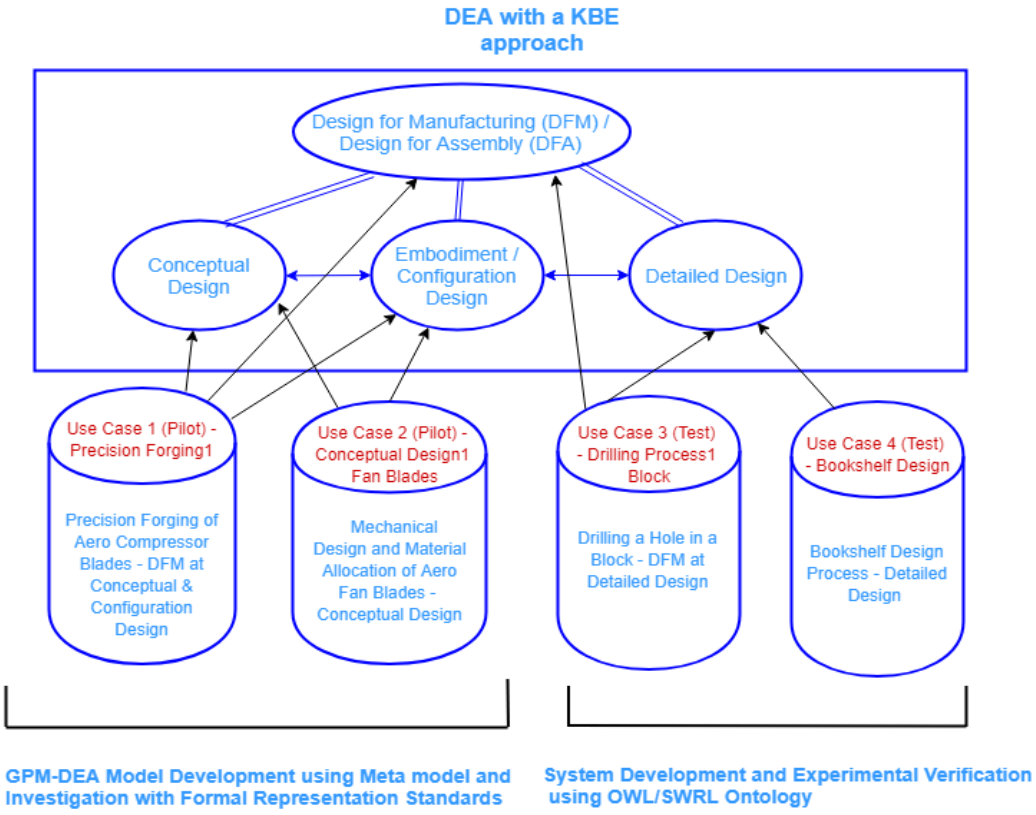


Figure 6-1: Use Case Allocation – Created by Author

Use Case 3, as compiled by the author focusses on creating a hole in a block with drilling process such that the block can be described with numeric values of geometric features. It will be used for system development on OWL/SWRL ontology as per the devised GPM-DEA schema or Meta model and is implemented in ParaPy as a KBE based DEA tool. Similarly, Use Case 4 is based on a bookshelf design process with numeric values to geometric features. This will be instantiated in GPM-DEA for formalisation in OWL/SWRL ontology. This has also been implemented in AML, CATIA knowledgeware and Siemens NX KF. Both the use – cases have been implemented in ontology as per the ontology development methodology discussed in research design in section 1.4.2. Use case 3 has been devised with motivation from (Hunter et al., 2005; Monfared, 2000) and the understanding of the research scope. Use Case 4 has been devised and adopted from (Lützenberger et al., 2012) from the LinkedDesign project whose focus is on KBE based automation with platform independent and neutral formal representation of engineering design knowledge. Knowledge has been added to the Use Case 4 by the author in terms of functions for individual activities and logic description for rules such that the generative modelling capability developed as part of this research can be illustrated.

6.3 Test Use Case 3: Creating a Hole in a Block with Drilling Process

The aim of this use case is to refine the system development as OWL/SWRL using the GPM-DEA schema as Meta Model and at the instance level with incorporation of product geometric accessible attributes as block dimensions in this work. The DEA process initiates from the sub-function structures and function mapping of activities and objects to that of the engineering design process through to the generation of rules to control the drilling process with its effect on block attributes. The following questions arise which will be verified in the next chapter –

1. *Can the instances of drilling process in a block for creating a hole be automatically generated based on function structures of individual activities such as drilling, reaming along with objects such as drill bit and engineering rules for controlling the effect on geometric attributes of the block?*
2. *Can the implementation (ontology and rule representation) of the generated activities, objects and rules generate appropriate and accurate numeric values to block attributes thus successfully enabling DEA with a KBE approach?*

The instantiation of GPM-DEA as informal/semiformal knowledge capture with Use case 3 with its formalisation in OWL/SWRL ontology as system development for formal representation is discussed in this section.

6.3.1 Function Structure Matching

As observed from Figure 6-2, the drilling process1 has 2 sub-functions – 'Cut hole of circular cross section' and 'Precision of hole dimensions'. It can be observed from Figure 6-3, all activities such as drill hole, ream hole, bore hole and punch hole satisfy function – 'Cut hole of circular cross section'. Similarly, the activity 'Set requirements of hole' satisfies the function – 'Precision of hole dimensions'. Thus drilling process1 can have all of these activities in the form of drill hole, ream hole, bore hole and punch hole along with 'Set requirements of hole'.

Engineering Design Process	Engineering Design Process Functional Requirement	Sub-Functions
Drilling Process1	<Drilling> is a <cutting process> that uses a <drill bit> to <cut a hole of circular crosssection> in <solid materials>. The <hole> can be of various dimensions depending upon the drill bit dimensions	Cut_hole_of_circular_cross_section + Precision_of_hole_dimensions
Precision Forging1	<Achieve an accuracy of +/-1mm in shape prediction>, as shape prediction accounts for a bulk of the manufacture objectives.	Achieve an accuracy of +/-1mm + shape prediction
Conceptual Design1 Fan Blades	The fan blades <spin to accelerate a mass of air> into the engine to <generate thrust> that propels the aircraft forward. Fan blades also function to <reduce total engine damage> from the <ingestion of various foreign objects> such as birds by radially deflecting outward such objects rather than passing them through to the core parts of the engine. The <dovetail attachment> of fan blades are used to <secure the blades to the hub or disk>. It is important to allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength	Generate thrust + reduce total engine damage from the ingestion of various foreign objects such as birds + secure the blades to the hub or disk + allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength

Figure 6-2: Drilling Process Functional Requirements & Sub Functions: Knowledgebase

Engineering Design Process Activity (Physical/Informatical)	Equivalent Virtual Activities	Functional Requirement --[Sub-Functions]
Create block (workpiece)	Protrude block (workpiece)	Manufacture block
Drill hole	Create hole + Subtract hole	Cut hole of circular cross section
Ream hole	Create hole + Subtract hole	Cut hole of circular cross section + enlarge hole + high surface finish
Bore hole	Create hole + Subtract hole	Cut hole of circular cross section + enlarge hole
Selection and positioning of reamer		
Set requirements of hole		Precision_of_hole_dimensions
Selection and positioning of drill bit		
Punch hole	Create hole + Subtract hole	Cut hole of circular cross section + shear force
Extrusion		Extruded stem long enough for the part to be handled in subsequent operations + base of the extruded part needs to have enough material for subsequent heading + shape prediction
Stamping		Enough energy in the top die to achieve the movement required to reduce the gap between the dies to the sum of the gap between the flash lines and the minimum running thickness + shape prediction
Heading		The heading cavity is filled out as much as possible even if the cavity does not get filled in its entirety + shape prediction
Blade Geometry Optimisation		Fan blades spin to accelerate a mass of air into the engine + generate thrust + 80% of the thrust delivered by fan + reduce total engine damage from the ingestion of various foreign objects such as birds
Dovetail Attachment		Secure the blades to the hub or disk
Material Selection		Allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength

Figure 6-3: Activities with Functions & Sub Functions: Knowledgebase

All the activities with their inputs, outputs, controls as rules and mechanisms as resources along with participating objects as inputs is shown in Figure 6-4. Their corresponding graphical representation is an ICOM box of IDEF0 standard in GPM-DEA as illustrated in next section 6.3.2. A snapshot of the rules controlling the activities is shown in Figure 6-5.

Engineering Design Process Activity (Physical/Informatical/ Virtual)	Objects - Product as primary object	Activity Inputs	Activity Outputs	Controls: Engineering Rules	Resources
Create block (workpiece)	Product - Block	Width (W), Height (H), Material (M)	Value assigned to W, H, M, Depth (D)	Dimension Rule, Depth Rule, Volume Rule, Material Rule	Fixture
Protrude block (workpiece)	Product - Block	Width (W), Height (H), Material (M)	Value assigned to W, H, M, Depth (D)	Dimension Rule, Depth Rule, Volume Rule, Material Rule	Fixture
Drill hole	Any instance of Product, Object - Drill Bit	Drill Bit Position Coordinates (A,B,C), Drill bit axis (C1), Drill bit diameter (DD), Drill Flute length (F)	Hole created	Hole Diameter Rule, Hole Depth Rule, Drill Location Rule, Drill Diameter Rule, Drill Length Rule	Fixture, Cutting fluid
Create hole	Any instance of Product	Hole position coordinates (X,Y,Z), Hole axis (C2), Hole Diameter (HD1), Hole depth (HD2)	Value assigned to X,Y,Z, C2, HD1, HD2	Hole Diameter Rule, Hole Depth Rule, Hole Volume Rule	Design Tool
Ream hole	Any instance of Product, Object - Reamer	Reamer Position Coordinates (A,B,C), Reamer axis (C1), Reamer diameter (DD), Reamer Flute length (F)	Hole created	Hole Diameter Rule, Hole Depth Rule, Reamer Location Rule, Reamer Diameter Rule, Reamer Length Rule	Fixture, Cutting fluid
Bore hole	Any instance of Product, Object - Boring mill machine	Boring mill Position Coordinates (A,B,C), Boring mill axis (C1), Boring mill tool diameter (DD), Boring mill tool cutting length (F)	Hole created	Hole Diameter Rule, Hole Depth Rule, Boring mill tool Location Rule, Boring mill tool Diameter Rule, Boring mill tool Length Rule	Fixture, Cutting fluid
Punch hole	Any instance of Product, Object - Punch	Punching tool Position Coordinates (A,B,C), Punching tool axis (C1), Punching tool diameter (DD), Punching tool cutting length (F)	Hole created	Hole Diameter Rule, Hole Depth Rule, Punching tool Location Rule,Punching tool Diameter Rule,Punching tool Length Rule	Fixture, Cutting fluid

Figure 6-4: Activities with Inputs, Outputs, Rules and Resources with Objects for Drilling Process: Knowledgebase

Controls: Engineering Rule	Engineering Rule Description	Type of Rule	Logic as basis of Rule
Dimension Rule	Minimum dimensions of the block is 50 mm, W>=50mm, H>=50 mm, D>=50mm)	Logic Rule, Math Rule, Geometry Rule	Analyse block for hole to be drilled
Volume Rule	V1=W*H*D	Logic Rule, Math Rule, Geometry Rule	Volume of solid material
Material Rule	If W>100 Then M = Metallic_Aluminium)	Heuristic Rule, Math Rule, Geometry Rule	
Depth Rule	D=W*1.5	Heuristic Rule, Math Rule, Geometry Rule	
Hole Depth Rule	Hole depth should be less than or equal to depth of block, HD2<=D	Logic Rule, Math Rule, Geometry Rule	Hole Depth Analysis
Hole Diameter Rule	HD1*1.25<W, HD1*1.25<H	Logic Rule, Math Rule, Geometry Rule	Hole Diamater Analysis
Hole Volume Rule	Volume of Hole (VH) = [(3.14*HD1*HD1)/4]*HD2)]	Logic Rule, Math Rule, Geometry Rule	Volume of solid material
Volume2 Rule	Final Volume (V2) = V1-HV)		
Drill Location Rule	Drill bit position coordinates (A,B,C) should coincide with hole position coordinates on the face (X,Y,Z), drill bit axis (Axis C1) should coincide with hole aixs (Axis C2)	Logic Rule, Configuration Rule	Drill bit equivalence
Drill Diameter Rule	Drill bit diameter (DD) should be equivalent to hole diameter (HD1), DD=HD1	Logic Rule, Configuration Rule	Drill bit equivalence
Drill Length Rule	Flute length (F) of drill bit should be greater than or equal to hole depth (HD2), F>=HD2)	Logic Rule, Configuration Rule	Drill bit equivalence
Process Rule1	If <Tolerance of the hole is less than 0.2 mm for high accuracy>perform reaming else drilling	Process Rule	

Figure 6-5: Engineering Rules controlling the Design Process Activities for Drilling Process: Knowledgebase

6.3.2 Informal / Semiformal Representation: GPM-DEA

The author has devised and instantiated an instance of drilling process in GPM-DEA as informal / semiformal representation as shown in Figure 6-6. Both physical and virtual activities are modelled as equivalent activities. For example physical activity ‘Assess block (workpiece)’ is equivalent to the virtual activity ‘Assess Protruded block (workpiece)’. Similarly, ‘Drill hole’ as a physical activity is equivalent to virtual activities – ‘Create hole’ and ‘Subtract hole’. ***As all activities are represented using IDEF0 notation*** for functional modelling, equivalent activities correspond to same function. The SWRL Function 4 developed in this work, discussed in section 5.5.3, executes the equivalency as neutral formal representation.

All activities are represented with inputs, controls as rules, outputs and mechanisms as resources (ICOM). The process rule for selection between drilling and reaming process based on tolerance of hole is represented with ***UML condition link***. Thus process-sequencing options are represented with red links with UML condition link for multiple ‘what-if’ scenarios. Sub-activity in the form of selection and positioning of drill bit as represented using blue link. The product model representing the initial and the final state as block and block with hole respectively is represented using UML class diagram with attributes and

engineering rules as methods affecting the attributes. ***UML class diagram*** has been used for representing all participating objects in the form of drill bit. ***SysML requirement diagram*** is used for representing the functional requirements of the drilling process as sub functions in context to the block and the drill bit as objects.

Figure 6-7 shows a snapshot of function matching of individual activities as function structures of drilling process functional requirements with links to rules in ICOM box. Figure 6-8 shows a snapshot of product as block in initial state and final state as block with hole with UML class diagram along with function matching of these objects. As it can be observed, various engineering rules such as ‘dimension, depth, material, hole depth and hole diameter rule’ are informally represented as methods inside UML class diagram along with attributes by the author. It also shows the interface of the process model to the product model.

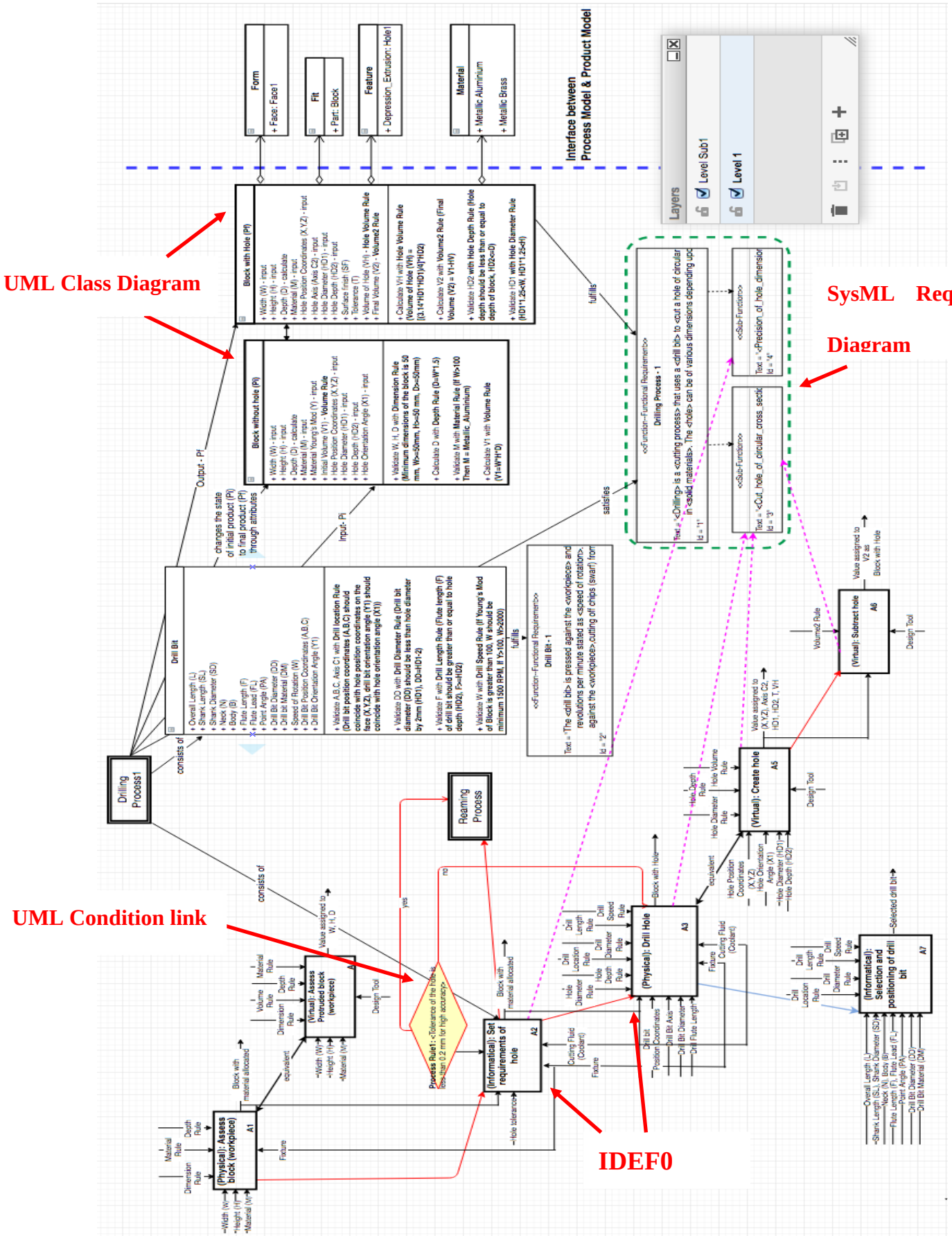


Figure 6-6: An Instance of Drilling Process in GPM-DEA: Informal / Semiformal Representation

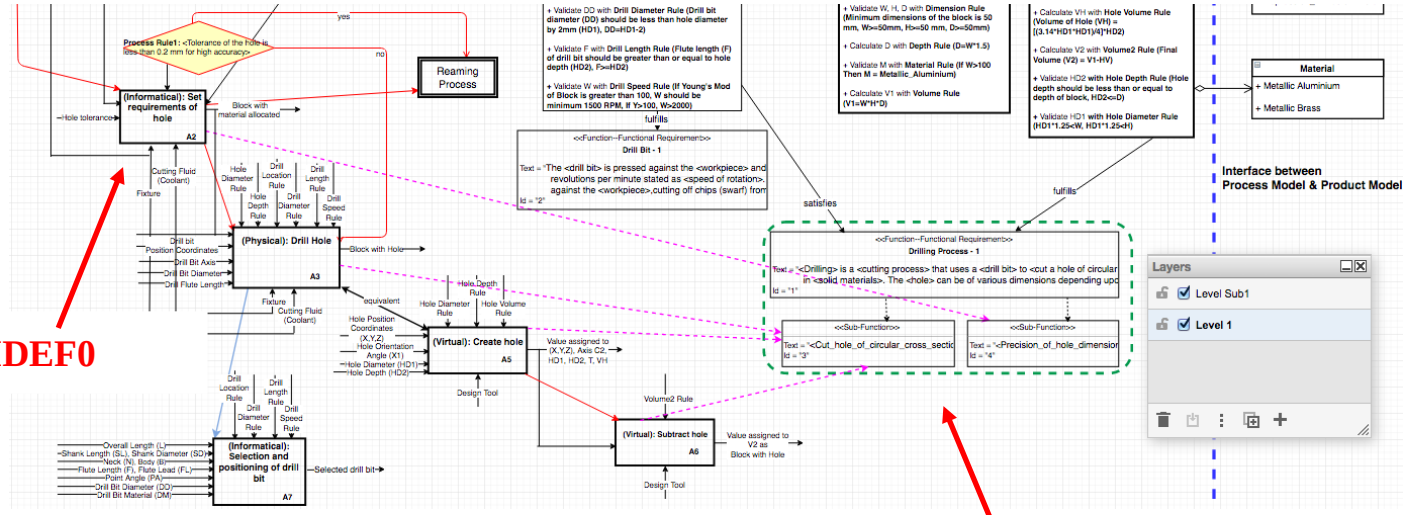


Figure 6-7: Function Structure Matching – Drilling Process Activities with Links to Rules

SysML Requirement Diagram

UML Class Diagram

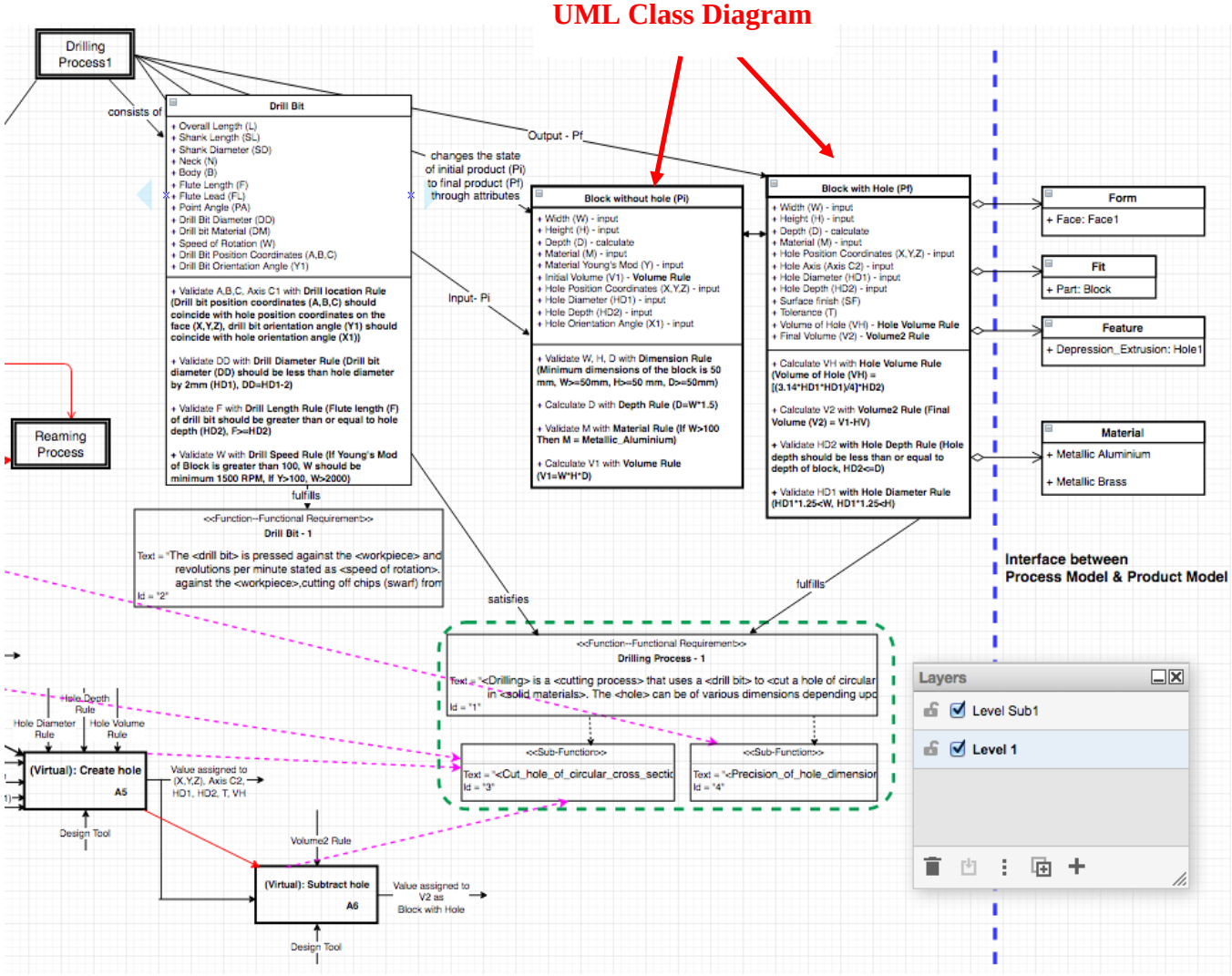


Figure 6-8: Product in Initial and Final State and Function Matching - Drilling Process Objects with Description of Rules

6.3.3 Formal Representation: OWL/SWRL

The instantiated GPM-DEA model for drilling process has been then represented in OWL/SWRL ontology as neutral formal representation by the author. The activities of the drilling process in GPM-DEA corresponding to the graphical representation in Figure 6-6and 6-7 as IDEF0 ICOM box are represented formally in OWL2 with associated id, inputs, outputs, resources and linkage to engineering rules using classes and properties. All the activities interlinked with product structure with attributes, function as sub-functions and behaviour corresponding to UML class diagram and SysML requirement diagram are also represented using OWL2 using classes and properties as illustrated in Figure 6-9.

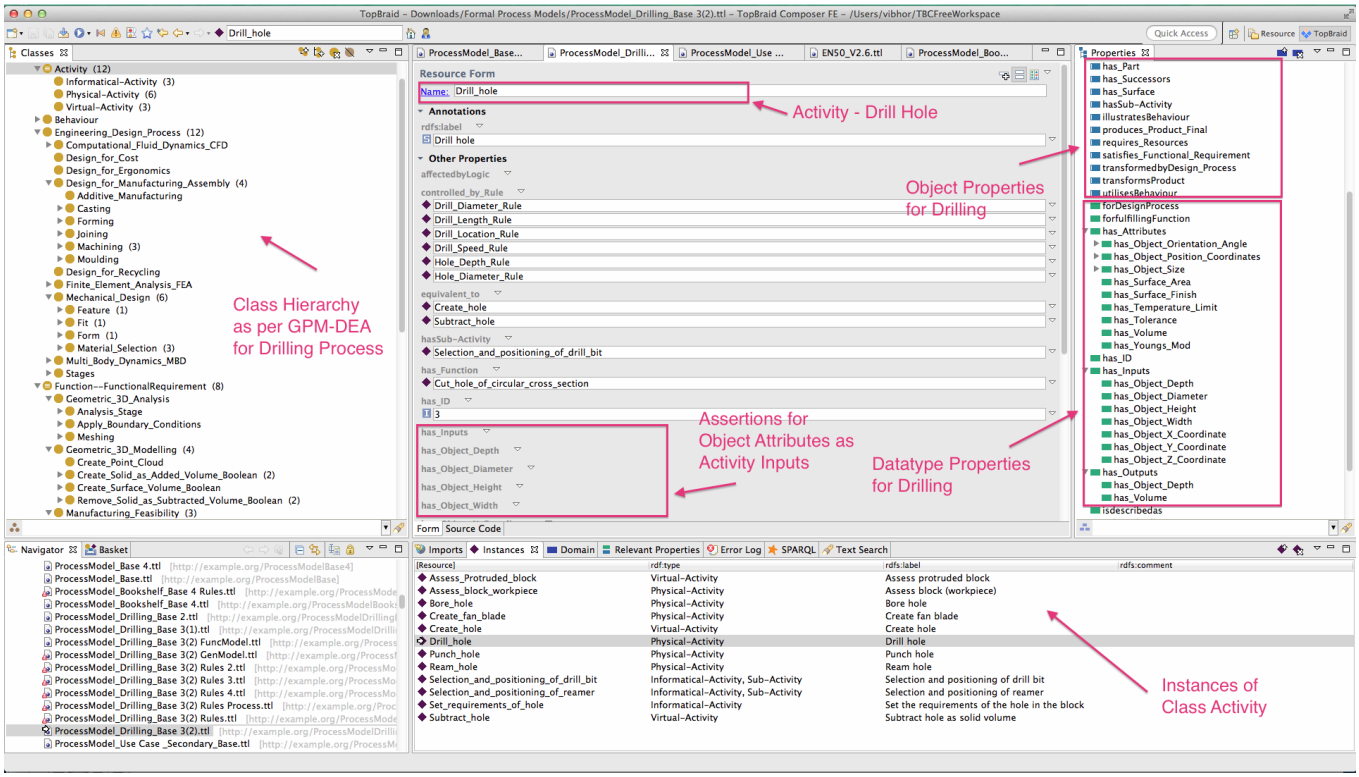


Figure 6-9: Drilling Process in OWL: TopBraid Composer FE

As observed from Figure 6-9, only OWL2 representation is utilised in Topbraid. All the classes with class-subclass relationship can be observed on the left. All the properties for inputs, outputs and other relationships marked as arrows in Fig 6-6and 6-7 are represented on the right under the properties tab in Figure 6-9. Instances have been produced on the bottom

and all the relationships between instances can be observed in the resource form in the centre tab. As also observed from Figure 6-9, the drill hole instance of activity tab illustrates its ID as 3, is controlled by rules such as Drill_Diameter_Rule, Drill_Length_Rule and Hole_Diameter_Rule. The equivalent activities and functions are also asserted using specified property in the form of 'has_Function'.

For the product attributes in UML class diagram as activity inputs and outputs as IDEF0 ICOM, in context to GPM-DEA, datatype properties have been created and instantiated in this work. As explained in section 5.5.1, (ProcessModel:has_Inputs) and (ProcessModel:has_Outputs) are the datatype properties created in GPM-DEA to assert activity inputs and outputs in terms of object attributes. All sub-properties of (ProcessModel:has_Attributes) such as (ProcessModel:has_Object_Size), (ProcessModel:has_Object_Position_Coordinates) and (ProcessModel:has_Object_Orientation_Angle), (ProcessModel:has_Volume) can be asserted as sub properties of (ProcessModel:has_Inputs) and (ProcessModel:has_Outputs). As observed from Figure 6-9, following properties as sub properties of (ProcessModel:has_Attributes) have been classified as sub properties of (ProcessModel:has_Inputs) as activity inputs –

I. ProcessModel:has_Object_Depth
II. ProcessModel:has_Object_Diameter
III. ProcessModel:has_Object_Height
IV. ProcessModel:has_Object_Width
V. ProcessModel:has_Object_X_Coordinate
VI. ProcessModel:has_Object_Y_Coordinate
VII. ProcessModel:has_Object_Z_Coordinate

Similarly, the following properties as sub properties of (ProcessModel:has_Attributes) have been classified as sub properties of (ProcessModel:has_Outputs) as activity outputs –

I. ProcessModel:has_Object_Depth

II. ProcessModel:has_Volume

It can be observed that the same property (ProcessModel:has_Object_Depth) has been classified under both (ProcessModel:has_Inputs) and (ProcessModel:has_Outputs) thus making the model flexible. UML class diagram attributes can thus be neutrally represented using OWL2 datatype properties. As illustrated in Figure 6-9, all the properties can be observed on the right tab.

However, as explained earlier, due to the limitations of OWL in representing n-ary relationships, generative modelling capabilities of GPM-DEA based on the functional requirements as sub function structures along withthe methods in UML class diagram as engineering rules based on logic and math the can't be represented using OWL2. These have been formally represented using SWRL in this research.

In GPM-DEA, it has been illustrated that IDEFO activities have function. Similarly, an object fulfills a function, and the design process satisfies functional requirement in the form of product function. As per the class-subclass relationship of function structures discussed in section 6.3.1, the sub function - 'Cut hole of circular cross section' is an instance of class 'Remove_Solid_as_Subtracted_Volume_Boolean' as a subclass of 'Geometric_3D_Modelling'. It further becomes an instance of class 'Subtract_Cylinder_Volume' at the lowest level. The sub function - 'Precision of hole dimensions' is an instance of class 'Precision_Accuracy' as a subclass of 'Quality_Control', which further is a subclass of 'Manufacturing_Feasibility'. Various instances of functions are shown with the help of Figure 6-10.

Other functions such as – ‘Enlarge_hole’ and ‘High_surface_finish’ have been allocated as instances of function-sub function class hierarchy as shown in Figure 6-10.

The SWRL functions for generative modelling capabilities of GPM-DEA using function structure matching in this research have been illustrated in section 5.5.3.Protégé offers a built in plugin for SWRL.

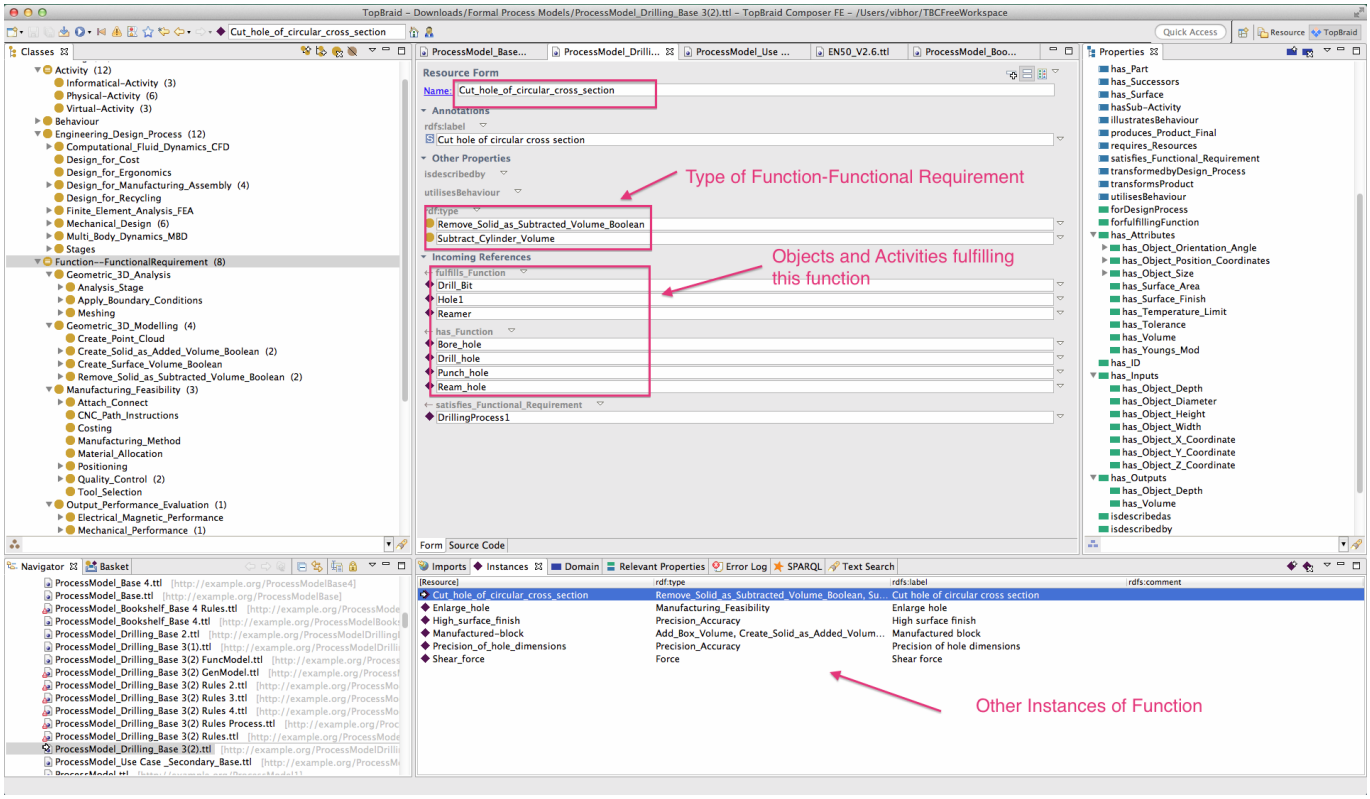


Figure 6-10: Instances of Functions – Drilling: Topbraid

The implementation of the SWRL generative modelling functions for drilling is illustrated with the help of Figure 6-11. The URI in the form of ProcessModel: was removed automatically while importing the turtle (.ttl) file from Topbraid to Protégé.

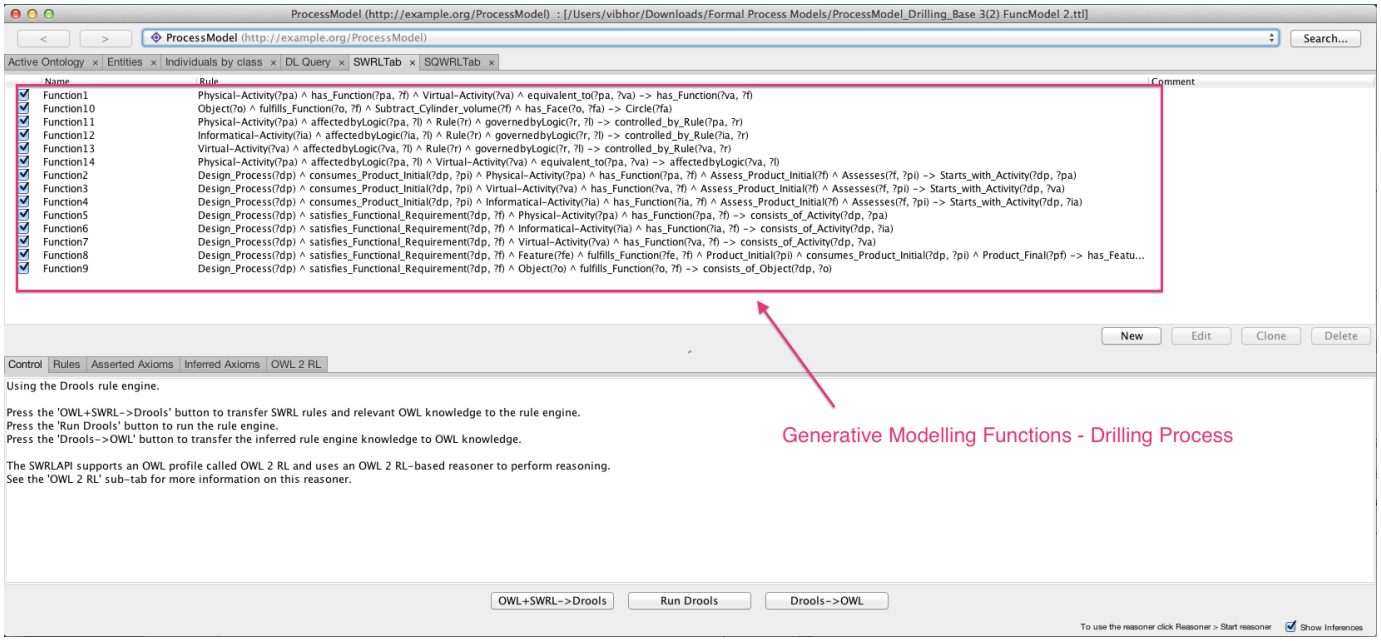


Figure 6-11: SWRL Functions - Generative Modelling in Drilling: Protégé

It is important to note that Function 10 has been specifically added and tailored to the drilling process to reflect the change in state of block through extrusion as subtracted volume with a constraint on created ontology that if the volume subtracted is a cylinder than the face of the block should be circular. Figure 6-11 illustrates the representation or codification of functions, which allow GPM-DEA to generate activities and objects based on the sub functions of each activity and the object along with rules based on logic.

The verification of the generative modelling capability of GPM-DEA through drilling use-case will be discussed in next chapter by testing the reasoning capability of the drools reasoner on SWRL axioms and SQWRL query language. For the instantiated drilling use case example in GPM-DEA, the initial product is the block and the final product is block with hole as feature after the drilling process has been performed. Multiple holes can be created as instances of Hole class as a subclass of Depression_Extrusion feature. SPARQL query for activity to function mapping for ‘Drill_hole’ and ‘Ream_hole’ activities is illustrated with Figure 6-12 and 6-13.

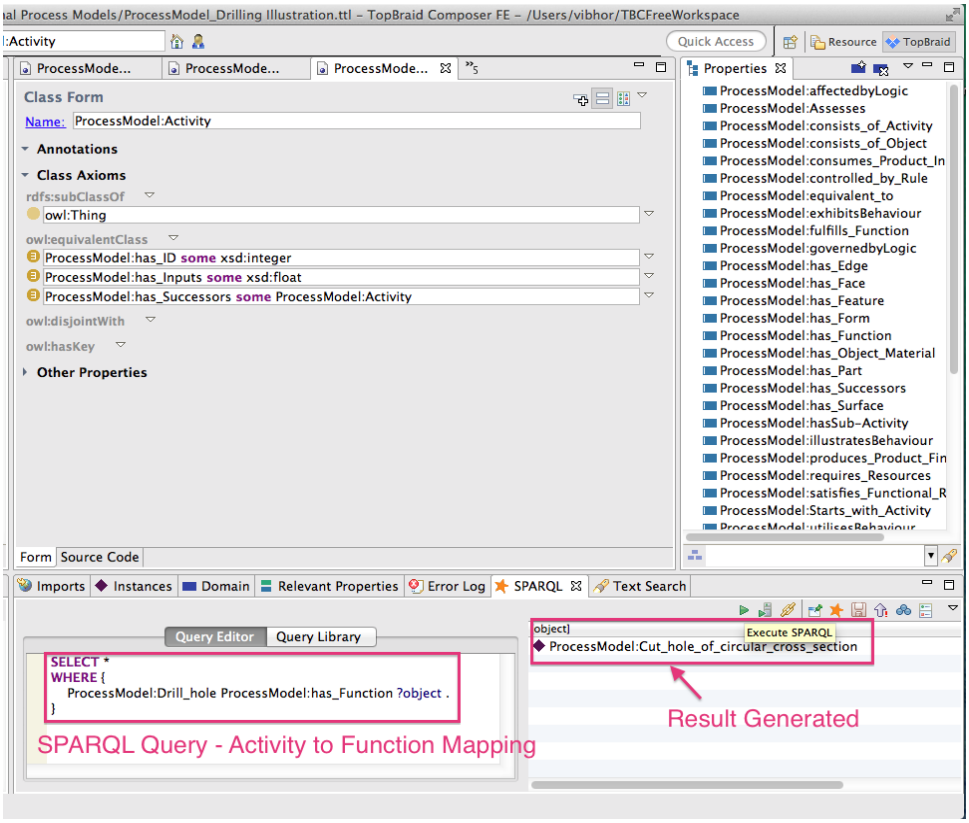


Figure 6-12: SPARQL Query Result: Activity to Function Mapping – Drill hole

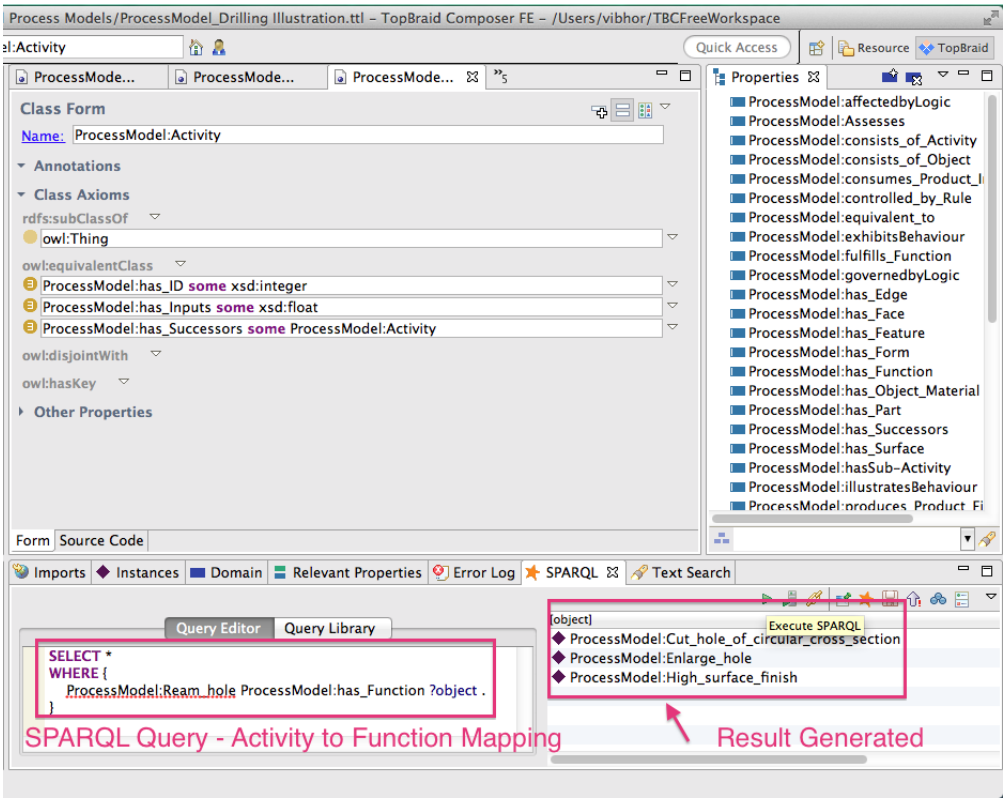


Figure 6-13: SPARQL Query result: Activity to Function Mapping – Ream hole

The function – ‘Cut_hole_of_circular_cross_section’ is a function of type ‘Subtract_Cylinder_Volume’, ‘Enlarge_hole’ belongs to Function of type ‘Manufacturing_Feasibility’ and ‘High_surface_finish’ is a function of type ‘Precision_Accuracy’. All the function structures of GPM-DEA have been elaborated in Figure 5-12 and 5-13 in section 5.5.2.

Similarly, SPARQL query for object to function mapping for physical objects – ‘Drill Bit’ and ‘Reamer’ is shown with Figure 6-14 and rule to logic description mapping with Figure 6-15.

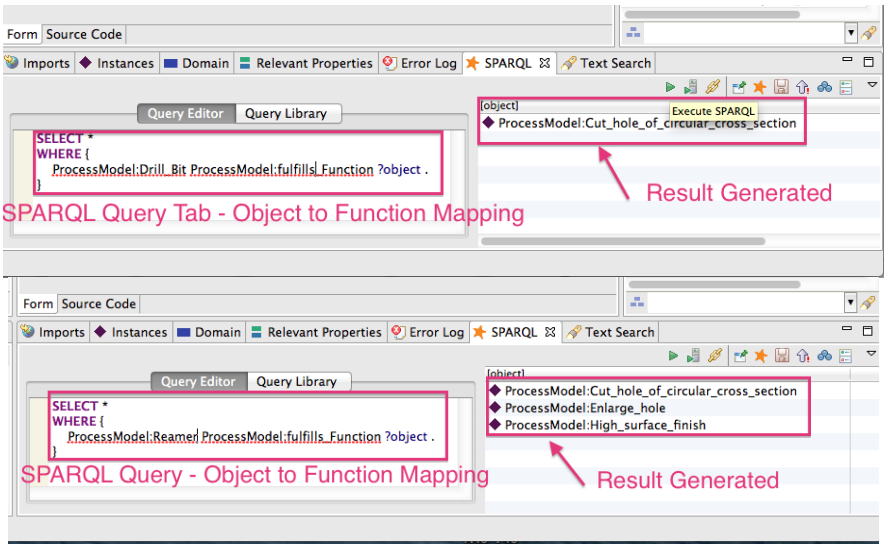


Figure 6-14: SPARQL Query Result – Object to Function Mapping – Drill bit and Reamer

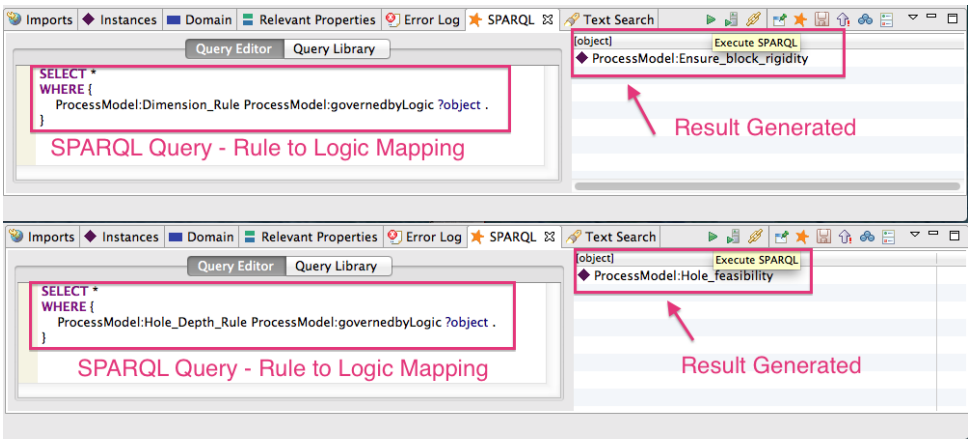


Figure 6-15: SPARQL Query Result – Rule to Logic Mapping – Drilling Process

The engineering rules represented informally in GPM-DEA as methods in UML have also been codified using SWRL as neutral formal representation in this research. From the knowledgebase, shown in Figure 6-5, consisting of engineering rules for the drilling use case, all the rules in SWRL axioms are illustrated as follows –

1. ***SWRL Dimension Rule*** - Minimum dimensions of the block is 50 mm, W>=50mm, H>=50 mm, D>=50mm)

   SWRL Representation - Product(?p) ^ hasWidth(?p, ?w) ^ swrlb:greaterThanOrEqual(?w, "50.0"^^xsd:float) ^ hasHeight(?p, ?h) ^ swrlb:greaterThanOrEqual(?h, "50.0"^^xsd:float) ^ hasDepth(?p, ?d) ^ swrlb:greaterThanOrEqual(?d, "50.0"^^xsd:float) -> sqwrl:select("Block adheres to dimensions")

2. ***SWRL Depth Rule*** - D=W*1.5

   SWRL Representation - Product(?p) ^ hasWidth(?p, ?w) ^ swrlb:multiply(?x, ?w, "1.5"^^xsd:float) -> hasDepth(?p, ?x)

3. ***SWRL Material Rule*** - If W>100 Then M = Metallic_Aluminium)

   SWRL Representation - Product(Block) ^ hasWidth(Block, ?w) ^ swrlb:greaterThan(?w, "100.0"^^xsd:float) -> hasMaterial(Block, Metallic_Aluminium)

4. ***SWRL Volume Rule*** - V1=W*H*D

   SWRL Representation - Product(?p) ^ hasWidth(?p, ?w) ^ hasHeight(?p, ?h) ^ hasDepth(?p, ?d) ^ swrlb:multiply(?v, ?w, ?h, ?d) -> hasVolume(?p, ?v)

5. ***SWRL Hole Depth Rule*** - Hole depth should be less than or equal to depth of block, HD2<=D

   SWRL Representation - Product(?p) ^ hasDepth(?p, ?d) ^ Hole(?h) ^ hasDepth(?h, ?d2) ^ swrlb:lessThanOrEqual(?d2, ?d) -> sqwrl:select(("Hole adheres to dimensions")
   Else
   Product(?p) ^ hasDepth(?p, ?y) ^ Hole(?h) ^ hasDepth(?h, ?z) ^swrlb:greaterThan(?z, ?y) -> sqwrl:select("Hole can't be created")

6. ***SWRL Hole Diameter Rule*** - HD1*1.25<W, HD1*1.25<H

SWRL Representation - Product(?p) ^ hasWidth(?p, ?a) ^ hasHeight(?p, ?b) ^ Hole(?h) ^ hasDiameter(?h, ?c) ^swrlb:multiply(?d, ?c, "1.25"^^xsd:float) ^ swrlb:lessThan(?d, ?a) ^ swrlb:lessThan(?d, ?b) ->sqwrl:select("Hole adheres to dimensions")

Else

Product(?p) ^ hasWidth(?p, ?e) ^ Hole(?h) ^ hasDiameter(?h, ?g) ^ swrlb:multiply(?i, ?g, "1.25"^^xsd:float) ^ swrlb:greaterThanOrEqual(?i, ?e) -> sqwrl:select("Hole can't be created")

Else

Product(?p) ^ hasHeight(?p, ?f) ^ Hole(?h) ^ hasDiameter(?h, ?g) ^ swrlb:multiply(?i, ?g, "1.25"^^xsd:float) ^ swrlb:greaterThanOrEqual(?i, ?f) -> sqwrl:select("Hole can't be created")

7. ***SWRL Hole Volume Rule*** - Volume of Hole (VH) = [(3.14*HD1*HD1)/4]*HD2)]

SWRL Representation - Hole(?h) ^ hasDiameter(?h, ?hd1) ^ hasDepth(?h, ?hd2) ^ swrlb:multiply(?x, "3.14"^^xsd:float, ?hd1, ?hd1, ?hd2) ^ swrlb:divide(?vh, ?x, "4.0"^^xsd:float) -> hasVolume(?h, ?vh)

8. ***SWRL Volume2 Rule*** - Final Volume (V2) = V1-HV)

SWRL Representation - Product_Initial(?p) ^ hasVolume(?p, ?v1) ^ Product_Final(?p2) ^ hasFeature(?p2, ?i) ^ Depression(?i) ^ hasVolume(?i, ?v2) ^ swrlb:subtract(?j, ?v1, ?v2) -> hasVolume(?p2, ?j)

9. ***SWRL Process Rule1*** - If <Tolerance of the hole is less than 0.2 mm for high accuracy>perform reaming else drilling

SWRL Representation - Activity(Set_requirements_of_hole) ^ Hole(?h) ^ has_Tolerance(?h, ?t) ^ swrlb:lessThan(?t, "0.2"^^xsd:float) -> has_Successors(Set_requirements_of_hole, ReamingProcess)

Else

Activity(Set_requirements_of_hole) ^ Hole(?h) ^ has_Tolerance(?h, ?t) ^ swrlb:greaterThan(?t, "0.2"^^xsd:float) -> has_Successors(Set_requirements_of_hole, Drill_hole)

Figure 6-16, Figure 6-17 and Figure 6-18 illustrate the SWRL representation of engineering rules for the drilling use case in Protégé IDE.

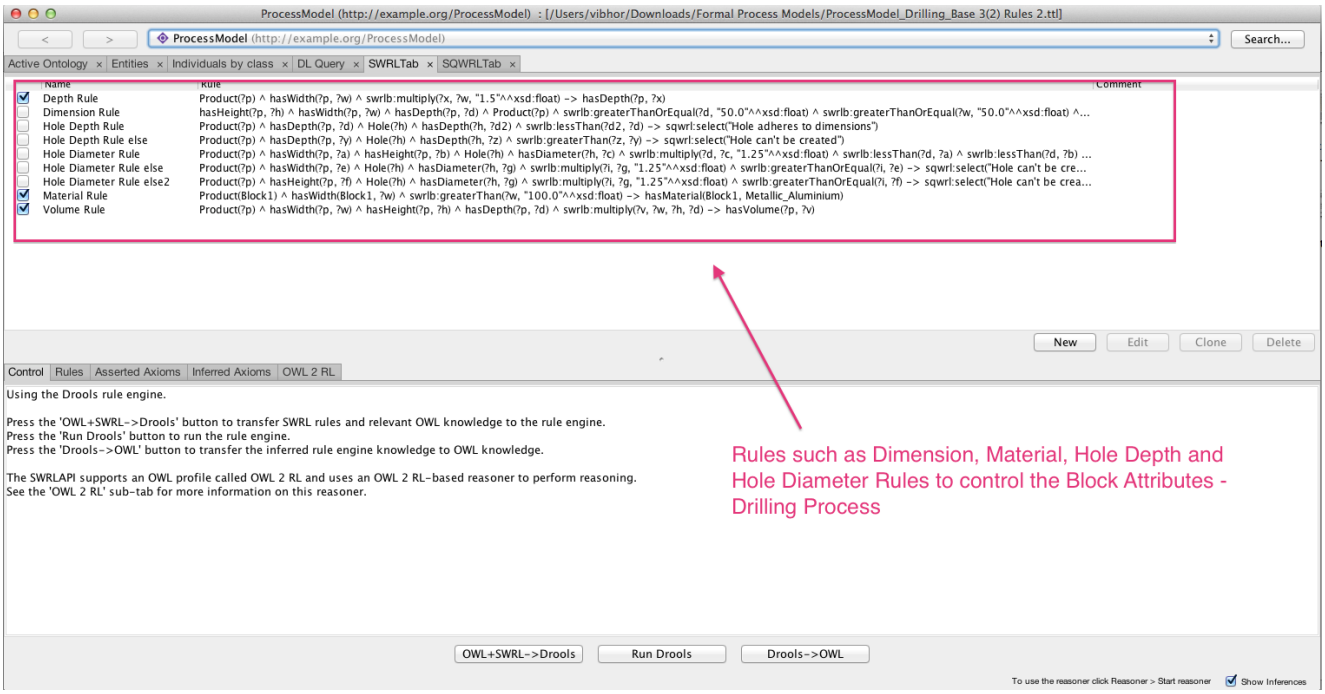


Figure 6-16: Engineering Rules – Drilling Process: Protégé

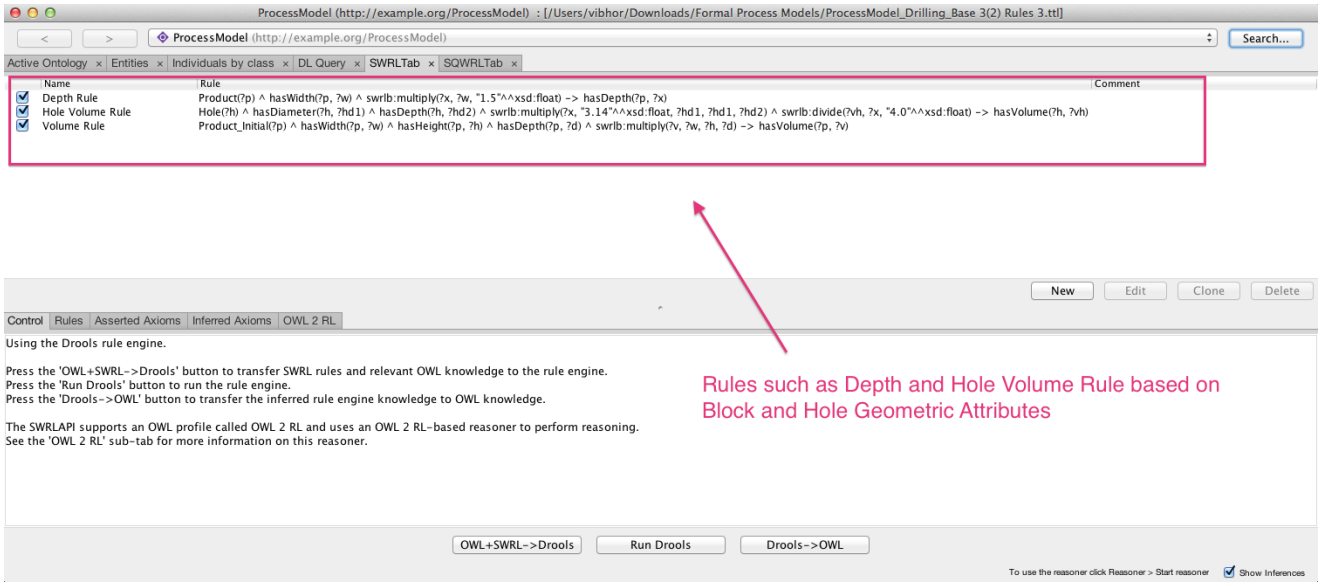


Figure 6-17: Engineering Rules 2 – Drilling Process: Protégé

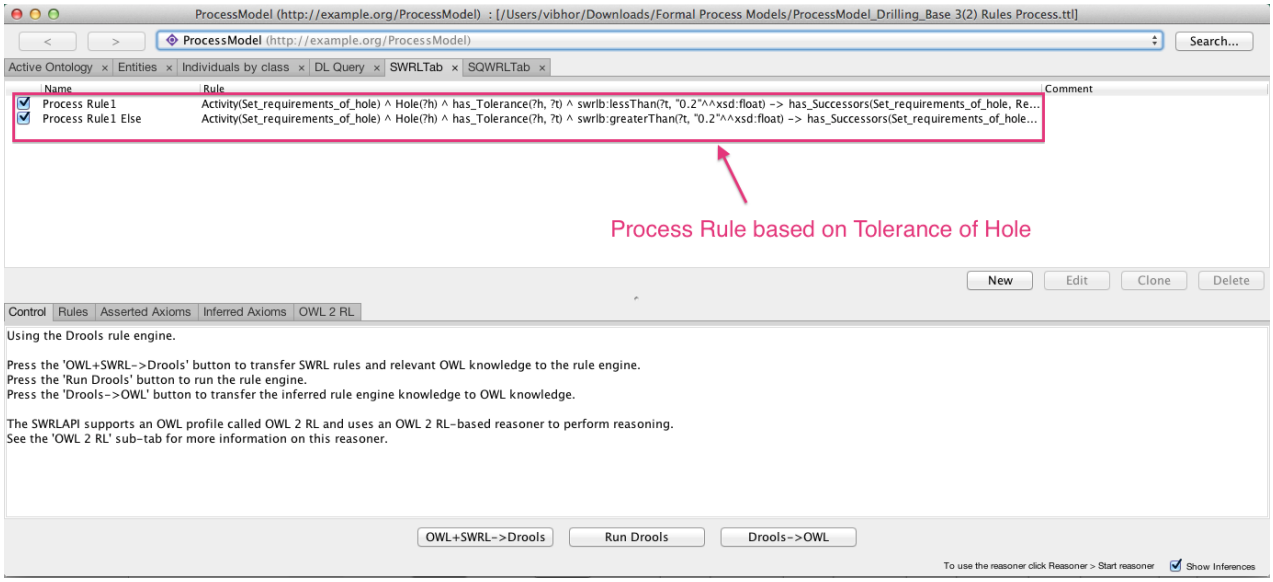


Figure 6-18: Engineering Rules 3 – Drilling Process: Protégé

6.4 Test Use Case 4: Designing a bookshelf (KBE and Neutral Formal Representation with MOKA methodology): Adapted from LinkedDesign

Pertaining to this research, the aim of this use case is to further refine and verify the system development as OWL/SWRL using the GPM-DEA schema as Meta Model and at the instance level with incorporation of product geometric accessible attributes as bookshelf dimensions and illustrate wider applicability. The DEA process initiates from the sub-function structures and function mapping of activities and objects to that of the engineering design process through to the generation of rules to control the bookshelf design process with its effect on bookshelf attributes. The following questions arise which will be verified in the next chapter –

1. *Can the instances of bookshelf design process be automatically generated based on function structures of individual activities along with objects and engineering rules for controlling the effect on geometric attributes of the bookshelf?*

2. *Can the implementation (ontology and rule representation) of the generated activities, objects and rules generate appropriate and accurate numeric values to bookshelf attributes thus successfully enabling DEA with a KBE approach?*

The instantiation of GPM-DEA with Use case 4 with its formalisation in OWL/SWRL as system development is discussed in this section.

6.4.1 Function Structure Matching

As observed from Figure 6-19, the bookshelf design process has 3 sub-functions – 'Detailed_design_3D_model_bookshelf', 'Variable_input_output_parameters' and 'Virtual_positioning'. It can be observed from Figure 6-20, activity 'Input bookshelf parameters' satisfies function – 'Detailed_design_3D_model_bookshelf'. Similarly, activities such as 'Compute parameters NDW, NSH' and 'Compute parameters SHL, WAL, SHS'satisfy function – 'Variable_input_output_parameters'. Similarly, the activity 'Positioning of the bookshelf' satisfies the function – 'Virtual_positioning'. Thus from the activity knowledgebase, bookshelf design processshould have all of the above mentioned four activities.

Engineering Design Process	Engineering Design Process Functional Requirement	Sub-Functions
Drilling Process1	<Drilling> is a <cutting process> that uses a <drill bit> to <cut a hole of circular crosssection> in <solid materials>. The <hole> can be of various dimensions depending upon the drill bit dimensions	Cut_hole_of_circular_cross_section + Precision_of_hole_dimensions
Precision Forging1	<Achieve an accuracy of +/-1mm in shape prediction>, as shape prediction accounts for a bulk of the manufacture objectives.	Achieve an accuracy of +/-1mm + shape prediction
Conceptual Design1 Fan Blades	The fan blades <spin to accelerate a mass of air> into the engine to <generate thrust> that propels the aircraft forward. Fan blades also function to <reduce total engine damage> from the <ingestion of various foreign objects> such as birds by radially deflecting outward such objects rather than passing them through to the core parts of the engine. The <dovetail attachment> of fan blades are used to <secure the blades to the hub or disk>. It is important to allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength	Generate thrust + reduce total engine damage from the ingestion of various foreign objects such as birds + secure the blades to the hub or disk + allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength
Bookshelf Design Process	Detailed design 3D model of bookshelf based on multiple scenarios of variable input and output parameters as virtual representation	Detailed_design_3D_model_bookshelf + Variable_input_output parameters + Virtual_positioning

Figure 6-19: Bookshelf Design Process Functional Requirements & Sub Functions: Knowledgebase

Engineering Design Process Activity (Physical/Informatical)	Equivalent Virtual Activities	Functional Requirement --[Sub-Functions]
Create block (workpiece)	Protrude block (workpiece)	Manufacture block
Drill hole	Create hole + Subtract hole	Cut hole of circular cross section
Ream hole	Create hole + Subtract hole	Cut hole of circular cross section + enlarge hole + high surface finish
Bore hole	Create hole + Subtract hole	Cut hole of circular cross section + enlarge hole
Selection and positioning of reamer		
Set requirements of hole		Precision_of_hole_dimensions
Selection and positioning of drill bit		
Punch hole	Create hole + Subtract hole	Cut hole of circular cross section + shear force
Extrusion		Extruded stem long enough for the part to be handled in subsequent operations + base of the extruded part needs to have enough material for subsequent heading + shape prediction
Stamping		Enough energy in the top die to achieve the movement required to reduce the gap between the dies to the sum of the gap between the flash lines and the minimum running thickness + shape prediction
Heading		The heading cavity is filled out as much as possible even if the cavity does not get filled in its entirety + shape prediction
Blade Geometry Optimisation		Fan blades spin to accelerate a mass of air into the engine + generate thrust + 80% of the thrust delivered by fan + reduce total engine damage from the ingestion of various foreign objects such as birds
Dovetail Attachment		Secure the blades to the hub or disk
Material Selection		Allocate material with high damage tolerance, ductility, high cycle fatigue (HCF) strength and yield strength
Input bookshelf parameters	Input bookshelf parameters	Detailed_design_3D_model_bookshelf
Compute parameters NDW, NSH	Compute parameters NDW, NSH	Variable input and output parameters
Compute parameters SHL, WAL, SHS	Compute parameters SHL, WAL, SHS	Variable input and output parameters
Positioning of the bookshelf	Positioning of the bookshelf	Virtual_positioning

Figure 6-20: Activities with Functions & Sub Functions: Knowledgebase

All the activities with their inputs, outputs, controls as rules and mechanisms as resources along with participating objects as inputs is shown in Figure 6-21. Their corresponding graphical representation is an ICOM box of IDEF0 standard in GPM-DEA as illustrated in next section 6.4.2. A snapshot of the rules controlling the activities is shown in Figure 6-22.

Engineering Design Process Activity (Physical/Informatical/ Virtual)	Objects - Product as primary object	Activity Inputs	Activity Outputs	Controls: Engineering Rules	Resources
Input bookshelf parameters	Product - Bookshelf	Width (W), Height (H), Depth (T), Horizontal length of 1 shelf (HS), Vertical length of 1 shelf (VS), Thickness of bottom shelf (TB), Thickness of dividing walls (TD), Thickness of side walls (TS), Thickness of top shelf (TT), Thickness of inner shelf (TSH)	Value assigned to W, H, T, HS, VS, TB, TD, TS, TT, TSH		Design tool
Compute parameters NDW, NSH	Product - Bookshelf	Value assigned to W, H, HS, VS	Value assigned to No. of dividing walls (NDW), No. of shelves (NSH)	Dividing Walls Rule , Shelves Rule	Design tool
Compute parameters SHL, WAL, SHS	Product - Bookshelf	Value assigned to W, H, TD, TS, TB, TT, TSH, Value assigned to NDW, NSH	Value assigned to Shelf Length (SHL), Length of side & dividing walls (WAL), Vertical space between shelves (SHS)	Shelf Length Rule, Side & Dividing Walls Rule, Vertical Space Rule	Design tool
Positioning of the bookshelf	Product - Bookshelf	Value assigned to W, TS, TB, TSH, Value assigned to SHL, WAL, SHS	Value assigned to X1-X7, Y1-Y7, Z1-Z7 as Bookshelf Position Coordinates	Dividing Wall Position Rule, Multiple Dividing Walls Position Rule, Shelf Position Rule, Multiple Shelf Position Rule, Side Wall Position Rule, Side Wall Right Position Rule, Topshelf Position Rule	Design tool

Figure 6-21: Activities with Inputs, Outputs, Rules and Resources with Objects for Bookshelf Design Process: Knowledgebase

Controls: Engineering Rule	Engineering Rule Description	Type of Rule
Dividing Walls Rule	NDW is based on HS and W, If (W<0.5*HS, "ERROR") elseif (W<=HS, NDW=0) else (NDW=Int(W/HS)-1)	Geometry Rule, Math Rule, Production Rule
Shelves Rule	(NSH is based on H and VS, If (VS>H, "ERROR") elseif (2*VS>H, NSH=0) else (NSH=Int((H/VS)-1))	Geometry Rule, Math Rule, Production Rule
Shelf Length Rule	(SHL=(W-(2*TS + TD*NDW))/(NDW+1))	Geometry Rule, Math Rule
Side & Dividing Walls Rule	(WAL=H-(TB +TT))	Geometry Rule, Math Rule
Vertical Space Rule	(SHS=(WAL-NSH*TSH)/NSH)	Geometry Rule, Math Rule
Dividing Wall Position Rule	(X1=TS+SHL,Y1=TB, Z1=0)	Geometry Rule, Math Rule
Multiple Dividing Walls Position Rule	(X2=TS+SHL,Y2=TB,Z2=0)	Geometry Rule, Math Rule
Shelf Position Rule	(X3=TS,Y3=TB-TSH,Z3=0)	Geometry Rule, Math Rule
Multiple Shelf Position Rule	(X4=TS,Y4=TB+SHS,Z4=0)	Geometry Rule, Math Rule
Side Wall Position Rule	(X5=0,Y5=TB,Z5=0)	Geometry Rule, Math Rule
Side Wall Right Position Rule	(X6=W-TS,Y6=TB,Z6=0)	Geometry Rule, Math Rule
Topshelf Position Rule	(X7=0,Y7=TB+WAL,Z7=0)	Geometry Rule, Math Rule

Figure 6-22: Engineering Rules controlling the Design Process Activities for Bookshelf Design Process: Knowledgebase

6.4.2 Informal / Semiformal Representation: GPM-DEA

An instance of bookshelf design process has been devised and instantiated in GPM-DEA as informal / semiformal representation as shown in Figure 6-23. All activities are virtual activities in context to the bookshelf design process as the process is realised at the detailed design stage in the form of geometric modelling. For example 'input bookshelf parameters' will allow user to enter input values to bookshelf geometric attributes such as Width (W), Height (H), Depth (T) and other attributes. All activities are represented using ***IDEF0*** notation for functional modelling, and satisfy a function. As explained in the working of GPM-DEA in section 5.3.1 with the help of Figure 5-3, if the functions of activities are not available in the knowledgebase as inputs then the user will need to enter the functions of activities and objects to successfully enable generative modelling capability of the model. All activities are represented with inputs, controls as rules, outputs and mechanisms as resources (ICOM).

The process-sequencing options are represented with red links as ***UML condition link*** for multiple 'what-if' scenarios. Sub-activities can be represented using blue link. The product model representing the initial and the final state as bookshelf design parameter values and the

designed bookshelf in virtual 3d model representation is represented using UML class diagram with attributes and engineering rules as methods affecting the attributes. As explained in Chapter 4 and 5, ***UML class diagram*** is used for representing all participating objects in the engineering design process. Similarly, ***SysML requirement diagram*** is used for representing the functional requirements of the bookshelf design process as sub functions in context to the bookshelf as product.

Figure 6-24 shows a snapshot of function matching of individual activities as function structures of bookshelf design process functional requirements with links to rules in ICOM box.

Figure 6-25 shows a snapshot of product as bookshelf in initial and final state with UML class diagram along with function matching of these objects. As it can be observed, various engineering rules such as dividing walls, shelves, side and dividing walls, sidewall position and topshelf position rules are informally represented as methods inside UML class diagram along with attributes in this research. It also shows the interface of the process model to the product model with part and assembly features of the bookshelf such as shelf, frame and dividing walls using ***UML composition and aggregation structural links***.

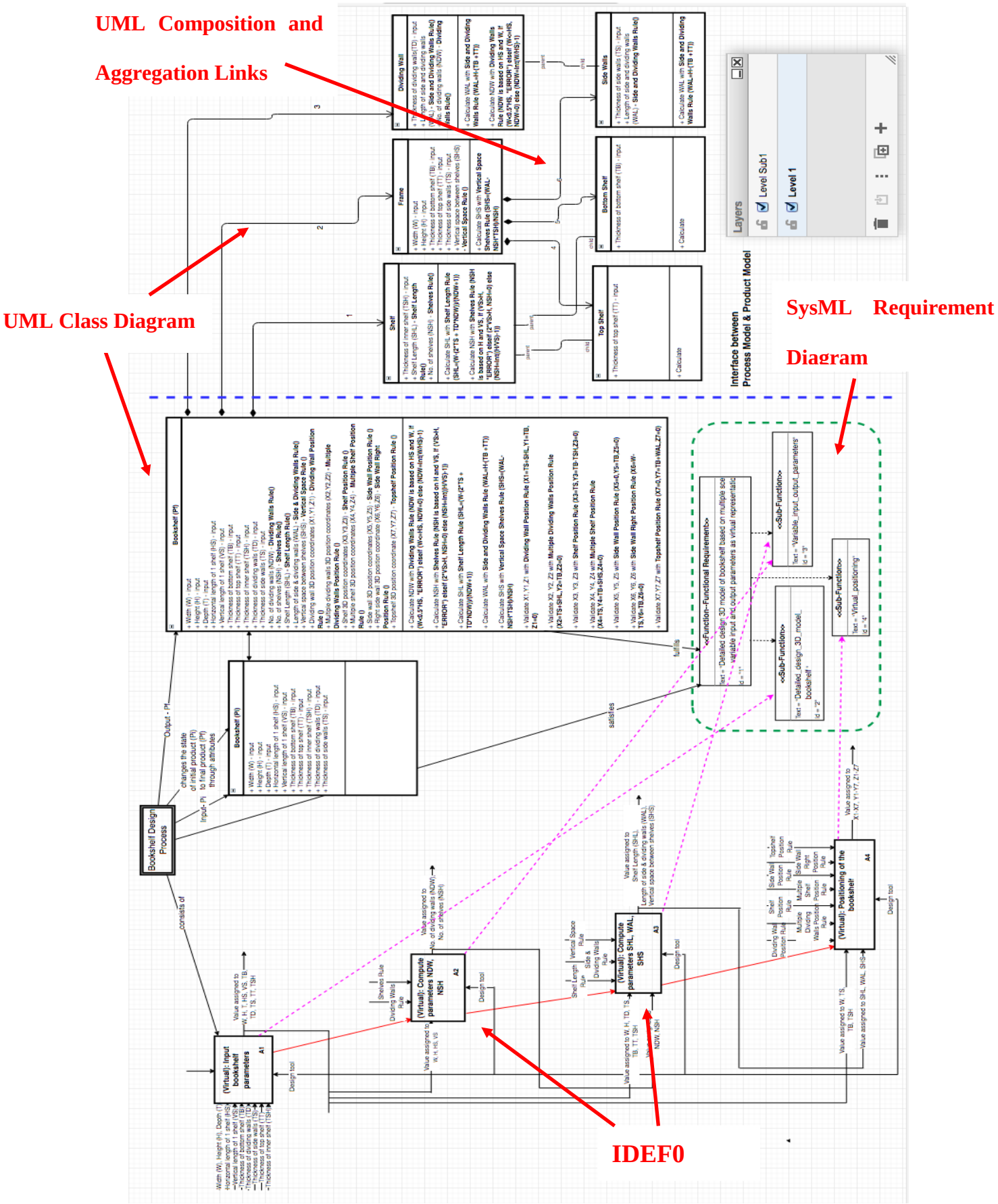


Figure 6-23: An Instance of Bookshelf Design Process in GPM-DEA: Informal / Semiformal Representation

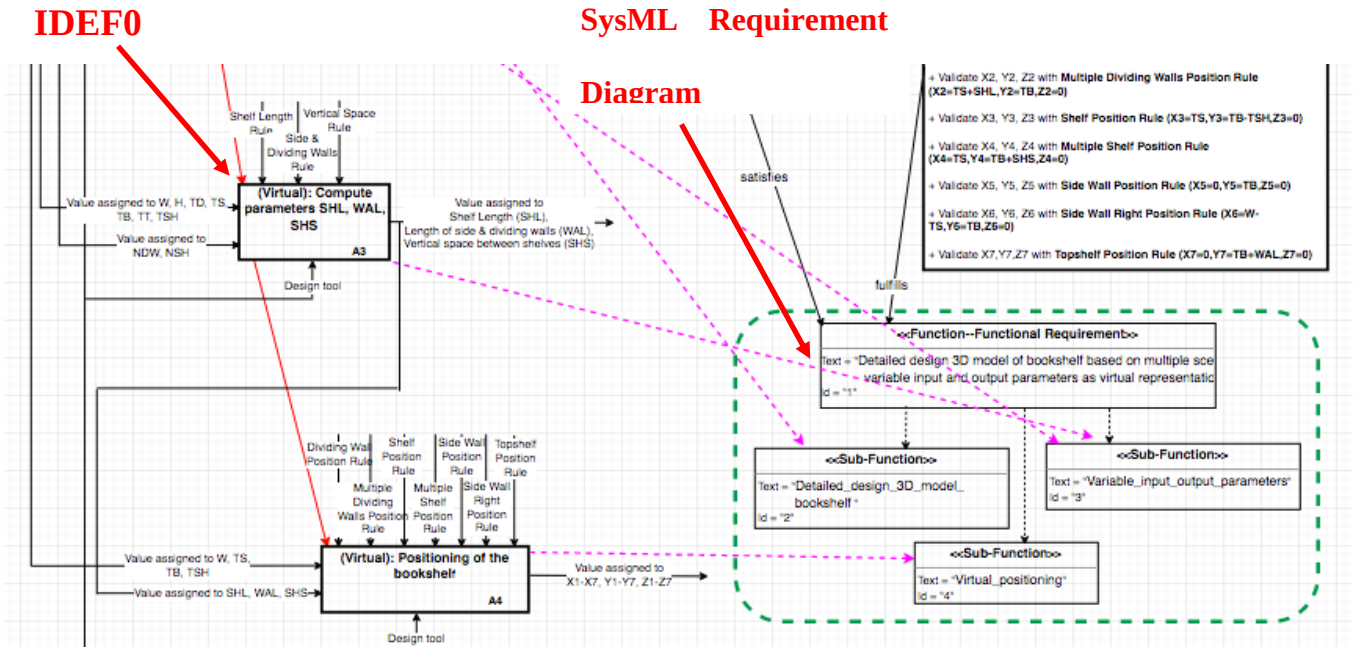


Figure 6-24: Function Structure Matching – Bookshelf Design Process Activities with Links to Rules

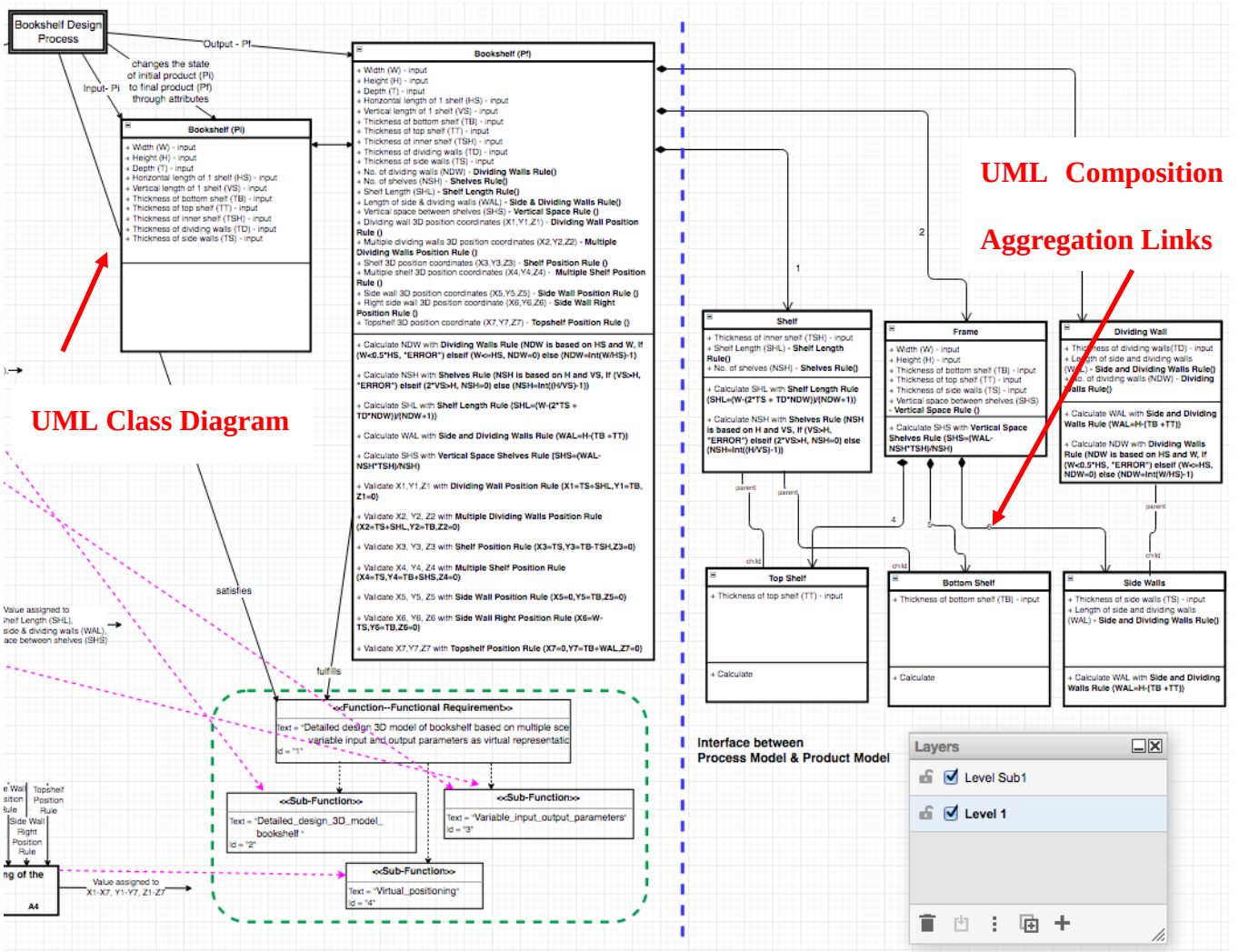


Figure 6-25: Product in Initial and Final State and Function Matching – Bookshelf Design Process Objects with Description of Rules

6.4.3 Formal Representation: OWL/SWRL

The instantiated GPM-DEA model for bookshelf design process has been then represented in OWL/SWRL as neutral formal representation in this research. The activities of the bookshelf design process in GPM-DEA corresponding to the graphical representation in Figure 6-23 and 6-24 as IDEF0 ICOM box are represented formally in OWL2 with associated id, inputs, outputs, resources and linkage to engineering rules using classes and properties. All the activities interlinked with product structure with attributes, function as sub-functions and behaviour corresponding to UML class diagram and SysML requirement diagram are also represented using OWL2 using classes and properties as illustrated in Figure 6-26.

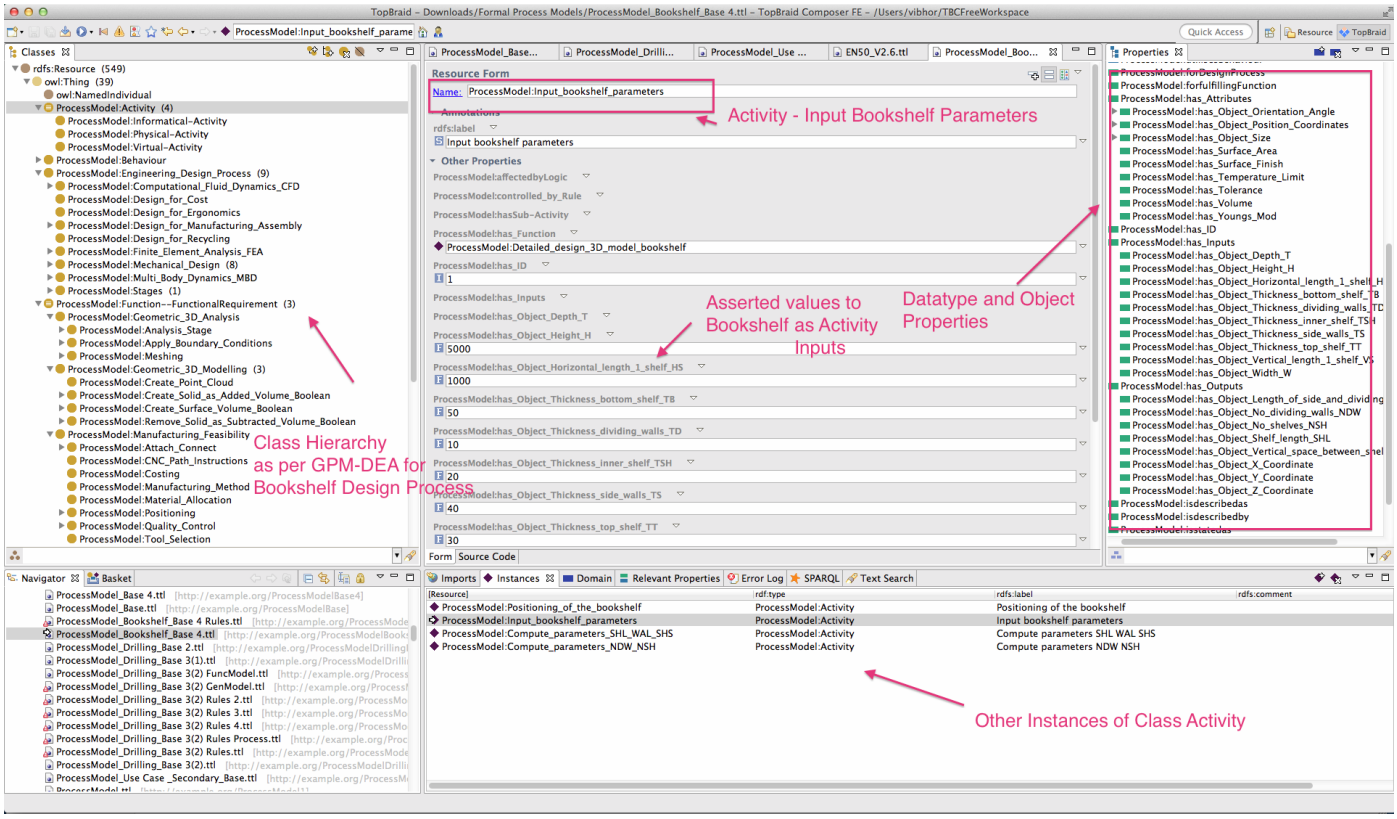


Figure 6-26: Bookshelf Design Process in OWL: TopBraid Composer FE

As observed from Figure 6-26, only OWL2 representation is utilised in Topbraid. All the classes with class-subclass relationship can be observed on the left. All the properties for inputs, outputs and other relationships marked as arrows in Fig 6-23 and 6-24 is represented

on the right under the properties tab in Figure 6-26. Instances have been produced on the bottom and all the relationships between instances can be observed in the resource form in the centre tab. As also observed from Figure 6-26, the 'input bookshelf parameters' instance of activity tab illustrates its ID as 1, allows user to enter input values to this activity in terms of bookshelf geometric attributes as object attributes such as Object_Height_H being given value 5000 mm, Object_Horizontal_length_1_shelf_HS being given value 1000 mm. The equivalent activities and functions are also asserted using specified property in the form of 'has_Function'.

For the product attributes in UML class diagram as activity inputs and outputs as IDEF0 ICOM, in context to GPM-DEA, datatype properties have been created and instantiated. As explained in section 5.5.1, (ProcessModel:has_Inputs) and (ProcessModel:has_Outputs) are the datatype properties created in GPM-DEA to assert activity inputs and outputs in terms of object attributes. All sub-properties of (ProcessModel:has_Attributes) such as (ProcessModel:has_Object_Size), (ProcessModel:has_Object_Position_Coordinates) and (ProcessModel:has_Object_Orientation_Angle), (ProcessModel:has_Volume) can be asserted as sub properties of (ProcessModel:has_Inputs) and (ProcessModel:has_Outputs). As observed from Figure 6-22, following properties as sub properties of (ProcessModel:has_Attributes) have been classified as sub properties of (ProcessModel:has_Inputs) as activity inputs –

I. ProcessModel:has_Object_Depth_T
II. ProcessModel:has_Object_Height_H
III. ProcessModel:has_Object_Horizontal_length_1_shelf_HS
IV. ProcessModel:has_Object_Thickness_bottom_shelf_TB
V. ProcessModel:has_Object_Thickness_dividing_walls_TD
VI. ProcessModel:has_Object_Thickness_inner_shelf_TSH

VII. ProcessModel:has_Object_Thickness_side_walls_TS

VIII. ProcessModel:has_Object_Thickness_top_shelf_TT

IX. ProcessModel:has_Object_Vertical_length_1_shelf_VS

X. ProcessModel:has_Object_Width_W

Similarly, the following properties as sub properties of (ProcessModel:has_Attributes) have been classified as sub properties of (ProcessModel:has_Outputs) as activity outputs –

I. ProcessModel:has_Object_Length_of_side_and_dividing_walls_WAL

II. ProcessModel:has_Object_No_dividing_walls_NDW

III. ProcessModel:has_Object_No_shelves_NSH

IV. ProcessModel:has_Object_Shelf_length_SHL

V. ProcessModel:has_Object_Vertical_space_between_shelves_SHS

VI. ProcessModel:has_Object_X_Coordinate

VII. ProcessModel:has_Object_Y_Coordinate

VIII. ProcessModel:has_Object_Z_Coordinate

UML class diagram attributes can thus be neutrally represented using OWL2 datatype properties. All the properties can be observed in Figure 6-26 on the right tab.

As per the class-subclass relationship of function structures discussed in section 6.4.1, the sub function - 'Detailed_design_3D_model_bookshelf' is an instance of class 'Geometric_3D_Modelling'. The sub functions - 'Variable_input_output_parameters' and 'Virtual_positioning' are also instances of class 'Geometric_3D_Modelling'. Various instances of functions are shown with the help of Figure 6-27.

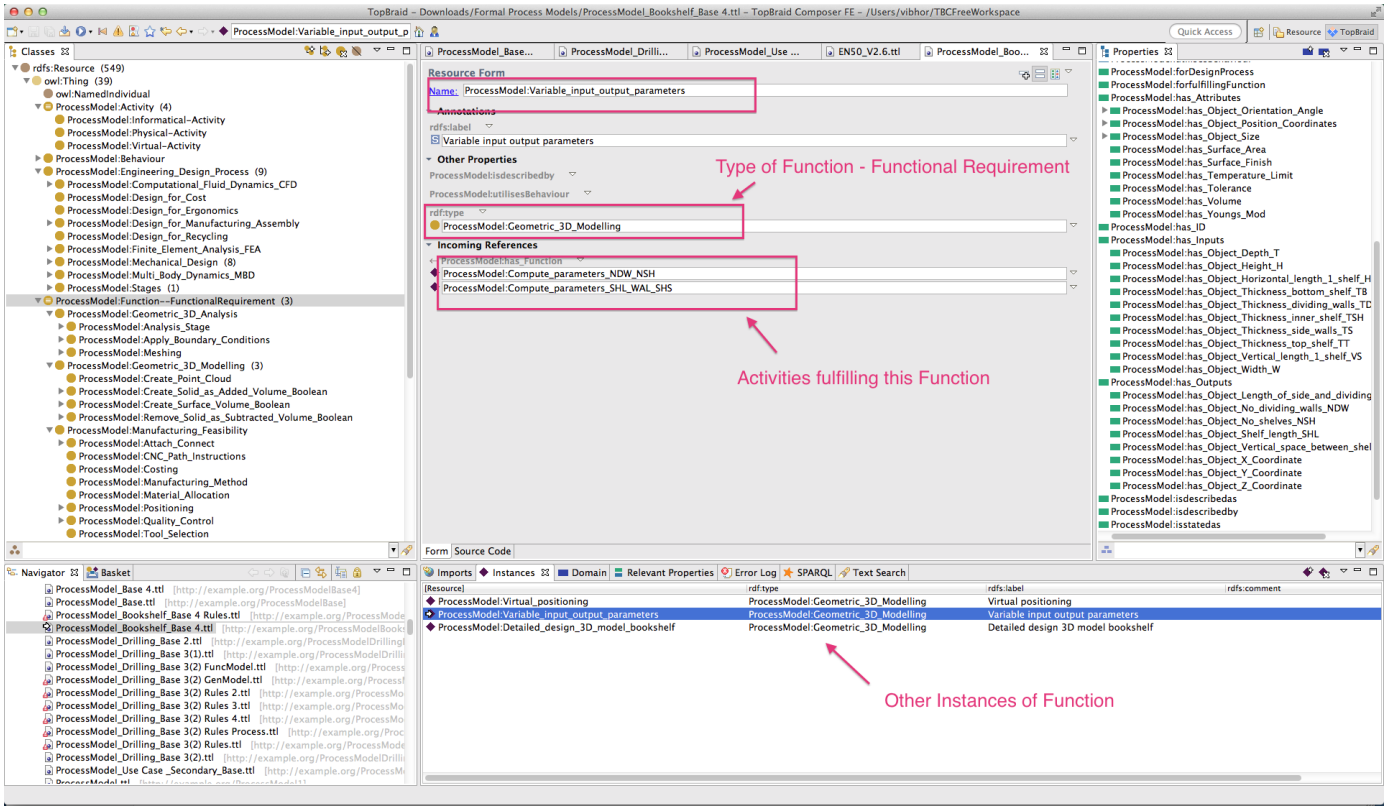


Figure 6-27: Instances of Functions – Bookshelf Design Process: Topbraid

The implementation of the SWRL generative modelling functions for bookshelf design process is illustrated with the help of Figure 6-28.

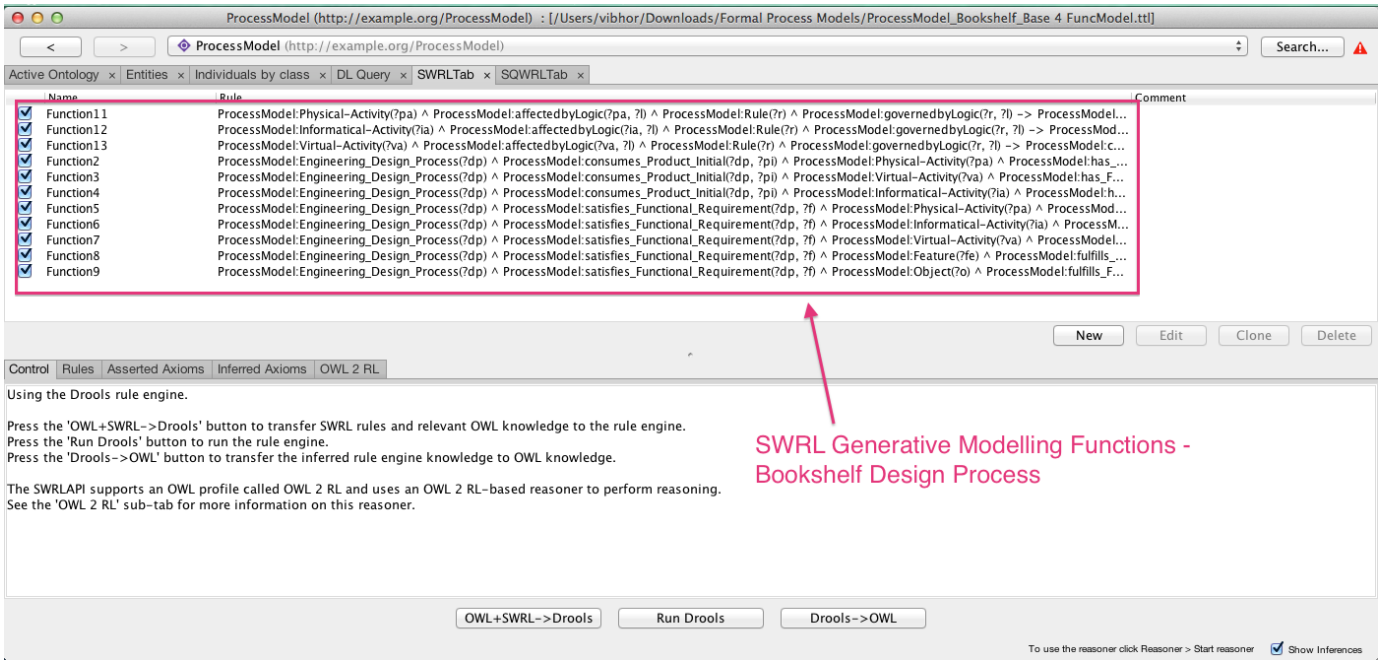


Figure 6-28: SWRL Functions - Generative Modelling in Bookshelf Design: Protégé

Figure 6-28 illustrates the representation or codification of functions developed in this work, which allow GPM-DEA to generate activities and objects based on the sub functions of each activity and the object along with rules based on logic. The verification of the generative modelling capability of GPM-DEA through bookshelf design use-case will be discussed in next chapter by testing the reasoning capability of the drools reasoner on SWRL axioms and SQWRL query language.

For the instantiated bookshelf design use case example in GPM-DEA, the final product is the virtual representation of bookshelf as 3D model.The fit class becomes the most crucial class in representing the part and assembly relations of the bookshelf. As illustrated with the help of Figure 6-23 and 6-25, there are 6 parts of the bookshelf – dividing walls, frame, shelves, bottom shelf, side walls and top shelf along with bookshelf and frame as assembly. All the parent child relationships are shown graphically in the informal/semiformal model. The assembly parts relations of the bookshelf are shown in OWL2 with the help of Figure 6-29.

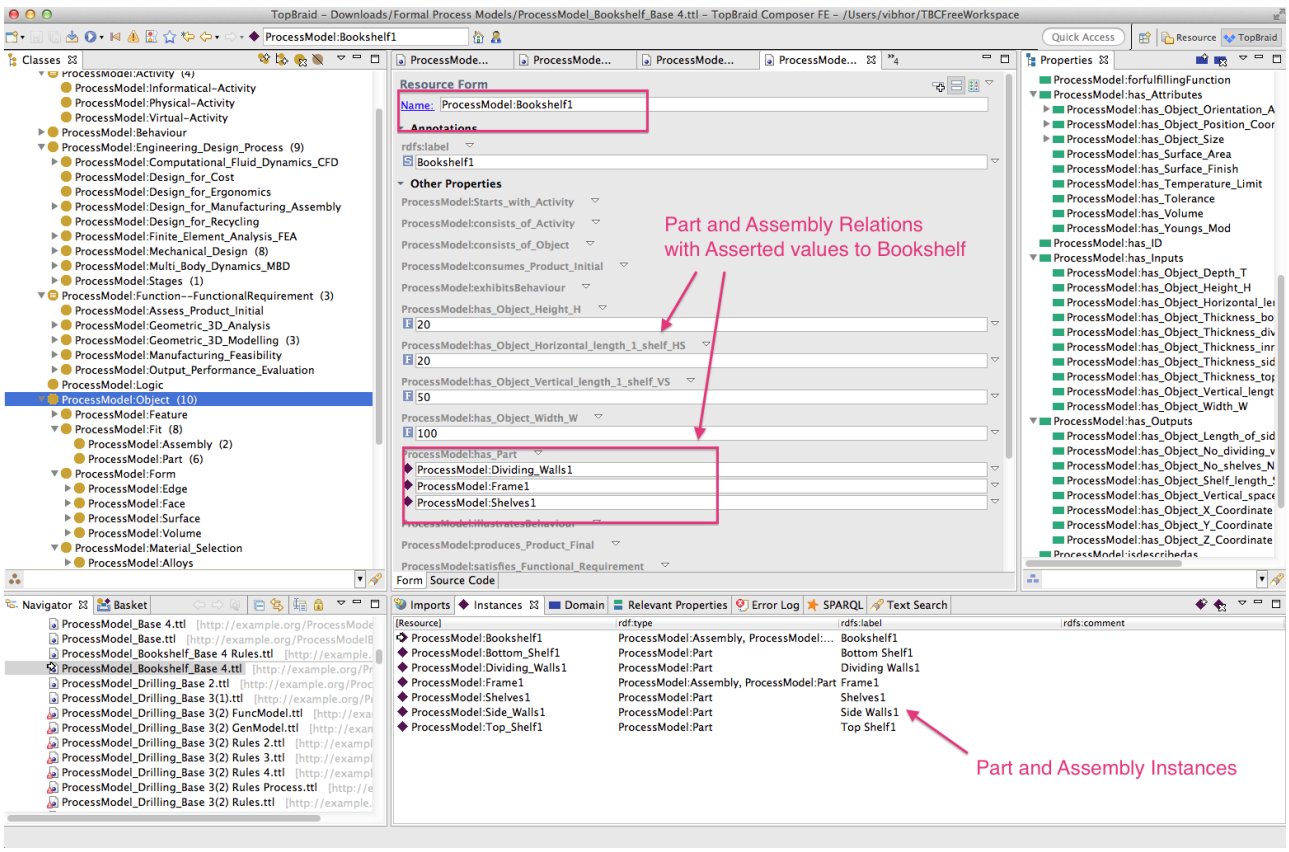


Figure 6-29: Fit: Assembly and Part Relations for Bookshelf: Topbraid Composer FE

The assembly relationships can be queried in the SPARQL query tab to generate the results required from the user. These queries results are illustrated with the help of Figure 6-30.

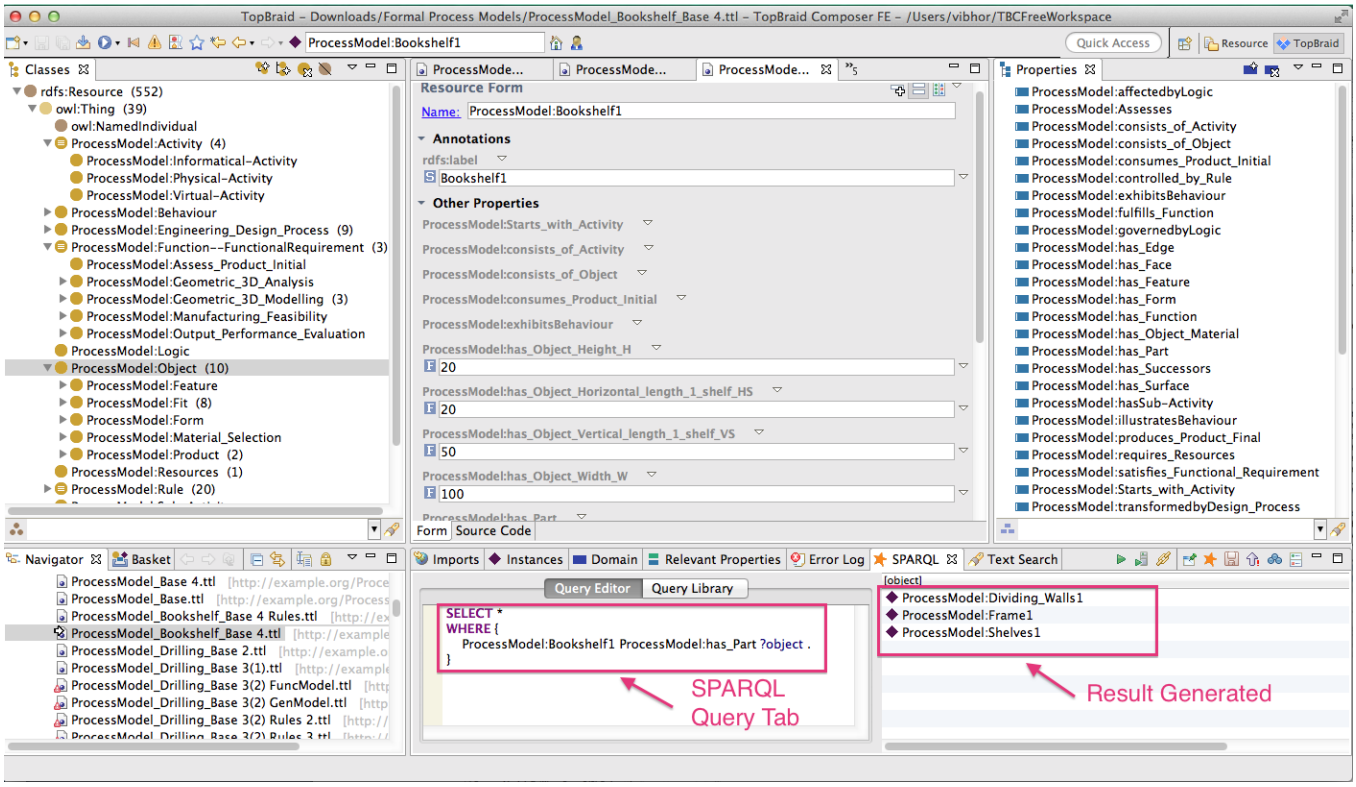


Figure 6-30: SPARQL Query Result: Bookshelf Part and Assembly Relations

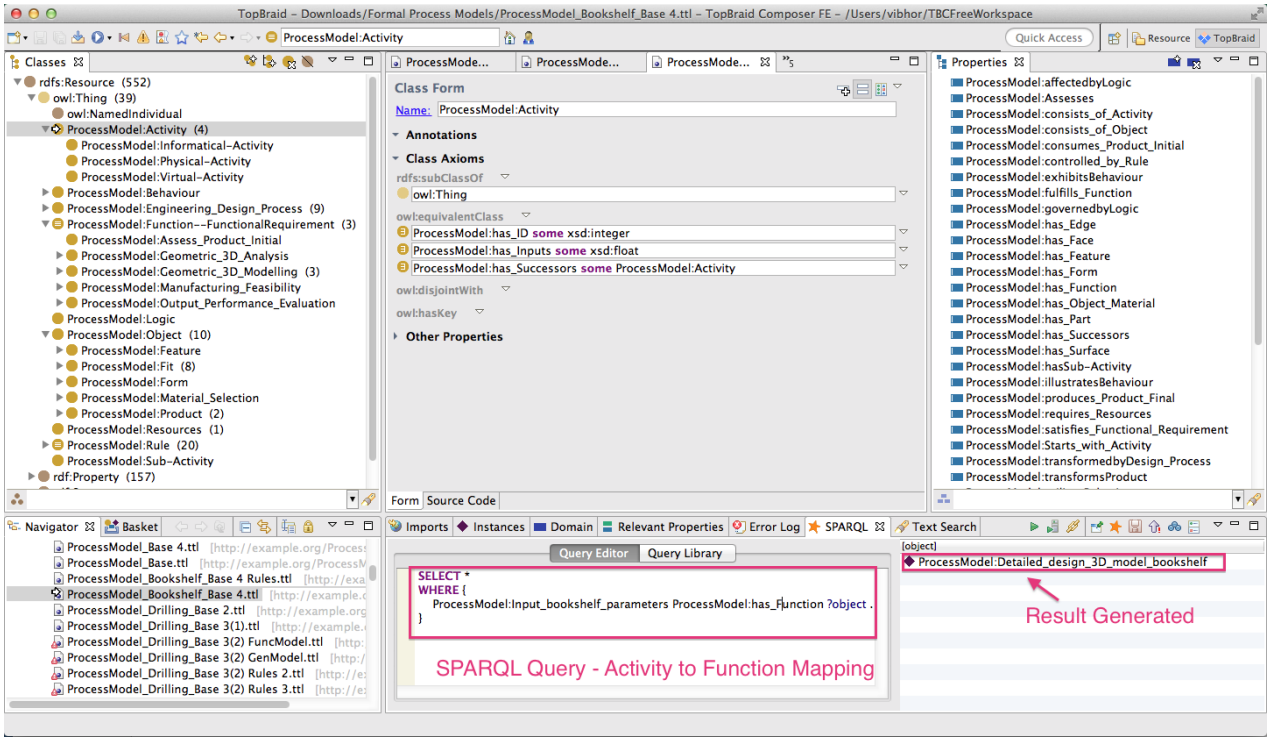


Figure 6-31: SPARQL Query Result – Activity Function Mapping – Bookshelf Design

Similarly, to illustrate the SPARQL query for activity-function mapping for the activity - 'Input bookshelf parameters' is illustrated with Figure 6-31.

SPARQL query for illustrating the rule to logic mapping for bookshelf design process with a few examples is shown in Figure 6-32.

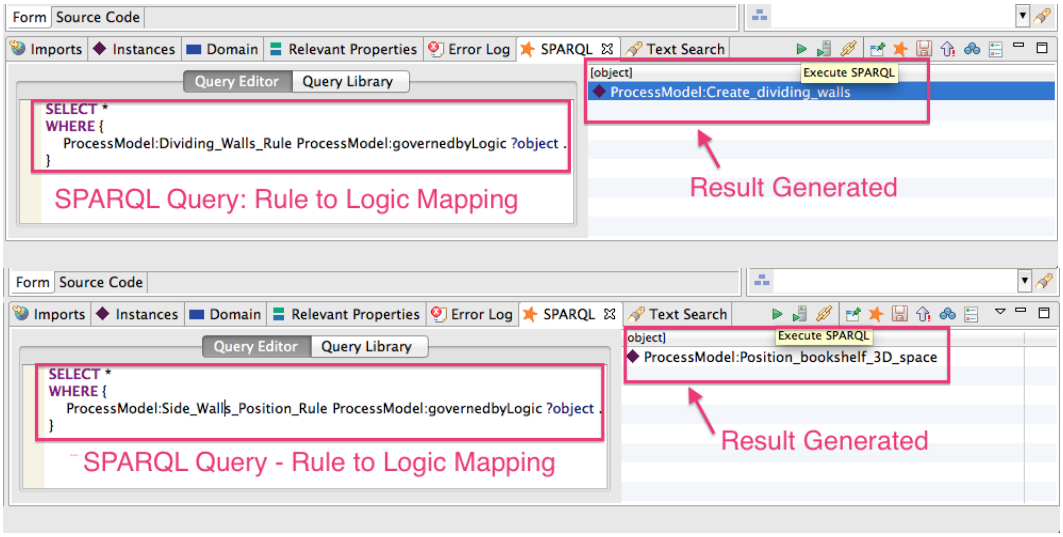


Figure 6-32: SPARQL Query Result: Rule to Logic Mapping – Bookshelf Design Process

The engineering rules represented informally in GPM-DEA as methods in UML are also codified using SWRL as neutral formal representation in this work. From the knowledgebase, shown in Figure 6-22, consisting of engineering rules for the bookshelf design use case, all the rules in SWRL axioms are illustrated as follows –

1. ***SWRL Dividing Walls Rule***– NDW is based on HS and W, If (W<0.5*HS, "ERROR") elseif (W<=HS, NDW=0) else (NDW=Int(W/HS)-1)

   SWRL Representation - Product(?p) ^ has_Object_Width_W(?p, ?w) ^ has_Object_Horizontal_length_1_shelf_HS(?p, ?hs) ^ swrlb:multiply(?x, "0.5"^^xsd:float, ?hs) ^ swrlb:lessThan(?w, ?x) -> sqwrl:select("Error - Too narrow for a bookshelf")

   And

   Product(?p) ^ has_Object_Width_W(?p, ?w) ^ has_Object_Horizontal_length_1_shelf_HS(?p, ?hs) ^ swrlb:multiply(?x, "0.5"^^xsd:float, ?hs) ^ swrlb:greaterThan(?w, ?x) ^ swrlb:lessThanOrEqual(?w, ?hs) -> has_Object_No_dividing_walls_NDW(?p, "0.0"^^xsd:float)

   And

   Product(?p) ^ has_Object_Width_W(?p, ?w) ^ has_Object_Horizontal_length_1_shelf_HS(?p, ?hs) ^ swrlb:greaterThan(?w, ?hs) ^

swrlb:divide(?y, ?w, ?hs) ^ swrlb:subtract(?z, ?y, "1.0"^^xsd:float) -> has_Object_No_dividing_walls_NDW(?p, ?z)

2. ***SWRL Shelves Rule*** - (NSH is based on H and VS, If (VS>H, "ERROR") elseif (2*VS>H, NSH=0) else (NSH=Int((H/VS)-1))

   SWRL Representation - Product(?p) ^ has_Object_Height_H(?p, ?h) ^ has_Object_Vertical_length_1_shelf_VS(?p, ?vs) ^ swrlb:greaterThan(?vs, ?h) -> sqwrl:select("Error - Too low for even one space in the bookshelf")
   And
   Product(?p) ^ has_Object_Height_H(?p, ?h) ^ has_Object_Vertical_length_1_shelf_VS(?p, ?vs) ^ swrlb:lessThan(?vs, ?h) ^ swrlb:multiply(?a, "2.0"^^xsd:float, ?vs) ^ swrlb:greaterThan(?a, ?h) -> has_Object_No_shelves_NSH(?p, "0.0"^^xsd:float)
   And
   Product(?p) ^ has_Object_Height_H(?p, ?h) ^ has_Object_Vertical_length_1_shelf_VS(?p, ?vs) ^ swrlb:multiply(?a, "2.0"^^xsd:float, ?vs) ^ swrlb:lessThan(?a, ?h) ^ swrlb:divide(?b, ?h, ?vs) ^ swrlb:subtract(?c, ?b, "1.0"^^xsd:float) -> has_Object_No_shelves_NSH(?p, ?c)

3. ***SWRL Shelf Length Rule*** - (SHL=(W-(2*TS + TD*NDW))/(NDW+1))

   SWRL Representation - Product(?p) ^ has_Object_Width_W(?p, ?w) ^ has_Object_Thickness_side_walls_TS(?p, ?ts) ^ has_Object_Thickness_dividing_walls_TD(?p, ?td) ^ has_Object_No_dividing_walls_NDW(?p, ?ndw) ^ swrlb:multiply(?a1, "2.0"^^xsd:float, ?ts) ^ swrlb:multiply(?b1, ?td, ?ndw) ^ swrlb:add(?c1, ?ndw, "1.0"^^xsd:float) ^ swrlb:add(?d1, ?a1, ?b1) ^ swrlb:subtract(?e1, ?w, ?d1) ^ swrlb:divide(?f1, ?e1, ?c1) -> has_Object_Shelf_length_SHL(?p, ?f1)

4. ***SWRL Side and Dividing Walls Rule*** - (WAL=H-(TB +TT))

   SWRL Representation - Product(?p) ^ has_Object_Height_H(?p, ?h) ^ has_Object_Thickness_bottom_shelf_TB(?p, ?tb) ^ has_Object_Thickness_top_shelf_TT(?p, ?tt) ^ swrlb:add(?d, ?tb, ?tt) ^ swrlb:subtract(?e, ?h, ?d) -> has_Object_Length_of_side_and_dividing_walls_WAL(?p, ?e)

5. ***SWRL Vertical Space Rule*** - (SHS=(WAL-NSH*TSH)/NSH)

   SWRL Representation - Product(?p) ^ has_Object_Length_of_side_and_dividing_walls_WAL(?p, ?wal) ^ has_Object_No_shelves_NSH(?p, ?nsh) ^ has_Object_Thickness_inner_shelf_TSH(?p, ?tsh) ^ swrlb:multiply(?f, ?nsh, ?tsh) ^

swrlb:subtract(?g, ?wal, ?f) ^ swrlb:divide(?h, ?g, ?nsh) -> has_Object_Vertical_space_between_shelves_SHS(?p, ?h)

6. ***SWRL Dividing Wall Position Rule*** - (X1=TS+SHL,Y1=TB, Z1=0)

   SWRL Representation - Part(Dividing_Walls1) ^ Product(?p) ^ has_Object_Thickness_side_walls_TS(?p, ?ts) ^ has_Object_Shelf_length_SHL(?p, ?shl) ^ has_Object_Thickness_bottom_shelf_TB(?p, ?tb) ^ swrlb:add(?i, ?ts, ?shl) -> has_Object_X_Coordinate(Dividing_Walls1, ?i) ^ has_Object_Y_Coordinate(Dividing_Walls1, ?tb) ^ has_Object_Z_Coordinate(Dividing_Walls1, "0.0"^^xsd:float)

7. ***SWRL Shelf Position Rule*** - (X3=TS,Y3=TB-TSH,Z3=0)

   SWRL Representation - Part(Shelves1) ^ Product(?p) ^ has_Object_Thickness_side_walls_TS(?p, ?ts) ^ has_Object_Thickness_bottom_shelf_TB(?p, ?tb) ^ has_Object_Thickness_inner_shelf_TSH(?p, ?tsh) ^swrlb:subtract(?j, ?tb, ?tsh) -> has_Object_X_Coordinate(Shelves1, ?ts) ^ has_Object_Y_Coordinate(Shelves1, ?j) ^ has_Object_Z_Coordinate(Shelves1, "0.0"^^xsd:float)

8. ***SWRL Side Walls Position Rule*** - (X5=0,Y5=TB,Z5=0)

   SWRL Representation - Part(Side_Walls1) ^ Product(?p) ^ has_Object_Thickness_bottom_shelf_TB(?p, ?tb) -> has_Object_X_Coordinate(Side_Walls1, "0.0"^^xsd:float) ^ has_Object_Y_Coordinate(Side_Walls1, ?tb) ^ has_Object_Z_Coordinate(Side_Walls1, "0.0"^^xsd:float)

9. ***SWRL Top Shelf Position Rule*** - (X7=0,Y7=TB+WAL,Z7=0)

   SWRL Representation - Part(Top_Shelf1) ^ Product(?p) ^ has_Object_Thickness_bottom_shelf_TB(?p, ?tb) ^ has_Object_Length_of_side_and_dividing_walls_WAL(?p, ?wal) ^ swrlb:add(?k, ?tb, ?wal) -> has_Object_X_Coordinate(Top_Shelf1, "0.0"^^xsd:float) ^ has_Object_Y_Coordinate(Top_Shelf1,?k) ^ has_Object_Z_Coordinate(Top_Shelf1, "0.0"^^xsd:float)

Figure 6-33 illustrates all the SWRL rules for the bookshelf use case implemented in protégé SWRL tab.

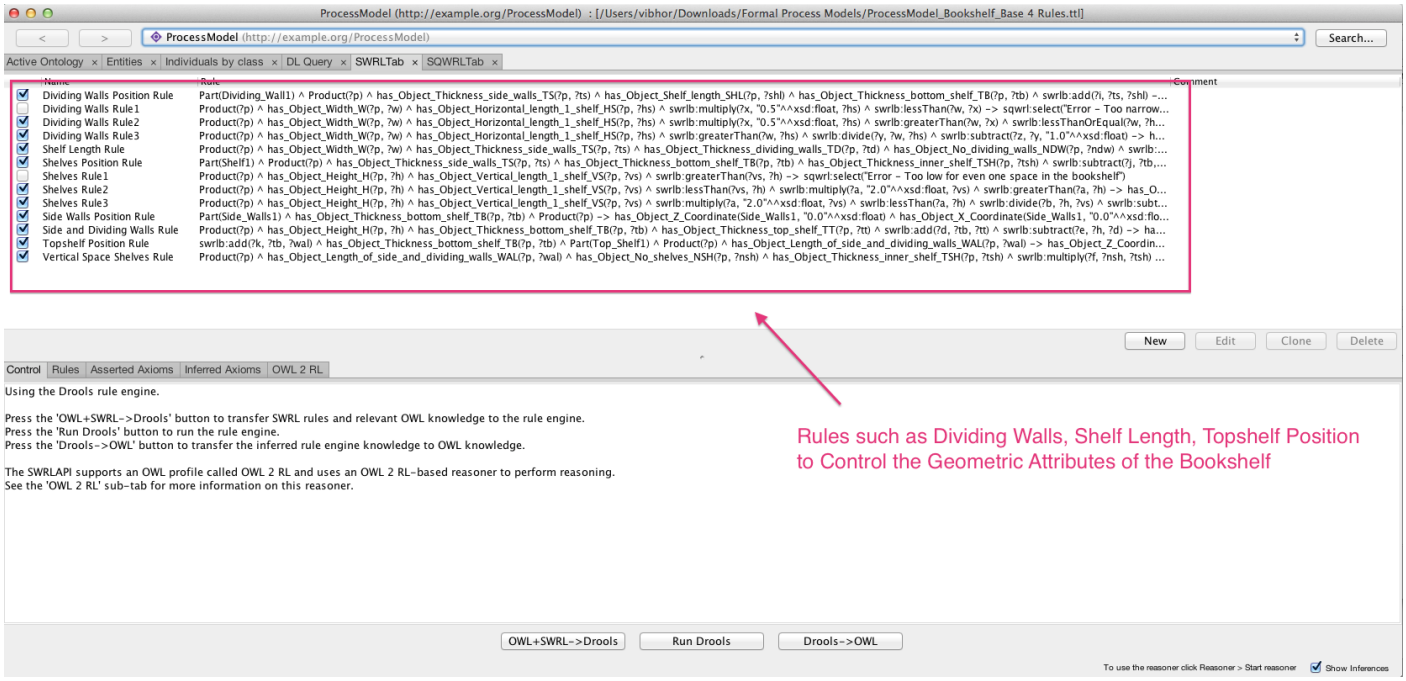


Figure 6-33: Engineering Rules – Bookshelf Design Process: Protégé

6.5 Summary

This chapter has discussed and elaborated on the 2 test use-cases for system development and verification of GPM-DEA in OWL/SWRL ontology and rule representation for DEA with a KBE approach with the effect of the process model on product's geometric attributes. The product's attributes can be accessed at the detailed design stage across proprietary platform specific DEA applications such as AML, ParaPy, CATIA Knowledgeware and Siemens NX KF. Both these use-cases follow the method of GPM-DEA schema mapping at the Meta model level and the instance level, developed as part of this research based on pilot use-cases and literature analysis, where the initial product is assessed at the beginning and the product with final state is produced at the completion of the process. The next chapter is going to perform experiments on these use-cases with appropriate reasoning and query mechanism and semantic clarity to test and verify the accuracy of the results produced from ontology and rule representation.

7 Experimental Verification of Knowledge Representation System

7.1 Introduction

This chapter elaborates on working and experimentation of the developed system with test use-cases in order to explore various aspects of GPM-DEA implementation in ontology and rule representation. It will provide experimental verification of the research hypothesis in order to satisfy and provide proof of the novelty of this research work. Test use-cases in the form of drilling a hole in a block and bookshelf design process collated from literature have been instantiated in GPM-DEA and then formalised in OWL/SWRL as platform independent and neutral representation as described in chapter 6 for system development. Aspects of both these use-cases such as rules with links to activities and objects generated from functional requirements, with their effect on product's geometric attributes have been implemented in proprietary platform specific DEA system applications such as AML, ParaPy with KBE functionalities and CATIA Knowledgeware, Siemens NX KF with parametric modelling providing GA. The comparison of the results generated from formal representation semantics of GPM-DEA in OWL/SWRL will also be performed with corresponding rule implementations in platform specific DEA systems.

7.2 Overview of the process model

Some of the critical aspects developed by this research that were discussed in section 5.5.2 are re-instated here as follows. ***These are considered to be an integral part for OWL/SWRL ontology implementation using GPM-DEA method or schema as the basis for DEA with a KBE approach for generative modelling as discussed in section 5.6.3***. These would target engineering design with focus on mechanical design and DFM/DFA with inclusion of both geometric and non-geometric knowledge thus incorporating F-B-S aspects of a process model for DEA.

1. ***Generation of activities based on sub-functions as functional requirements***
2. ***Generation of objects based on sub-functions as functional requirements***
3. ***Generation of engineering rules for activities based on logic as the basis of rules***
4. ***Assessment of initial product to generate the initial activity of the process model***
5. ***Virtual and physical activity functional equivalence***

To test the above formulated criteria, experimental system verification should satisfy the following points in a nutshell –

I. ***Generative Modelling*** - The formal system should generate activities and objects of the engineering design process based on the devised function structures as part of functional requirements. It should also generate rules for activities based on logic. For a generic process an initial step should be assessment of an object as product initial. Also, for a DFM process with manufacturing knowledge, both the physical and virtual representation of a product should be incorporated.

II. ***SWRL Rules*** - The engineering rules that are generated can incorporate product knowledge such as configuration and attributes which can be accessed during detailed geometric modelling such as features, parts, assemblies, location and orientation inside a virtual environment

III. ***Output*** - The output of SWRL rules as platform independent and neutral representation through reasoning and query should produce accurate results, which should match the values upon execution of these rules inside platform specific DEA systems. This will ensure the robustness and reusability of loaded ontology, as the SWRL rules will only produce accurate results if the class hierarchy and properties of ontology with instances has been modelled correctly. If the results of the SWRL rules controlling the product parameters and configuration match to the specific rule outputs inside platform specific DEA systems such as AML, ParaPy and GA based

CATIA Knowledgeware and Siemens NX KF; this will prove that the ontology and rule representation works appropriately.

This will satisfy the aims and objectives by verifying the working of GPM-DEA, which provides the method through schema to use ontologies as platform independent and neutral representation in context of DEA with a KBE approach with semantic clarity, traceability and transparency of concepts and relationships. This will ensure re-usability of modelled knowledge as well.

7.3 Design of the Experimental System

Figure 7-1 illustrates the method of experimental system verification adopted by the author. The first stage consists of the process knowledgebase consisting of mechanical design process with DFM knowledge as high level intermediate and low level concepts formulated as part of this research in section 4.3 of chapter 4. The second stage leads to formulation of GPM-DEA based on the Author's Metamodel as per developed concepts and relationships with generative modelling capabilities for generation of activities and objects based on functional requirements along with rules controlling the product's attributes based on logic and assessment of initial product. This is in line with the development and working of GPM-DEA as described in section 5.3 of chapter 5. The mapping of the various concepts and relationships as shown in Figure 7-1 is described in section 5.2 and 5.3. GPM-DEA is described using a graphical representation as lightweight formalism using DrawIo. This is saved as an XML file. The method of development of GPM-DEA along with its neutral formal representation semantics in this research has been based on the findings of chapter 4 and described in detail in chapter 5. The third stage is the platform independent and neutral formal representation of GPM-DEA using OWL/SWRL formalism as a .ttl file. The equivalent implementation of GPM-DEA in OWL/SWRL as ontology and rule representation is described in section 5.5 of chapter 5 thus providing a method to use ontologies in the

context of DEA. Test use-cases have been elaborated with their formalisation as per the developed GPM-DEA schema or method in chapter 6. Their inference and query results as part of experimental verification of the developed system are discussed here.

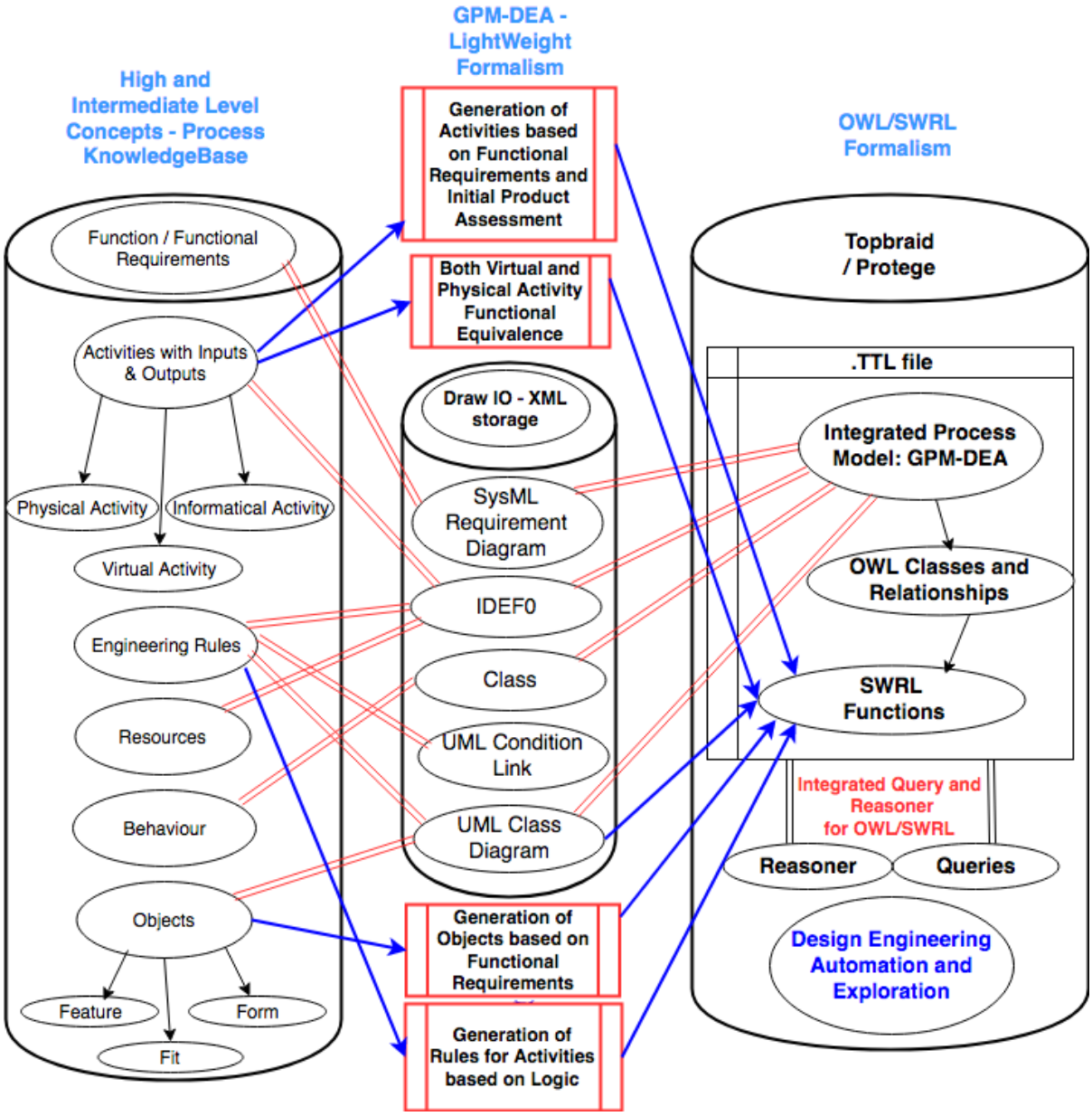


Figure 7-1: Overview of Formalisation of GPM-DEA & Experimental System Investigation

Protégé is a tool that enables an integration of OWL2 ontology and SWRL as a rule language through an in-built interface. This is the most important stage for experimental system investigation and verification in this research. The generative capability of GPM-DEA has been represented using SWRL functions as explained in section 5.5.3 in chapter 5. This is

based on function structures described in section 5.5.2. Similarly, the engineering rules have also been represented using SWRL functions. Querying and inference (automated reasoning) is performed on the integrated knowledgebase as OWL/SWRL with preserved semantics using SQWRL and Drools reasoner on top of SPARQL and Pellet reasoner enabling DEA and exploration. The reasoning results and the query results are added as axioms in the existing knowledgebase and can be saved as new .ttl file. If there are any conflicts in results, modifications can be made in the classes and properties with instances for both text and values such that the reasoner and query can then generate accurate results. All assertions and queries with Pellet reasoner and SPARQL query on OWL2 ontology, Drools reasoner and SQWRL on SWRL rule language have been tested and verified.

7.4 Illustration of Experiments

The following structured experiments have been devised to test and verify various research aspects of this thesis. These will be tested with the drilling process and bookshelf design process ontology and rule representation along with the discussion on results.

1. ***Generative Modelling Capability*** - Do the SWRL functions represented through the inbuilt plugin enable generative modelling by*automatically generating activities and objects* that fulfil the same *sub-functions as functional requirement* of the design process along with assessment of the initial product? This includes virtual and physical activity functional equivalence and generation of engineering rules for activities based on logic as the basis of rules.
2. ***SWRL Rules with Variation in Values*** - Do the *SWRL engineering rules* represented through the inbuilt plugin *add axioms* on to the existing knowledgebase with both *object and datatype properties* with real and float values to product attributes? How does the system *handle variation in values* assigned?

3. ***SQWRL Query with Violation in Asserted Values***- Does the *SQWRL return the correct result while querying* the knowledgebase? How does the *system handle the violations in assertions* against the engineering rules?

4. ***Comparison of SWRL and SQWRL Rule Outputs to Platform Specific DEA Systems***- Does the *SWRL/SQWRL outputs match to the outputs of axioms inside a DEA system*?

7.5 Use Case 3: Experimentation

The first step in the experimental verification of the developed GPM-DEA in OWL/SWRL for design process of drilling a hole in a block is the deployment of the instantiated model. As observed from Figure 6-11 in section 6.3.3 in chapter 6, the generative modelling functions for drilling use case have been represented using SWRL. Figure 7-2 shows the loaded ontology in Protégé where the drilling process has 3 functional requirements with the axiom – satisfies_Functional_Requirement (section E).

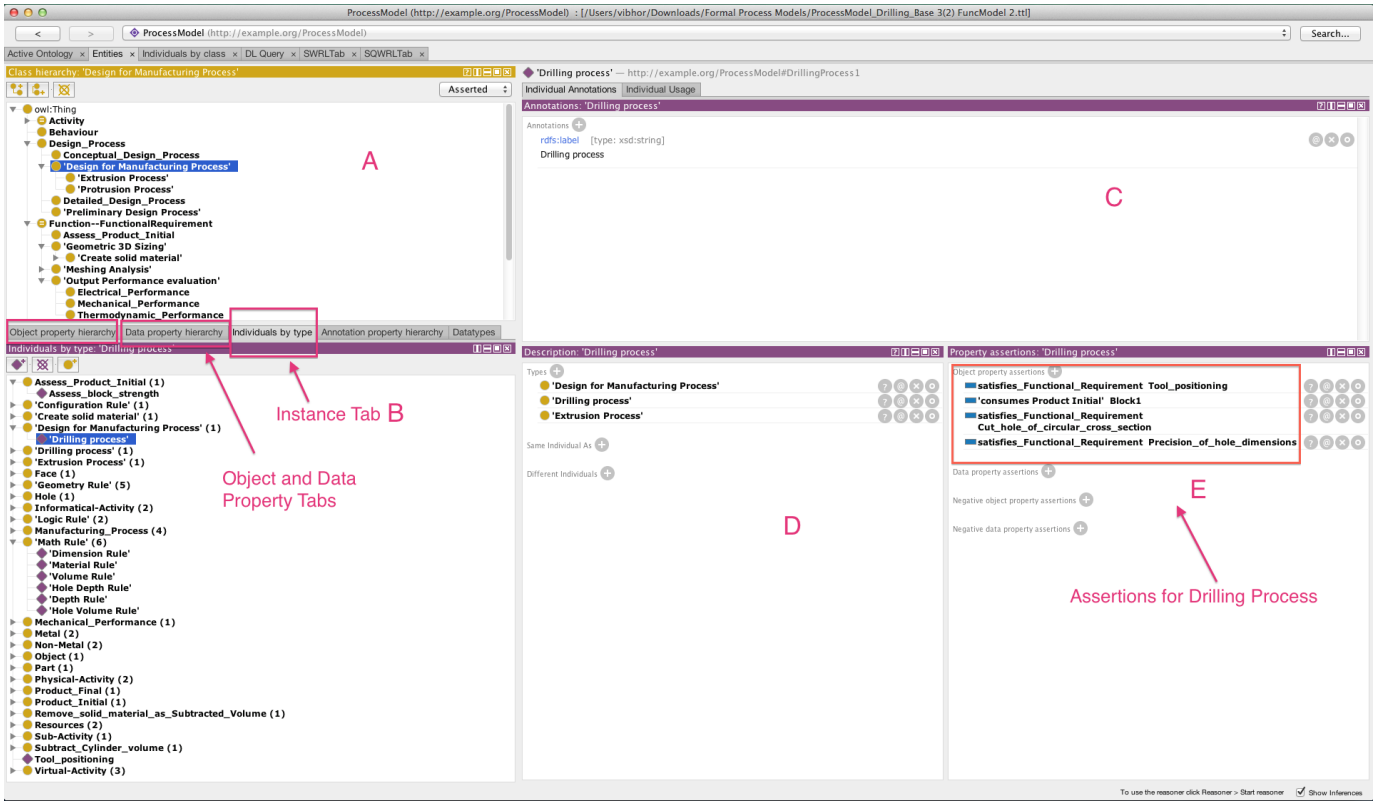


Figure 7-2: Drilling Process Ontology: Protégé

All the classes with hierarchical structure (section A) and binary relationships of GPM-DEA for drilling process as properties have been instantiated (section B) and populated in the IDE as axioms. It illustrates both physical and virtual view of drilling process by allocating it as a subclass of extrusion process as well (section D). Text annotation properties (section C) provide semantic clarity to the axioms. Using this standardised tab, other instances can be populated in the corresponding tabs in protégé IDE. All the experiments for drilling process in a block are discussed in this section.

7.5.1 Experiment 1 – Generative Modelling Capability

Figure 7-3 shows a snapshot where assertions have been made for ‘drill hole’ and ‘assess block’ activity as marked in red rectangles. Assertions have been made for the functional requirements as sub functions of the activity, which will be tested in this section. The first step is to activate the Pellet Reasoner followed by the Drools reasoner. Figure 7-4 illustrates the tab that enables this functionality in the protégé IDE.

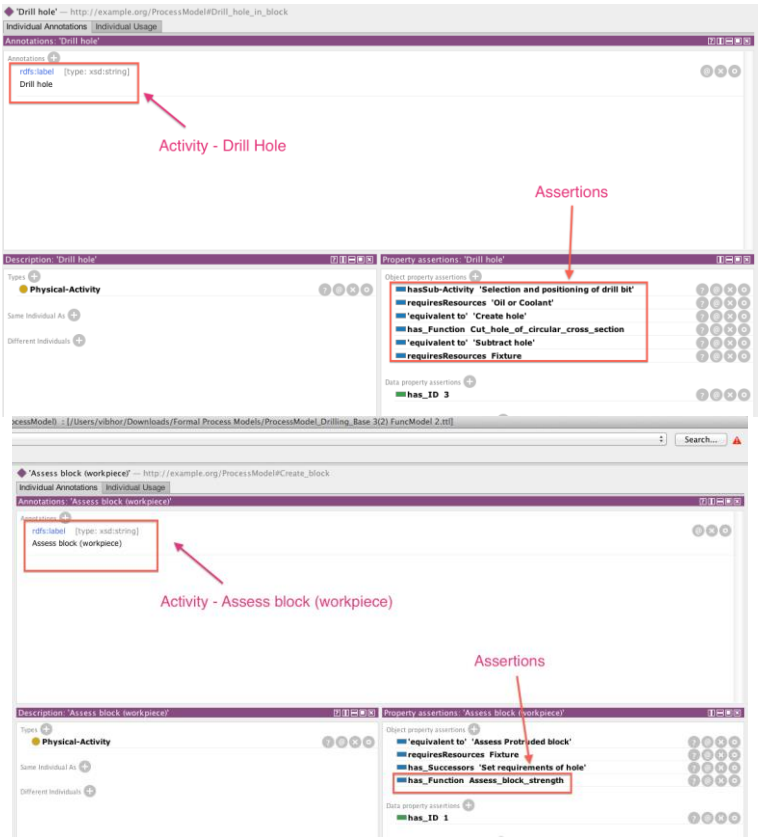


Figure 7-3: Axioms assertion for Drill Hole and Assess Block Activity with Sub-functions

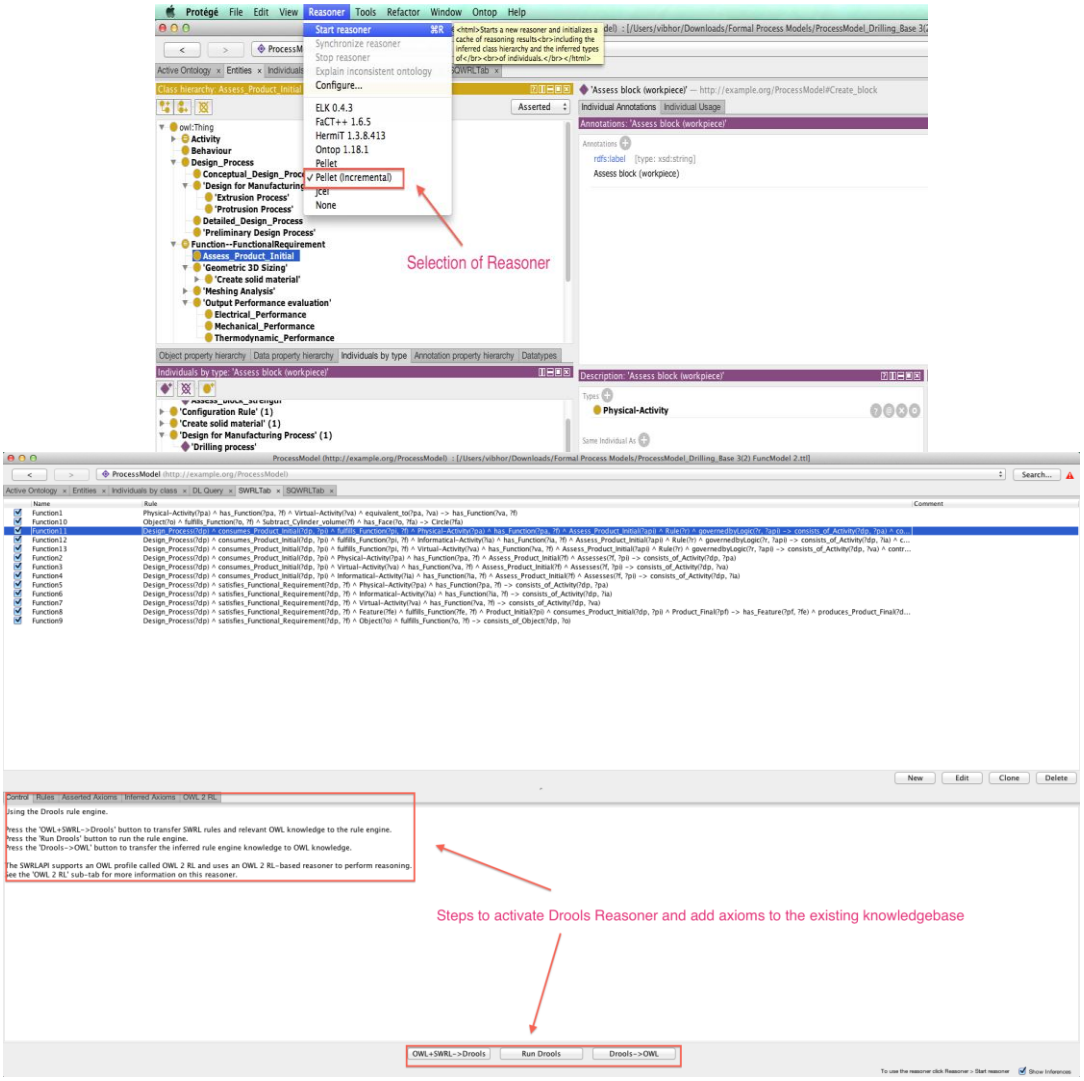


Figure 7-4: Activating the Pellet and Drools Reasoner

It can be observed from section E of Figure 7-2 and Figure 7-3 assertions, both the activities in the form of – 'assess block' and 'drill hole' satisfy sub-functions, which are equivalent to the function structures as part of functional requirements of the drilling process. As observed from Figure 7-5, upon activating both Pellet and Drools reasoners, all the activities in the knowledgebase which match the drilling process functional requirements have been added as axioms due to the SWRL generative modelling functions developed in this research. As per the assessment of the block as the initial product, the axiom – 'Starts_with_Activity' indicates that the drilling process for block needs to start with the activity 'Assess block', which has an equivalent virtual representation in the form of 'Assess protruded block'.

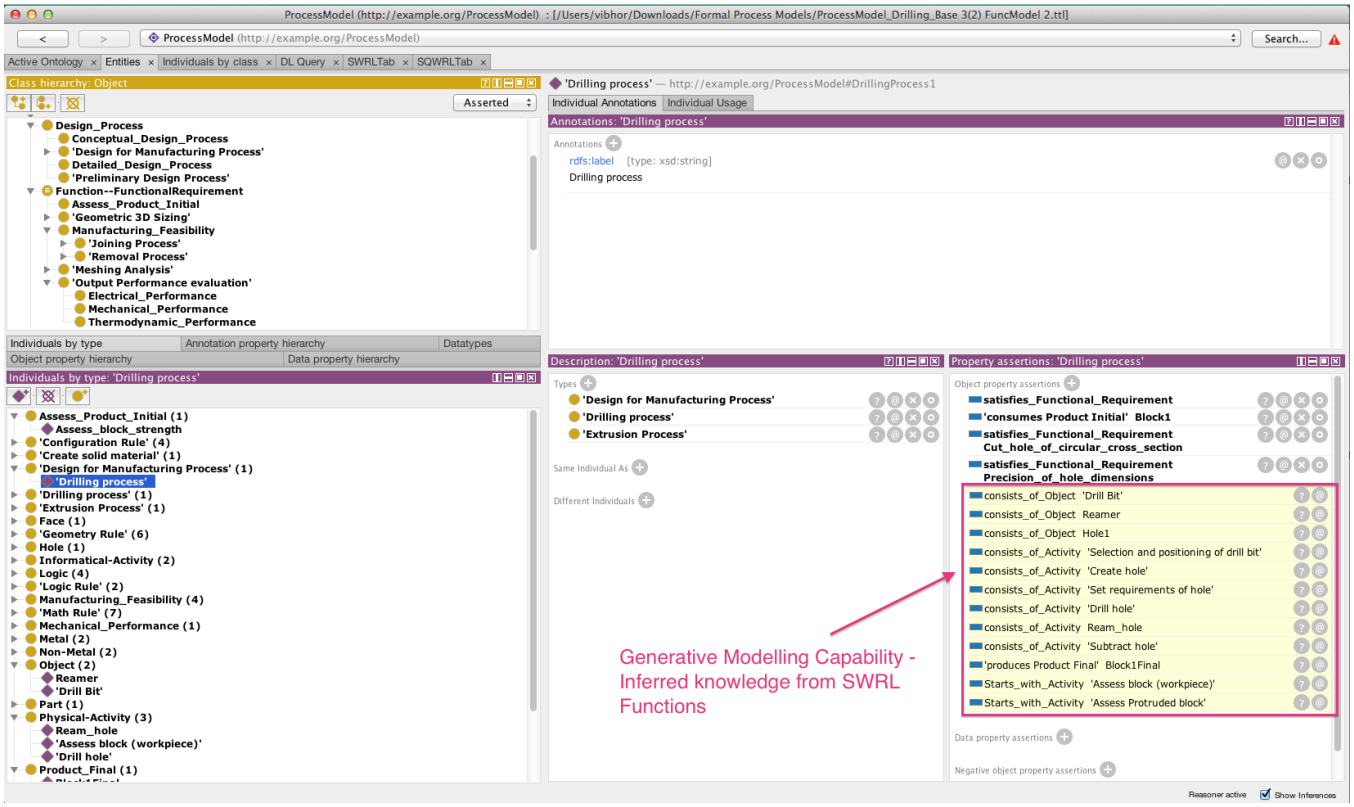


Figure 7-5: Generative Modelling Capability - SWRL functions activated for drilling process ontology for Block

It is important to notice that 'Drill hole' is a physical activity, which also has equivalent virtual activities in the form of –'Create hole' and 'Subtract hole', which are realised in the virtual engineering environment. These activities fulfil the same functions due to the *SWRL Function 7* stated in section 5.5.3 in chapter 5 and implemented for drilling process for block in section 6.3.3 in chapter 6. As observed from Figure 7-5, these virtual activities are also automatically generated for drilling process due to the inference on *generative modelling functions*. Thus GPM-DEA provides both physical and the virtual representation of the drilling process in terms of design process and manufacturing process requirements with the SWRL functions developed as part of this research. Figure 7-6 illustrates inferred knowledge with Pellet reasoner for 'Assess block' and equivalent 'Assess protruded block' as initial activities as well as 'Drill hole' and other activities of the drilling process for block. The SWRL functions are illustrated in section 5.5.3 in chapter 5 and implemented for this use-

case in section 6.3.3 in chapter 6. Some engineering rules are governed by logic such as Dimension, Material, Hole Depth, Hole Diameter Rule and others in this case. Heuristic rules are not governed by logic and are disjoint from this relation.

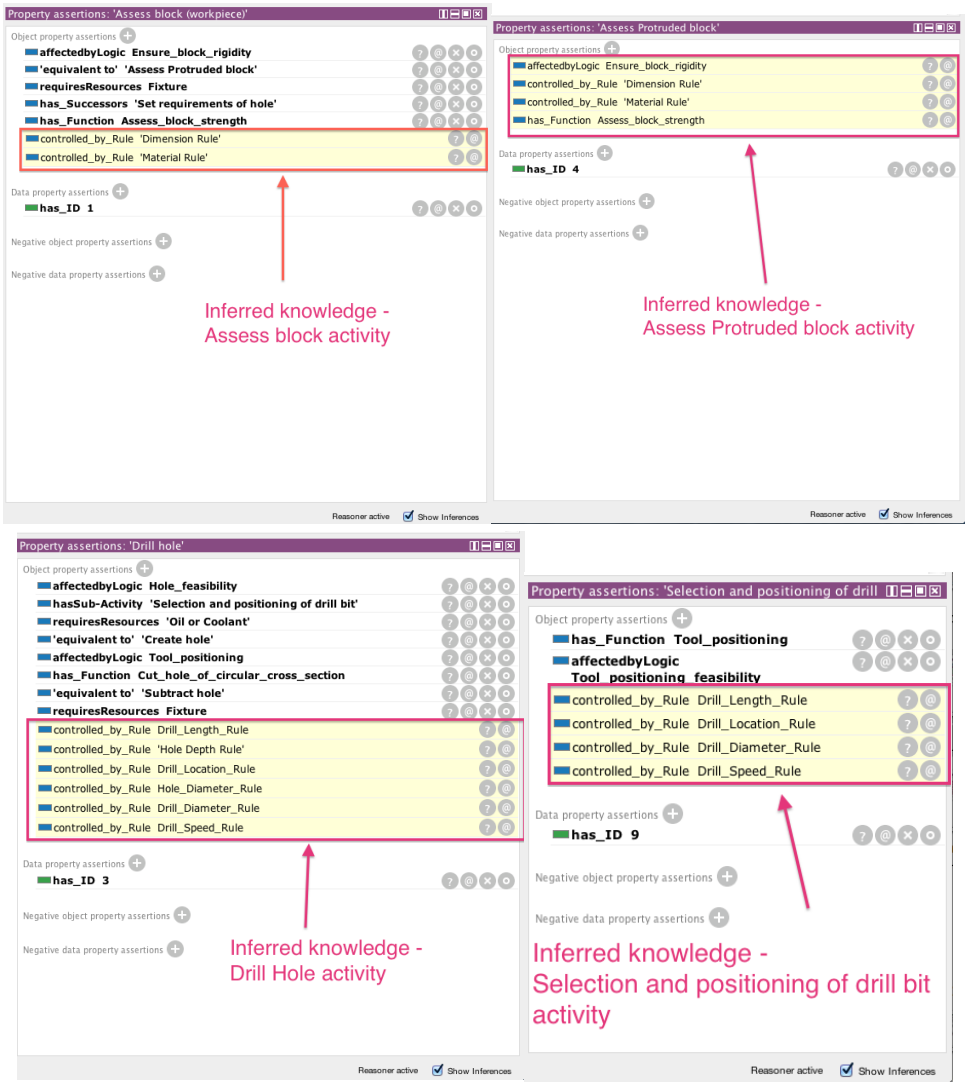


Figure 7-6: Inferred knowledge – Drilling Process Activities

7.5.2 Experiment 2 – SWRL Rules with Variation in Values

As observed from inferred knowledge in Figure 7-6, 'Assess block' activity is controlled by the following rules – 'Dimension, Material' due to logic relation. However, as observed from the graphical representation of Drilling process1 in Figure 6-6 in chapter 6, 'Assess block' activity is also controlled by the Depth rule. This relation was not inferred, as Depth rule is not associated with logic in the knowledgebase. 'Assess block' activity has equivalent virtual

activity – ‘Assess protruded block’ which is also controlled by ‘Volume rule’, as the block occupies 3D volume in a virtual domain. Figure 7-7 illustrates the loaded block and hole attribute values along with its position coordinates as the initial product for the drilling process.

Figure 7-7: Asserted and Inferred values to Block and Hole attributes - Drilling Process Ontology / SWRL Rules for Block

As also observed from inferred knowledge in Figure 7-6, some of the rules that control the ‘Drill hole’ activity are Hole Depth and Hole Diameter Rule. All these rules are explained in section 6.3.3 in chapter 6. Figure 6-16 and 6-17 shows the SWRL representation of these engineering rules for the drilling process ontology.

Figure 7-7 also illustrates the inferred knowledge in the form of *Block Depth* and *Volume* along with *Volume of Hole* when the Drools reasoner is activated for the SWRL rules. As the asserted width of the block is less than 100.00 mm, no material is allocated to the block as per the Material Rule. Upon changing values of Block and Hole in terms of its size and coordinates using datatype properties, changes in output values to Block Depth, Volume and Hole Volume can be observed from Figure 7-8.

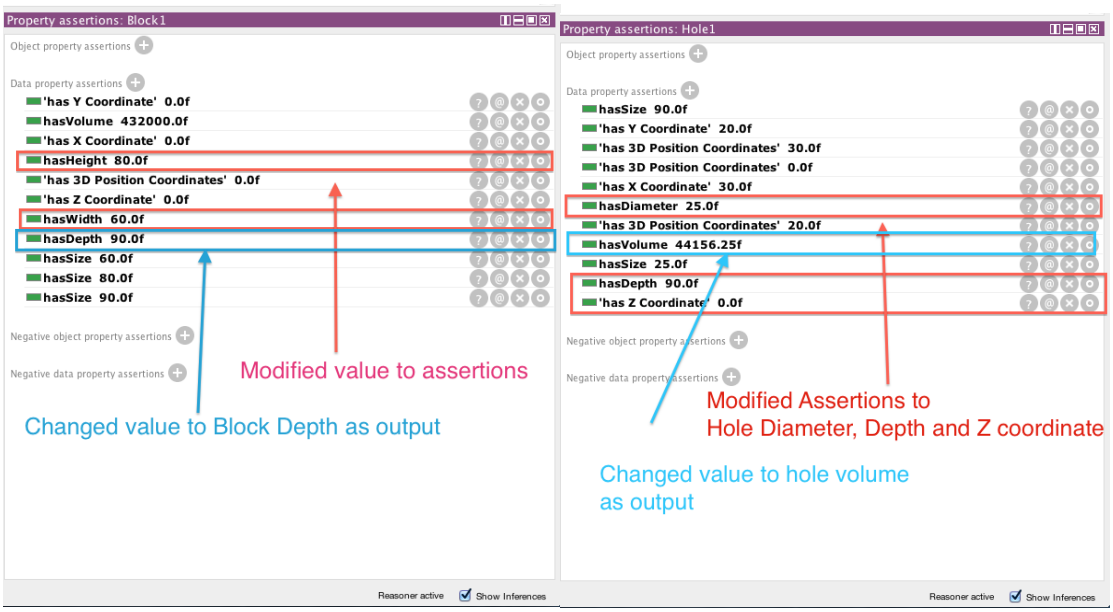


Figure 7-8: Modification in Asserted Values with Variation in Output Values - Drilling Process Ontology / SWRL Rules for Block

Figure 7-9 shows the implementation of Process Rule1 based on the Tolerance of hole as asserted value. According to the semantics of the Process Rule shown in Figure 6-18 in chapter 6, *if the tolerance of the hole is less than 0.2mm reaming* should be performed, *else drilling* should be performed.

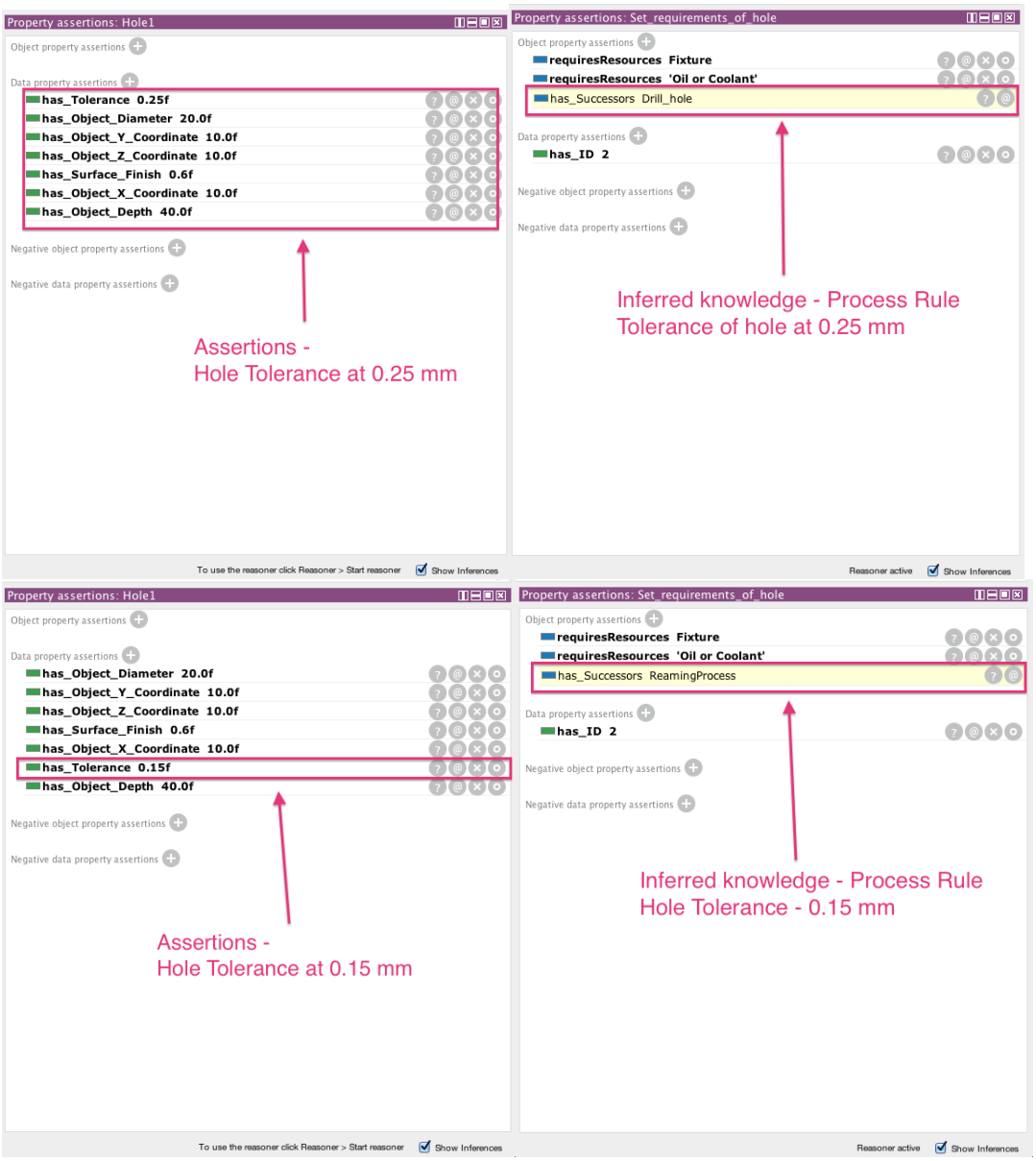


Figure 7-9: Process Rule1: Drilling Process SWRL

7.5.3 Experiment 3 – SQWRL Query with Violation in Asserted Values

The SQWRL runs the query on the OWL knowledgebase as the SWRL API supports an OWL profile as OWL 2 RL based reasoner in the form of drools (Horridge et al., 2011; Kuba, 2012). For the asserted value to block and hole attributes in Figure 7-7, the query results for all the 3 rules are illustrated with the help of Figure 7-10. All the results are satisfied as none of the asserted values violate any of the engineering rules as represented in SWRL.

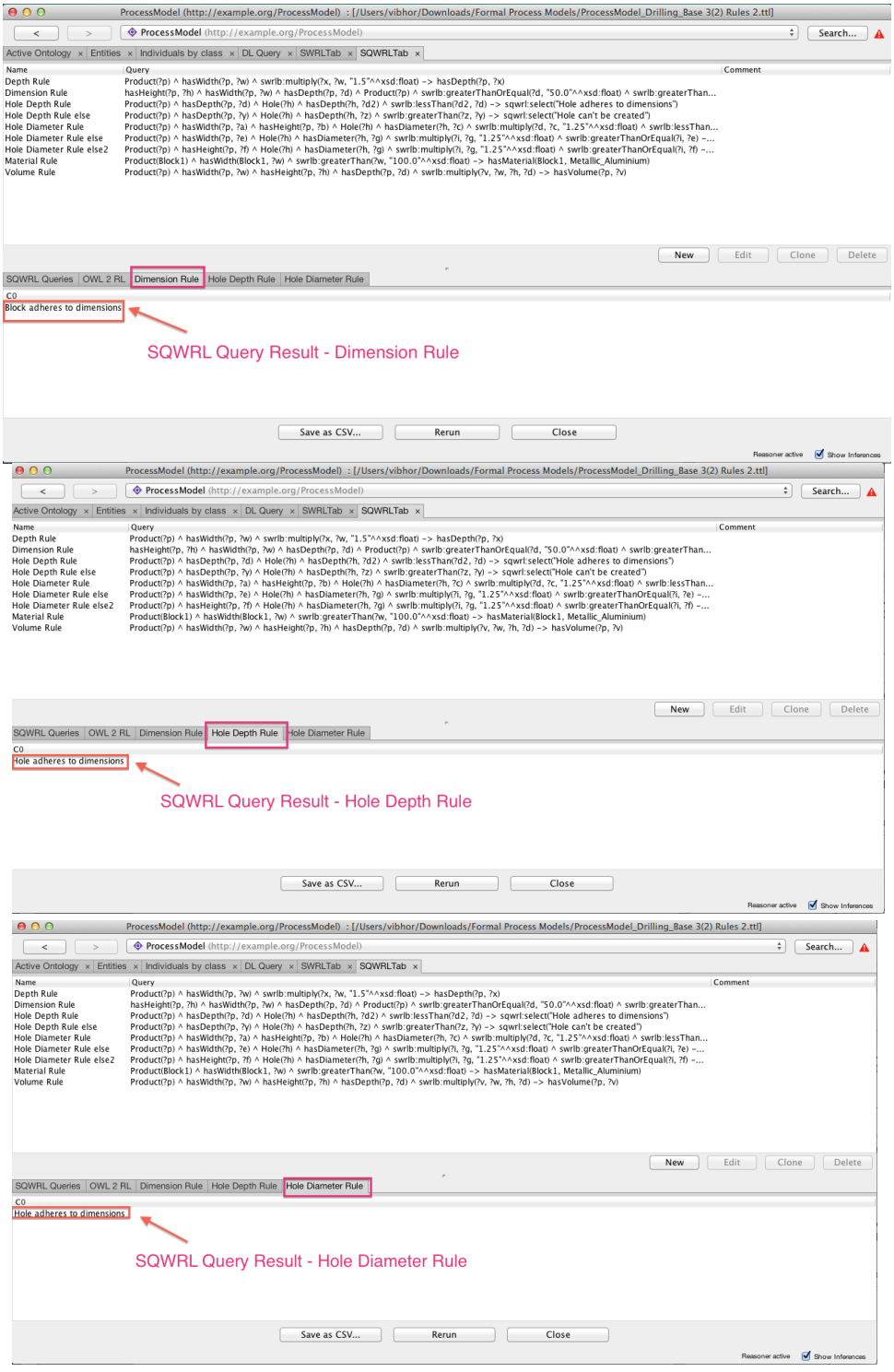


Figure 7-10: Query Results: SQWRL Rules

'Assess block' activity is controlled by Dimension rule whereas 'Drill hole' activity is controlled by Hole Depth rule. A few violations in terms of *block width value 49.0 mm (<50.0 mm as per dimension rule)* and *hole depth value 76.0 mm (>{1.5*50}=75.0 mm [block depth])* are asserted as shown in Figure 7-11. As illustrated, the activity – 'Drill hole'

is also controlled by Hole Diameter rule. A violation is asserted to hole attributes in terms of *hole diameter 50.0 mm ({50*1.25}=62.5>60.0 mm [block width])* as shown in Figure 7-11. All the SQWRL query results are illustrated in accordance with violation of Dimension, Hole Depth and Hole Diameter Rule in line with semantic clarity.

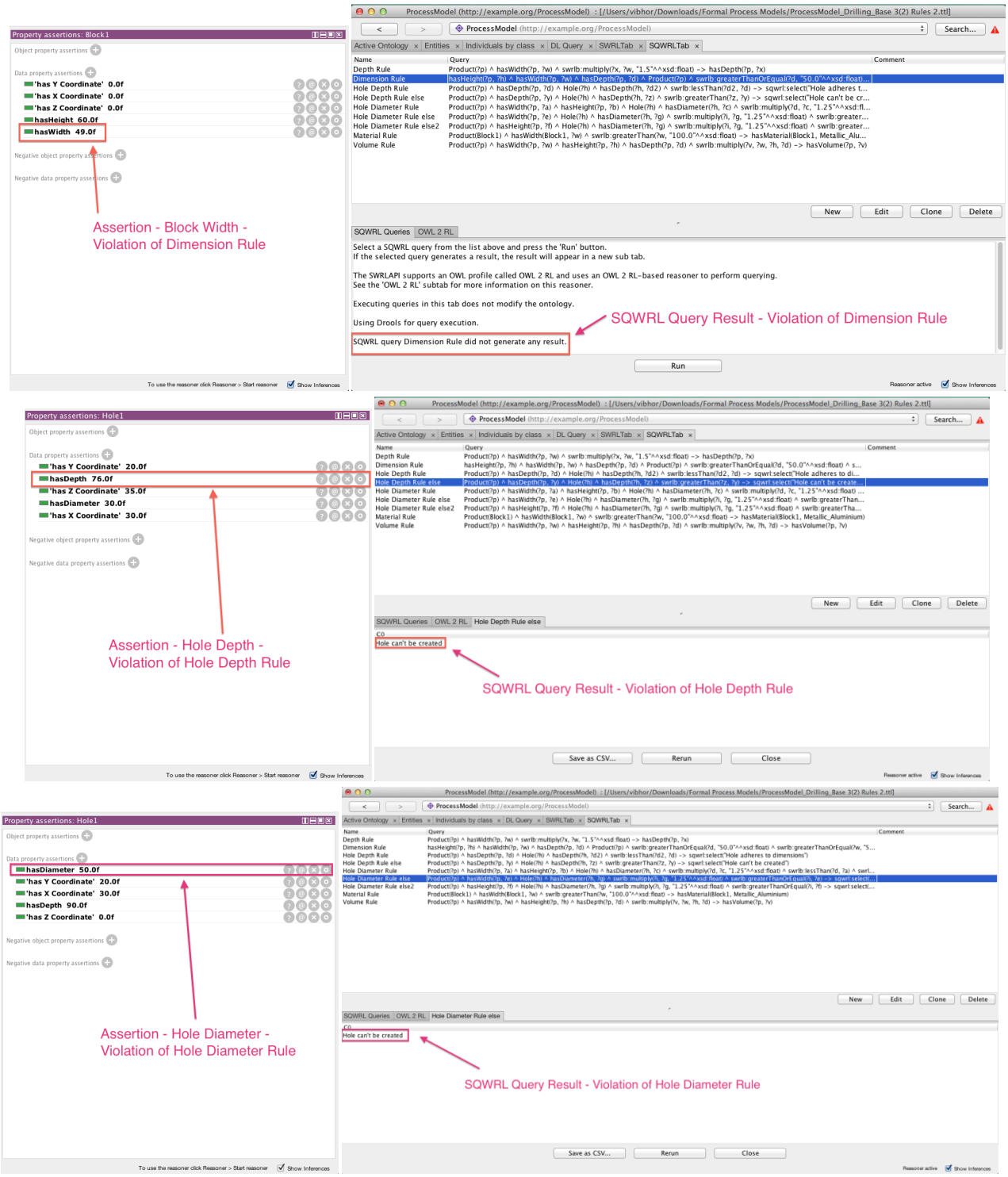


Figure 7-11: Violation of Asserted Axioms against Dimension Rule, Hole Depth Rule and Hole Diameter Rule – OWL/SWRL

7.5.4 Experiment 4 – Comparison of SWRL and SQWRL Rule Outputs to Platform Specific DEA Systems

An instance of the rules of the drilling process for block have been represented and codified inside ParaPy as a platform specific DEA system by the author. Similar variations to values as OWL/SWRL have been performed inside ParaPy and the results have been compared in this section. It is important to note that ParaPy is based on inbuilt classes and has a built in Graphical User Interface (GUI) geometry modeller to reflect the changes in product's state whereas the present OWL/SWRL representation reflects the changes in the query (SQWRL) and reasoning (SWRL) tab without the visual representation of product's geometric state. As observed from Figure 7-7 in section 7.5.2, same values have been instantiated for both block and hole inside ParaPy as observed from Figure 7-12 and 7-13. The representation of engineering rules follows O-O representation in the form of a method.

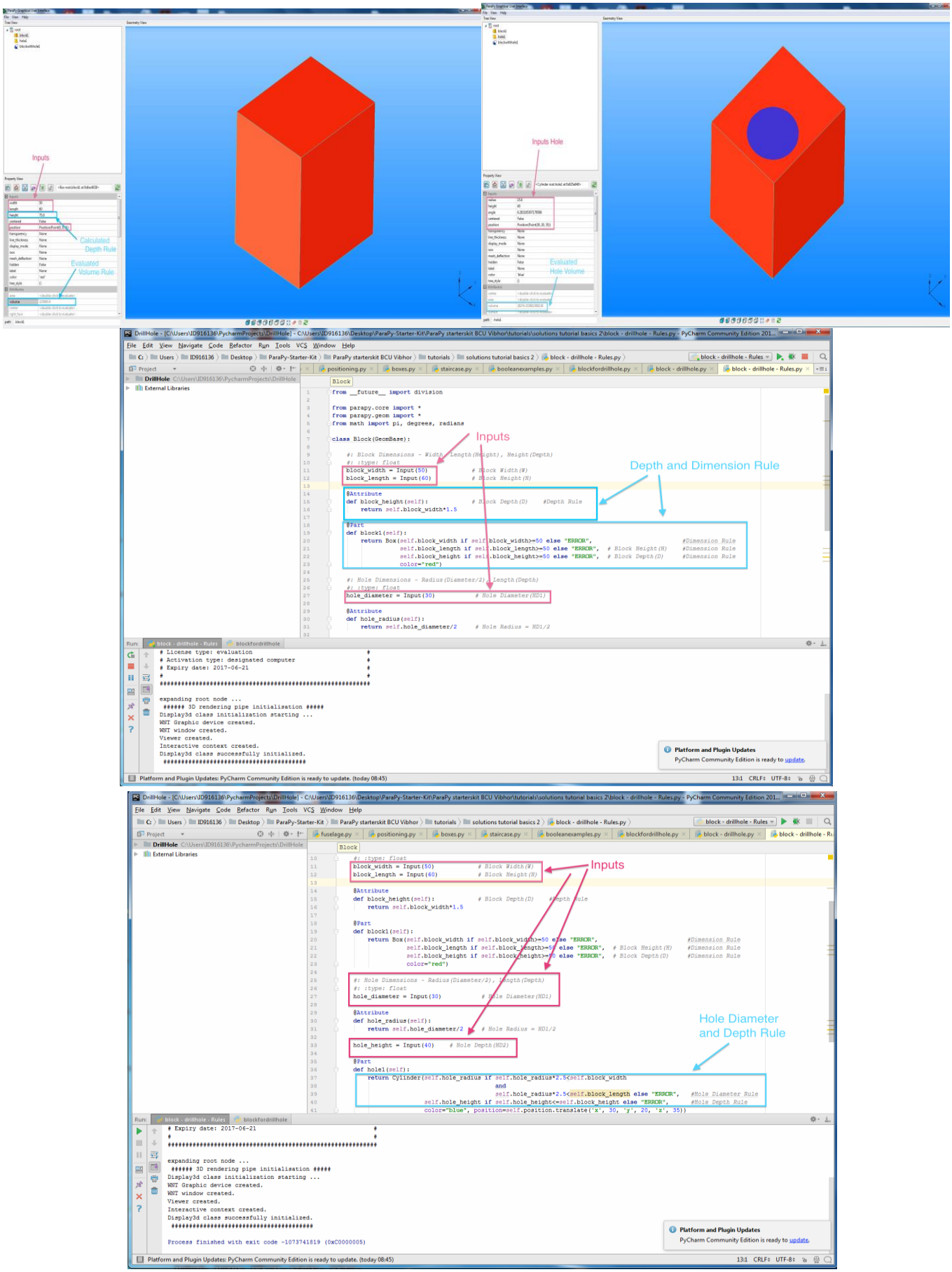


Figure 7-12: Inputs and Evaluated values inside ParaPy: Drilling Process – Block

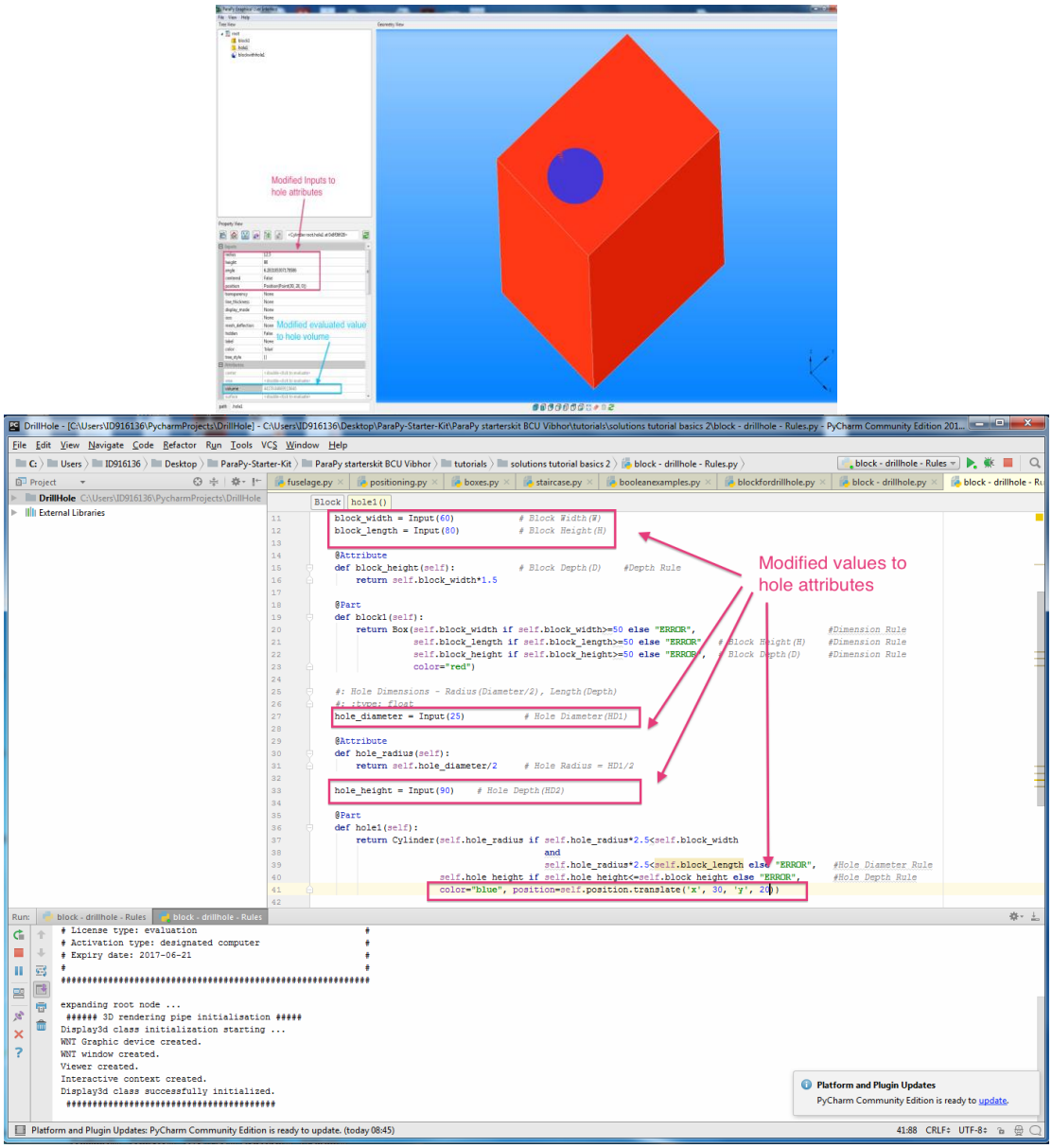


Figure 7-13: Inputs and Evaluated values with modifications to asserted values inside ParaPy: Drilling Process – Block

It can be observed that the calculated and evaluated values for block and hole attributes are same as the values inside OWL/SWRL (platform independent and neutral representation) inferred knowledge in Protégé IDE. Similarly, upon modifications in the asserted values to hole attributes, which are same as those in OWL/SWRL representation of drilling process in Figure 7-8, same values are evaluated inside ParaPy as a platform specific DEA system as observed from bottom Figure 7-13. However, there is a slight difference in the *volume of the*

hole as calculated in *SWRL as 28260.0 mm^3 and 44156.25 mm^3* as observed from Figure 7-7and 7-8 against the calculated value of *28274.33 mm^3 and 44178.64 mm^3* as observed from Figure 7-12 and 7-13. This is due to the fact that a value of 3.14 is used in SWRL rule, which is rounded up to two decimal places against the actual value of π (3.141592653589793238) inside ParaPy. A few violations are introduced for the block attributes (*Block Width=49.0 mm< 50.0 mm as per Dimension Rule*), hole attributes (*Hole Depth=76.0 mm>{1.5*50}=75.0 mm [Block Depth] as per Hole Depth Rule*), Hole Diameter (*Hole Radius=25.0 mm {2.5*25}=62.5>60.0 mm Block Width*), all of which are of same value in OWL/SWRL in Figure 7-11.

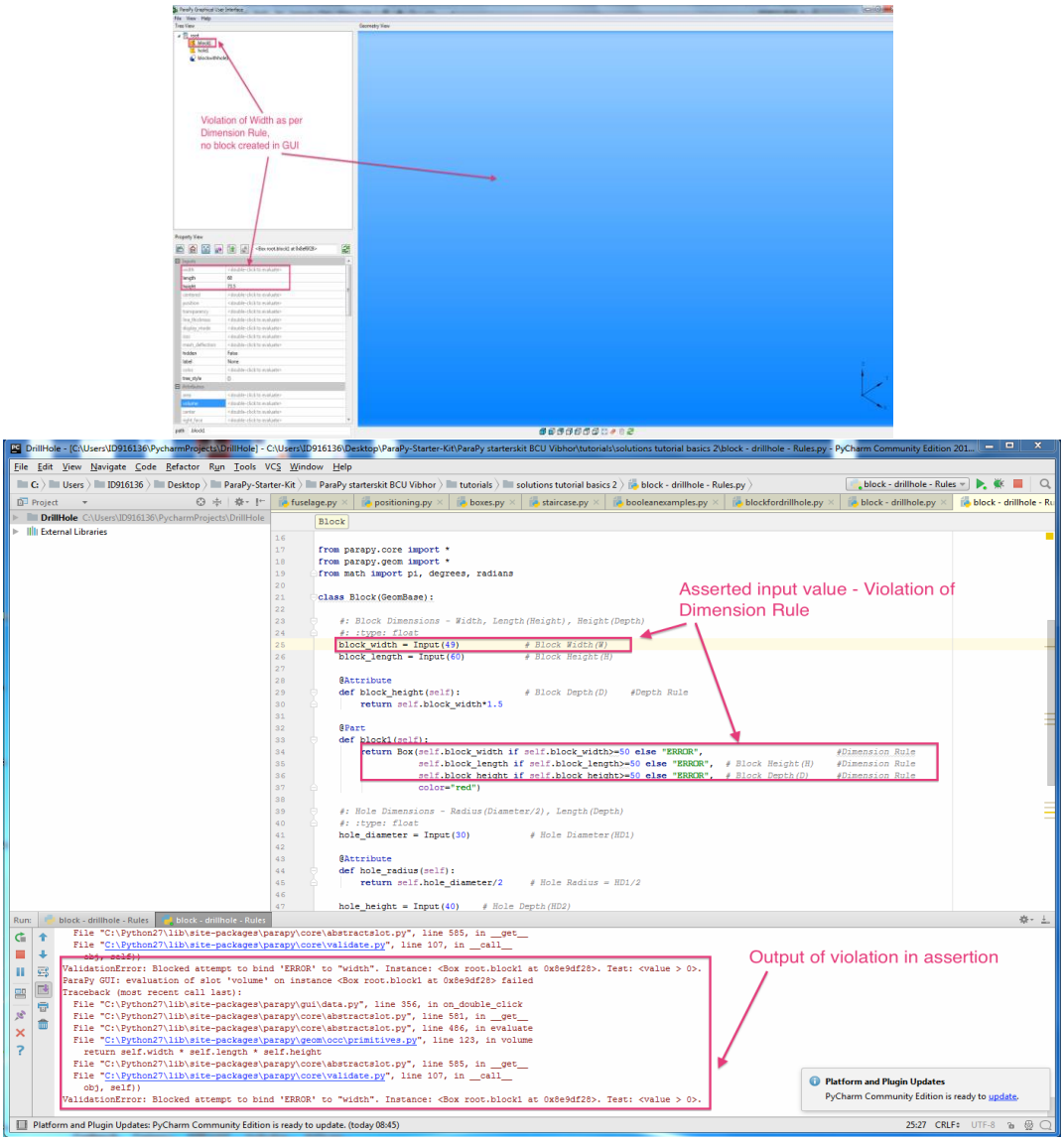


Figure 7-14: Violation of Asserted Axioms against Dimension Rule - ParaPy

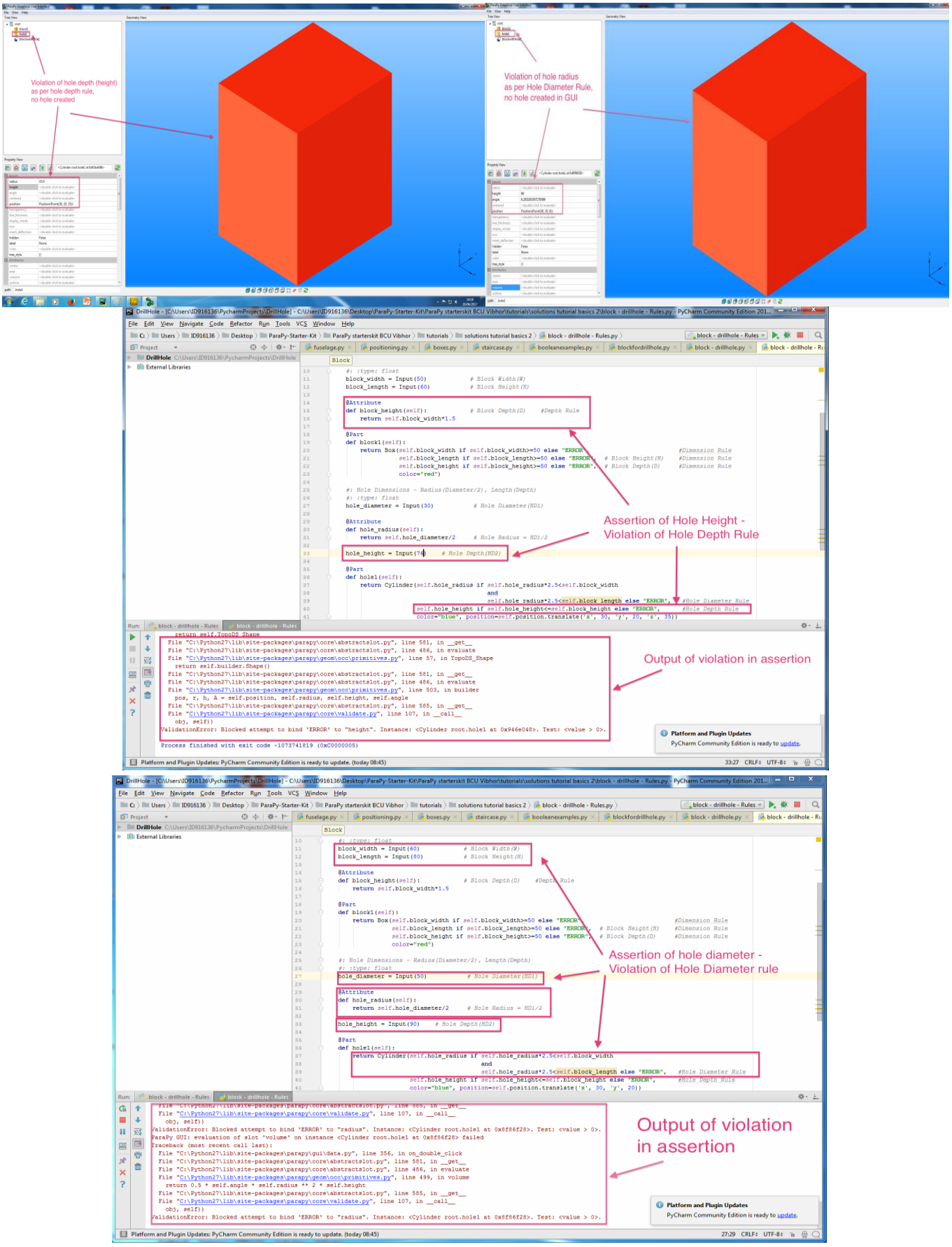


Figure 7-15: Violation of Asserted Axioms against Hole Depth and Hole Diameter Rule – ParaPy

Figure 7-14 illustrates the output in ParaPy as no block is created in the graphical user interface (GUI) for violation of Dimension rule. Similarly, as observed from Figure 7-15, no hole is created in GUI for violation of Hole Depth and Hole Diameter Rule. The text in the run and compiler window also indicates that it could not bind the value to the Block Width, Hole Height (Depth) and Hole Radius respectively, which are the same results in the query tab in SQWRL in Figure 7-11.

7.6 Use Case 4: Experimentation

This use-case has been derived from the LinkedDesign project (Lützenberger et al., 2012) as illustrated in chapter 6 with addition of knowledge to develop a more comprehensive knowledgebase for this research. The main purpose of this use-case is to ensure the proposed working of GPM-DEA through its OWL/SWRL representation for a bookshelf design process, which varies from the design process of drilling a hole in a block and thus creates a different product. Similar steps and experiments have been conducted for this use-case to illustrate the generic and uniform working of the developed process model GPM-DEA enabling DEA through its neutral formal representation. This further strengthens the research hypothesis and provides verification to the research objectives.

The first step in the experimental verification is the deployment of the instantiated model. As observed from Figure 6-28 in chapter 6, the *generative modelling functions for bookshelf design process*use case have been represented using SWRL. Figure 7-16 shows the loaded ontology in Protégé where the bookshelf design process has 3 functional requirements with the axiom – satisfies_Functional_Requirement (section E).

All the classes (section A), binary relationships of GPM-DEA and text annotation properties (section C) for bookshelf design process as properties (section B) have been instantiated and

populated in the IDE as axioms. All the experiments for bookshelf design process are discussed in this section.

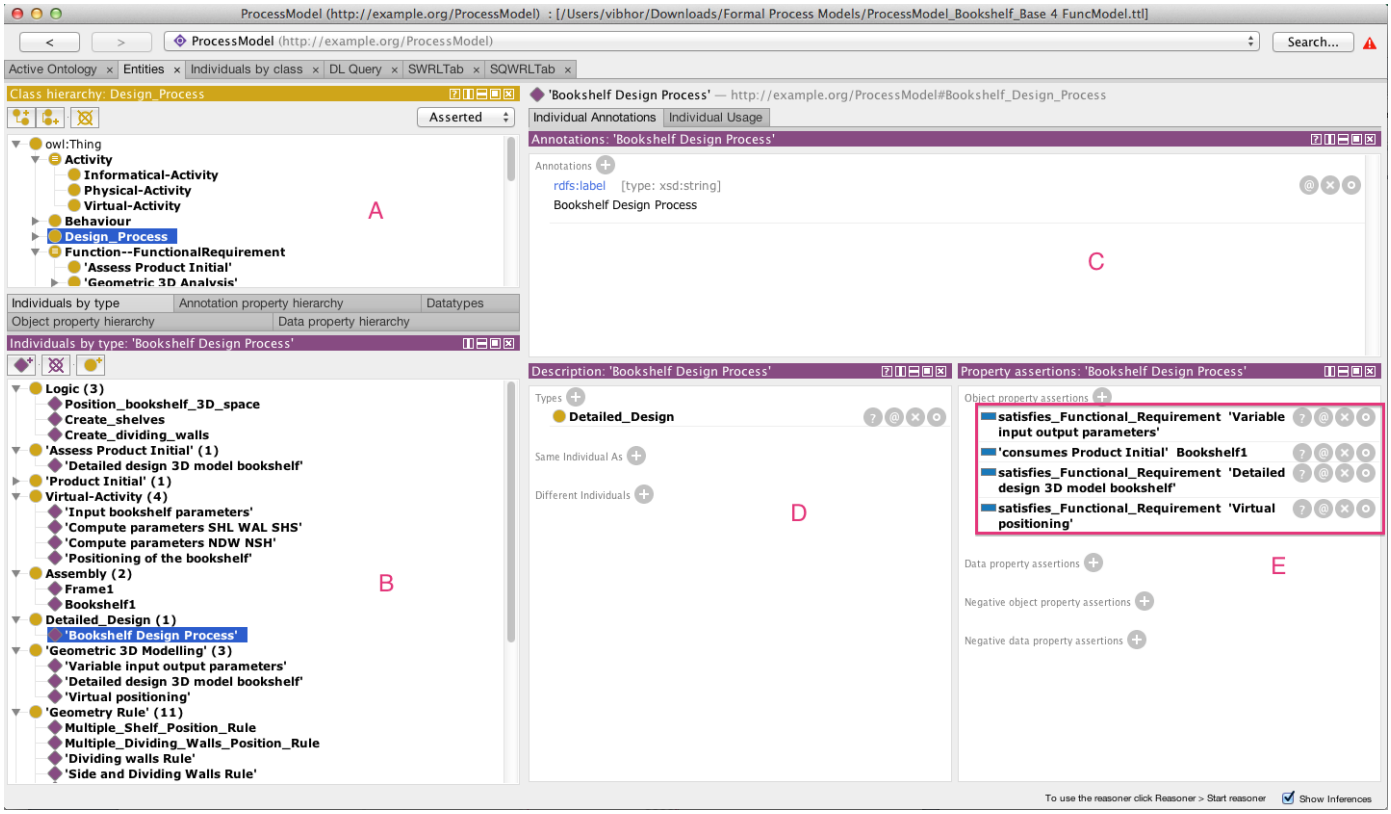


Figure 7-16: Bookshelf Design Process: Ontology

7.6.1 Experiment 1 – Generative Modelling Capability

Figure 7-17 illustrates the asserted axioms for activities such as 'Input bookshelf parameters', 'Compute parameters NDW NSH' and 'Positioning of the bookshelf'. Sub-functions for these activities have been instantiated using object property – 'has_Function' and bookshelf attributes have been allocated using datatype properties as a subclass of – 'has_Inputs' as explained in section 6.2.2.3 in chapter 6.

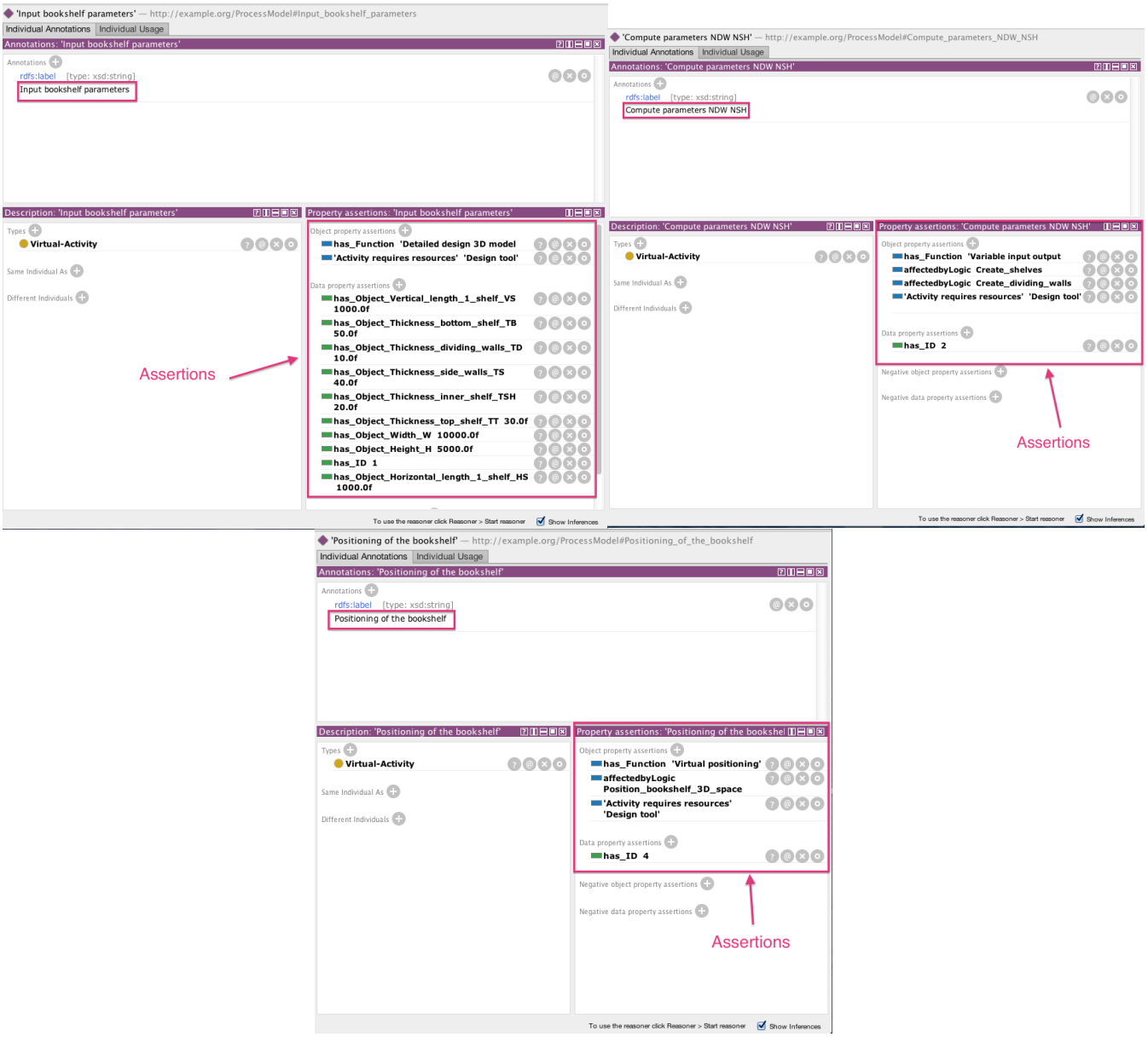


Figure 7-17: Axiom Assertions for Activities: Object and Datatype properties

Generative modelling capabilities of this research are illustrated with the help of Figure 7-18. Upon activating the Pellet and Drools reasoner, all the activities are inferred as the individual activity sub-functions match the functional requirements of the bookshelf design process. As per the assessment of initial product based on SWRL functions, '*Starts with activity'* axiom is also inferred. Similarly, engineering rules such as 'Dividing walls Rule', 'Shelves Rule', 'Side walls position Rule' are governed by logic, which is represented as text under 'Logic'

class. As individual activities are also affected by this logic, SWRL functions infer the rules controlling the individual activities.

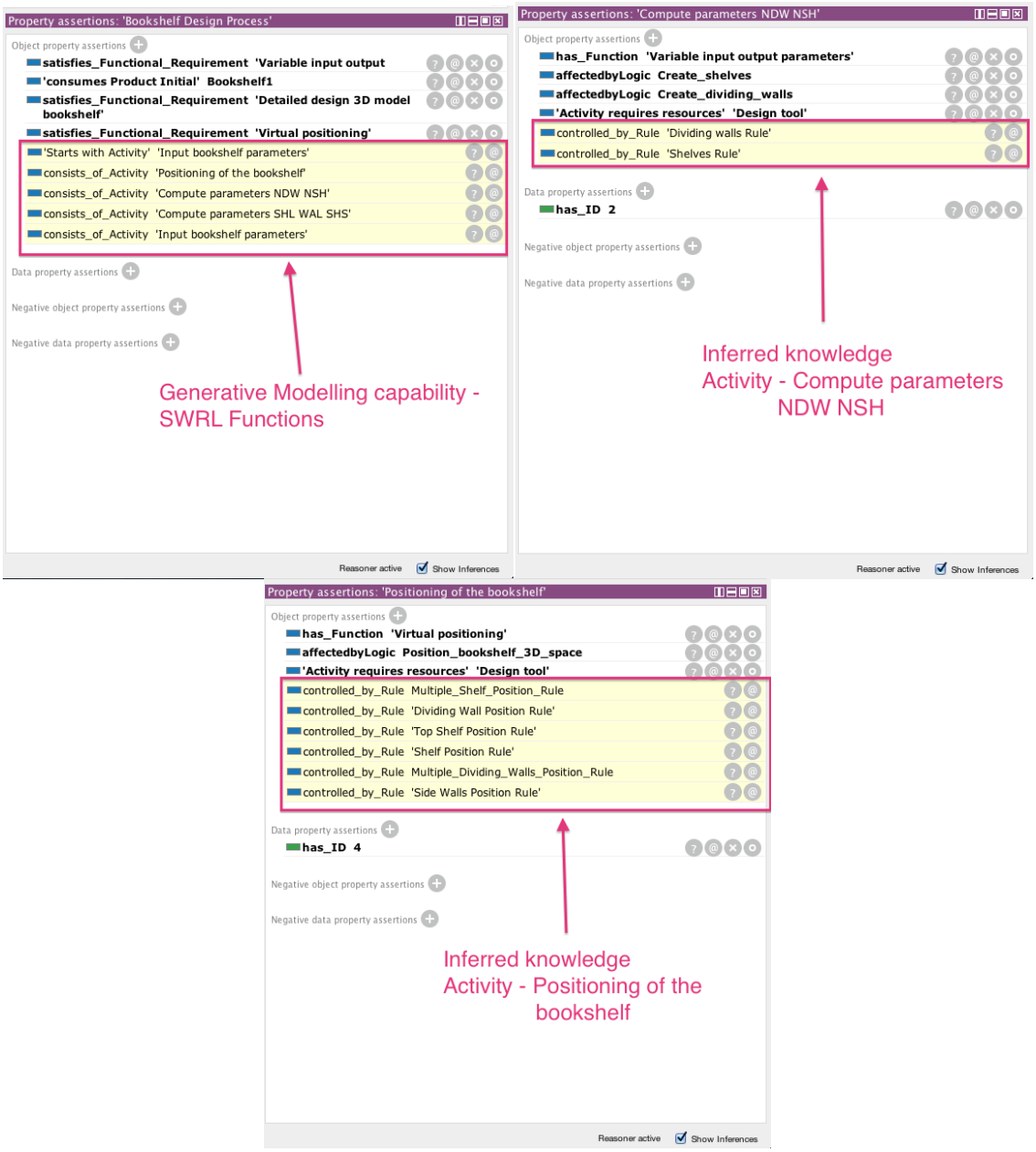


Figure 7-18: Generative Modelling and Inferred Knowledge – Bookshelf design process – SWRL functions

7.6.2 Experiment 2 - SWRL Rules with Variation in Values

As observed from Figure 7-18, the rules controlling the activities based on logic are inferred. Values are asserted to bookshelf attributes using inputs property as shown in Figure 7-19. The SWRL rules for the bookshelf are represented in the SWRL tab as illustrated in Figure 6-

33 in chapter 6. Upon activating the pellet and drools reasoner and addition of axioms to the knowledgebase enables deduction of other attributes based on all the generated rules at specified asserted values, which are inferred as shown in Figure 7-19 along with other rules which are not based on logic.

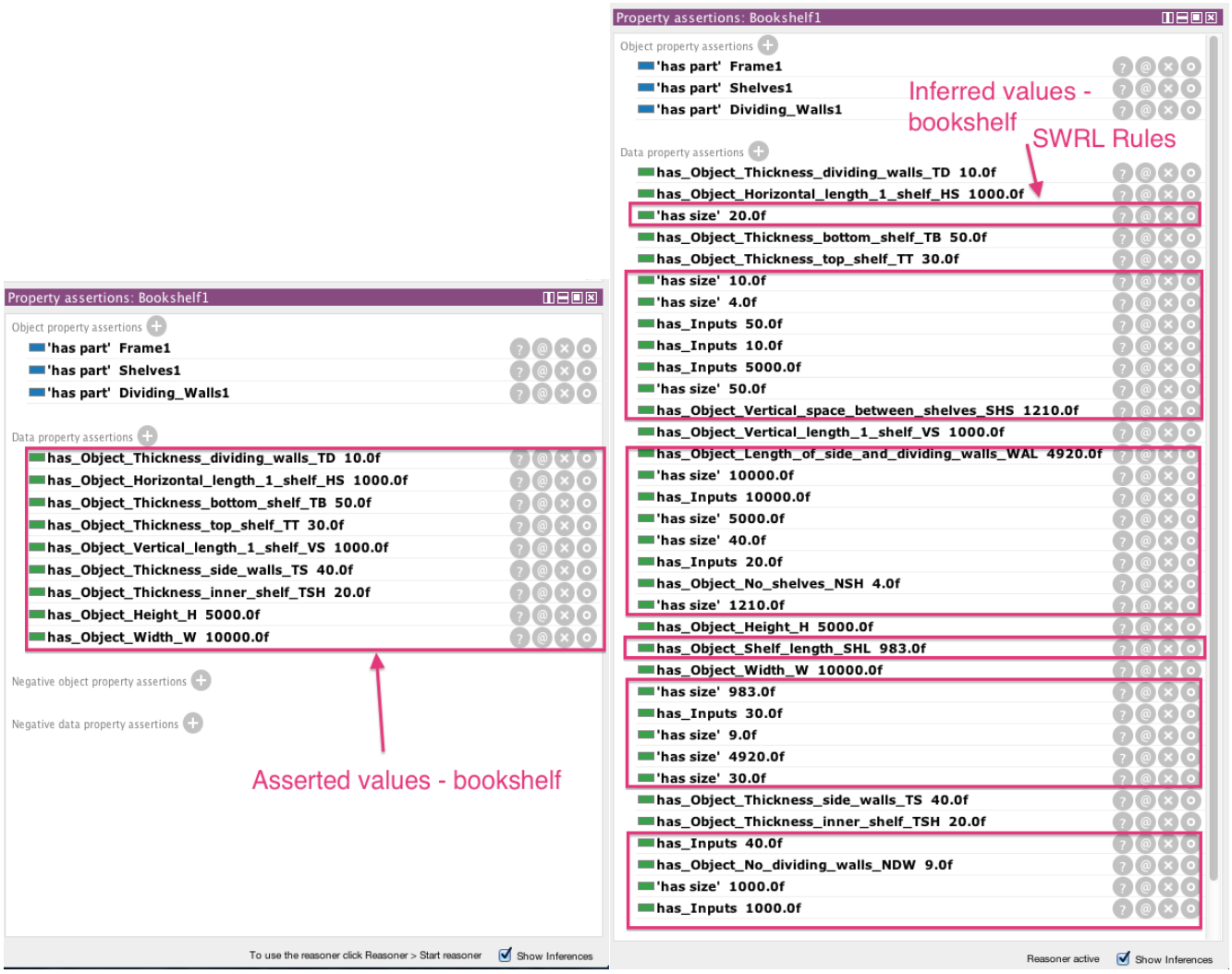


Figure 7-19: Asserted and Inferred values to Bookshelf Attributes: SWRL Rules

Similarly, the asserted and inferred values to subassembly components of Bookshelf such as the Dividing walls, Shelves and Frames are also shown in Figure 7-19. All the corresponding informal part and assembly relations of the bookshelf are illustrated in Figure 6-23 and 6-25 in chapter 6.

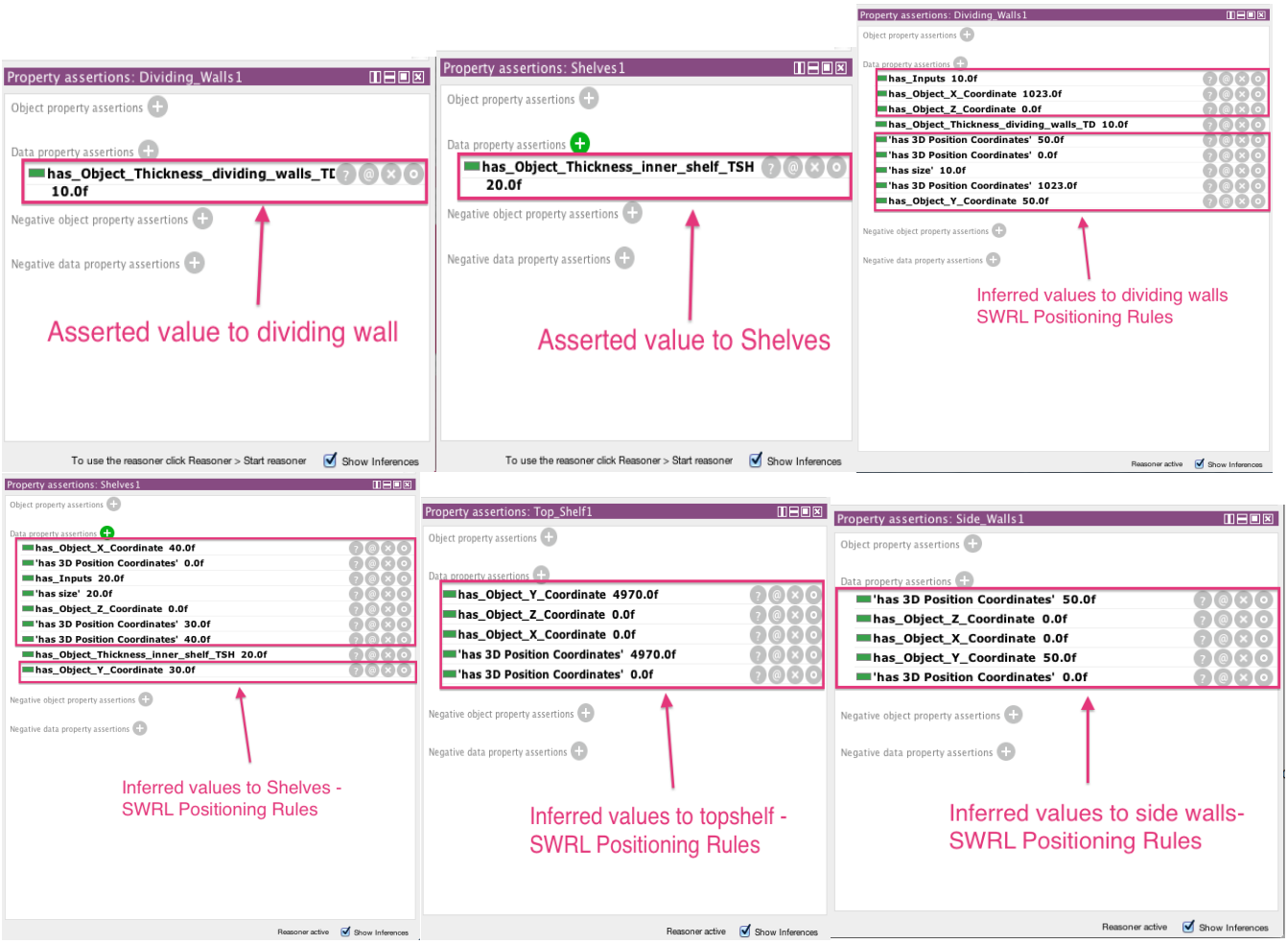


Figure 7-20: Asserted and Inferred value to Bookshelf Sub-assembly: SWRL Rules

To illustrate changes in the inferred values as per changes in asserted value of the bookshelf and its subassembly attributes, as per the SWRL engineering rules, a few dimensions are altered as shown in Figure 7-21. The changes in asserted values are illustrated with the help of Figure 7-22.

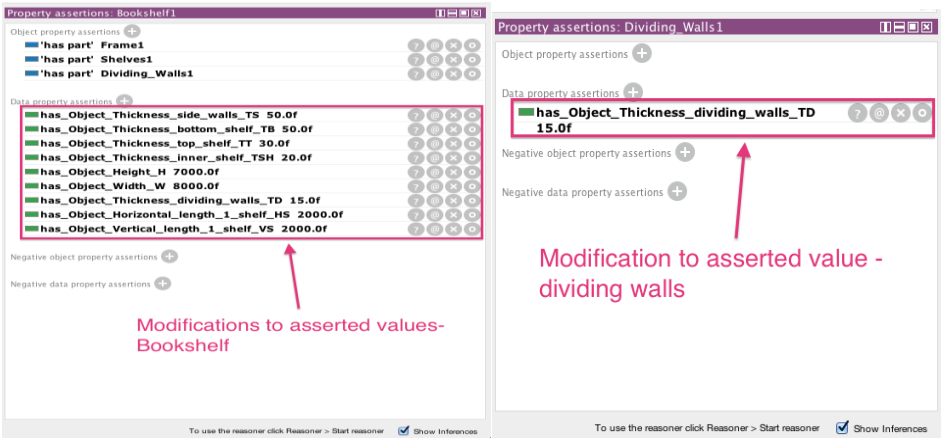


Figure 7-21: Modifications in asserted values – Bookshelf and subassembly attributes

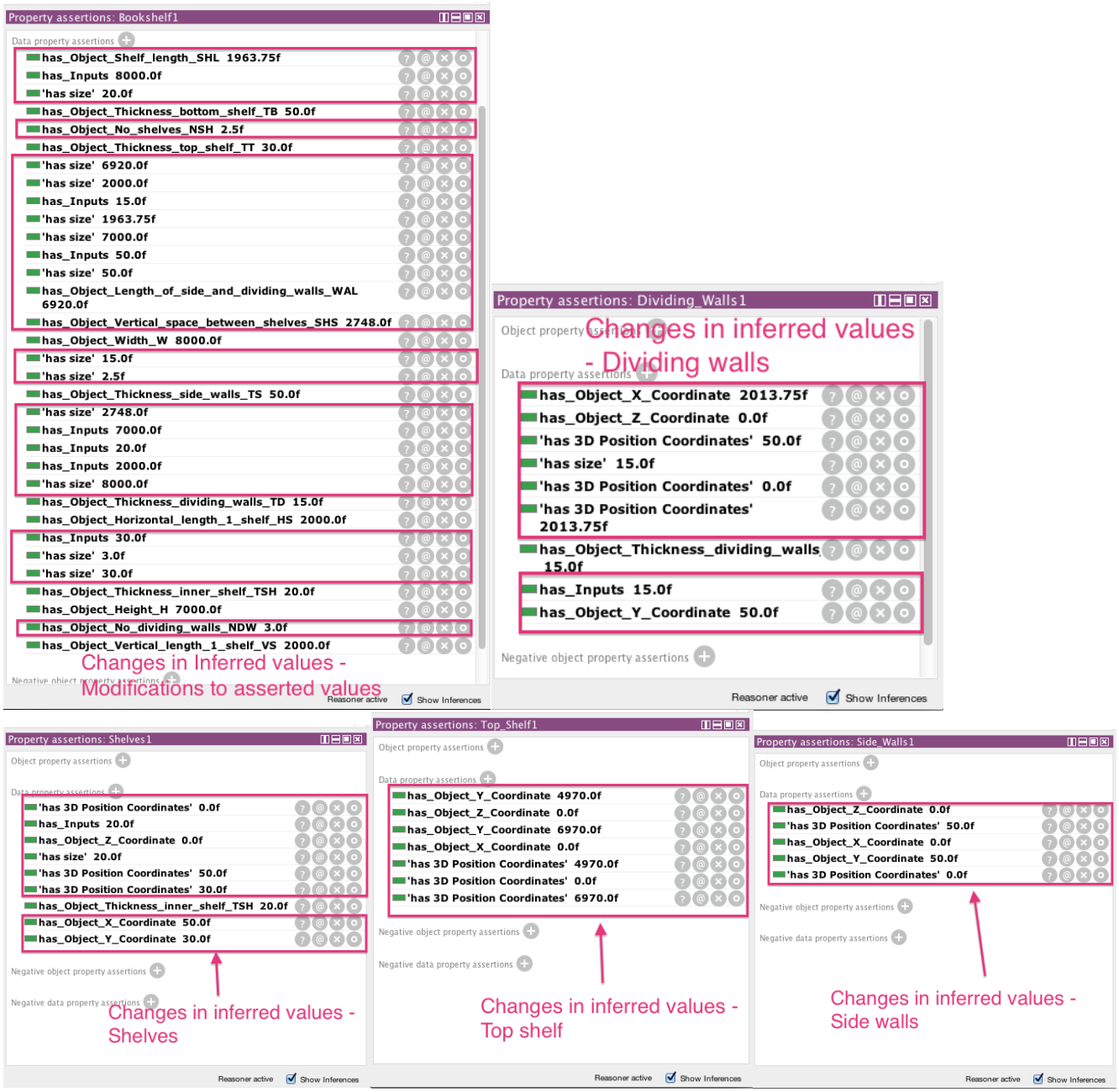


Figure 7-22: Changes in Inferred Values: Bookshelf and Subassembly attributes

As, it can be observed by comparison of Figures 7-19, 7-20, 7-21 and 7-22, in spite of the fact that the value of *Thickness of inner shelf (TSH)* is kept at the same value of 20.0 mm, the position co-ordinates of the shelves in the virtual space still change as inferred values as they are dependent upon other attributes such as *Thickness of side walls (TS)* and *Thickness of bottom shelf (TB)* along with *TSH as per the Shelves Position SWRL rule*. All the other attributes such as *No. of dividing walls (NDW), No. of shelves (NSH), Vertical*

*spacebetween*Shelves (SHS), *Shelf Length (SHL) and Length of Side and Dividing walls (WAL)* are altered as per their corresponding SWRL rules such as *Dividing walls rule, Shelves, Shelf Length, Side and Dividing Walls along with Vertical Space Shelves rule*.

Similarly, the position coordinates of the Dividing walls, Shelves, Top shelf and Side walls are also altered as per the SWRL positioning rules such as Dividing walls Position rule, Topshelf position and Side Walls position rule.

7.6.3 Experiment 3 – SQWRL Query with Violation in Asserted Values

A few violations are asserted as per the Dividing walls rule and Shelves rule to calculate the No. of Dividing walls and Shelves. As per the semantics and the SWRL representation, violations to bookshelf attributes are illustrated with the help of Figure 7-23.

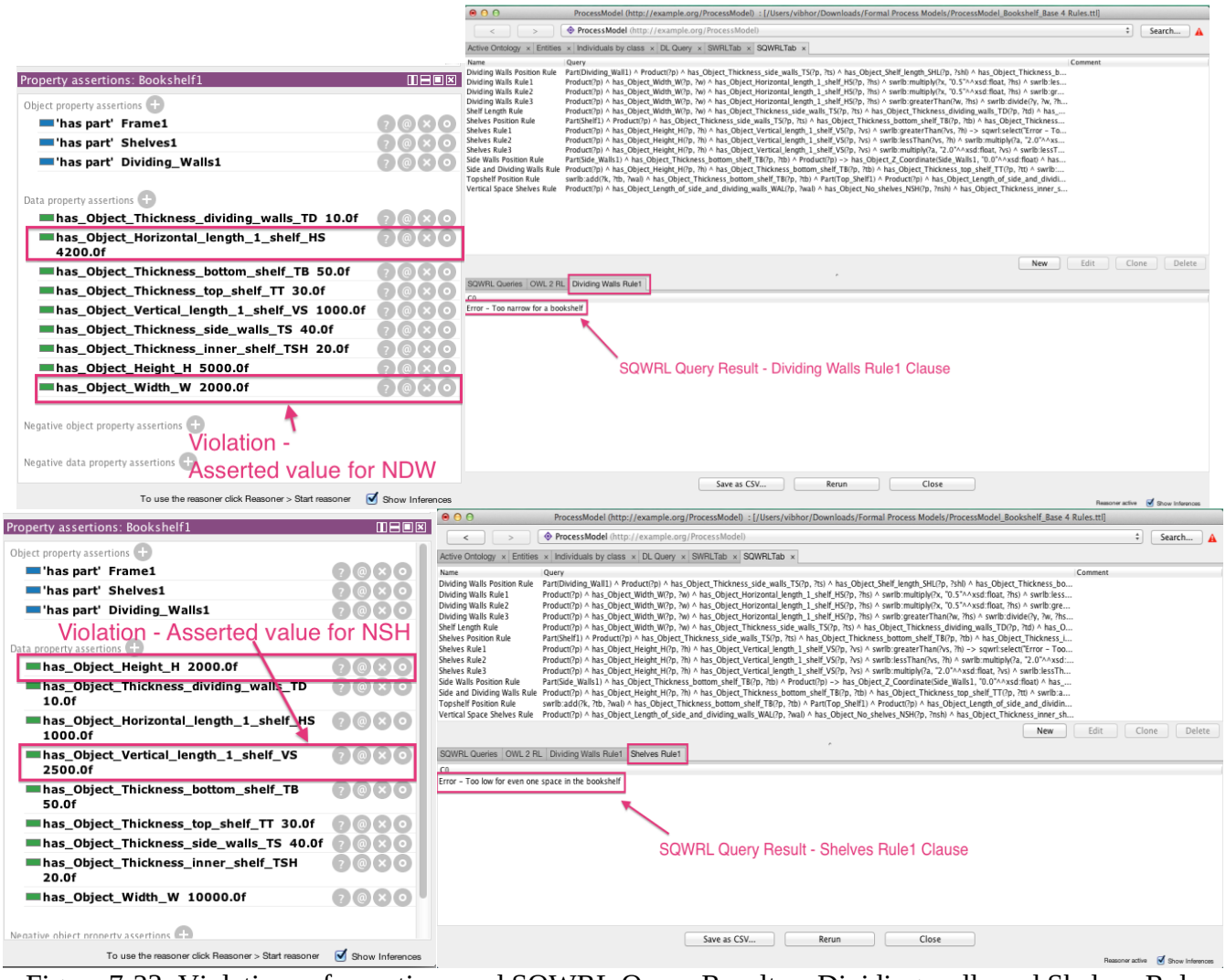


Figure 7-23: Violations of assertions and SQWRL Query Results – Dividing walls and Shelves Rule

As per the semantics of Dividing walls rule, *If (W<HS*0.5)*, then its an error as no dividing wall can be created. Thus as per assertion in figure 7-23, W=2000.0 mm, HS=4200.0 mm and hence the initial clause is correct. The SQWRL query returns the result '*ERROR- Too narrow for a bookshelf*' as illustrated. Similarly, as per the *Shelves rule, If (VS>H)*, then no shelf can be created. Thus as per asserted values in Figure 7-23, VS = 2500.0 mm, H=2000.0 mm and thus the SQWRL query returns '*ERROR – Too low for even one space in the bookshelf*'.

Same values to W=8000.0 mm, HS=2000.0 mm, VS=2000.0 mm and H=7000.0 mm are asserted to bookshelf in Figure 7-24 and 7-21. As observed from Figure 7-22, NDW is inferred at value 3.0, the query result of Dividing walls rule1 clause *should not return 'ERROR'.*

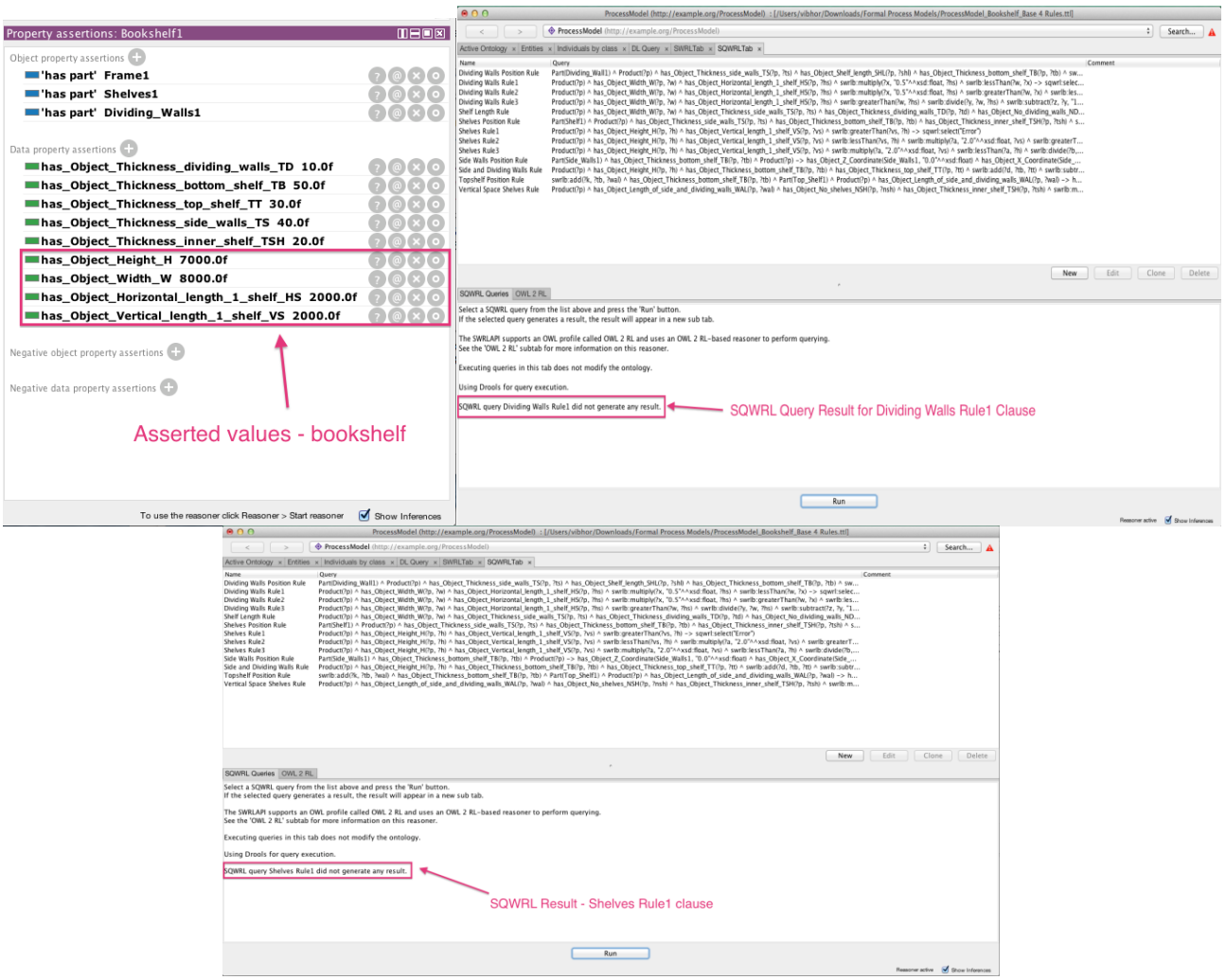


Figure 7-24: Modifications to Asserted Values and Change in SQWRL Query Results – Dividing walls and Shelves Rule

Similarly, as observed from Figure 7-22, NSH is inferred at 2.5, the query result of Shelves rule1 clause *should not return 'ERROR'*. As can be observed from Figure 7-22, the SQWRL result are not generated which is in line with the semantic clarity of the represented SWRL syntax of the represented rules.

7.6.4 Experiment 4 - Comparison of SWRL and SQWRL Rule Outputs to Platform Specific DEA Systems

This section has elaborated upon the comparison of testing to attributes of bookshelf in GPM-DEA with OWL/SWRL and its implementation inside proprietary DEA systems such as AML, Siemens NX Knowledge Fusion (KF) and CATIA Knowledgeware as part of this thesis. Although the bookshelf has been implemented in all three DEA systems, the method of implementation varies as AML is a true KBE system and enables generative modelling through functional requirements but GA based CAD systems such as Siemens NX KF and CATIA knowledgeware enable parametric modelling but don't enable generative modelling. Thus the knowledge analysis is performed after the geometric design stage in Siemens NX KF and CATIA knowledgeware whereas the knowledge analysis is done prior to the geometric design stage in DEA through a KBE approach, which is the adopted method in this research.

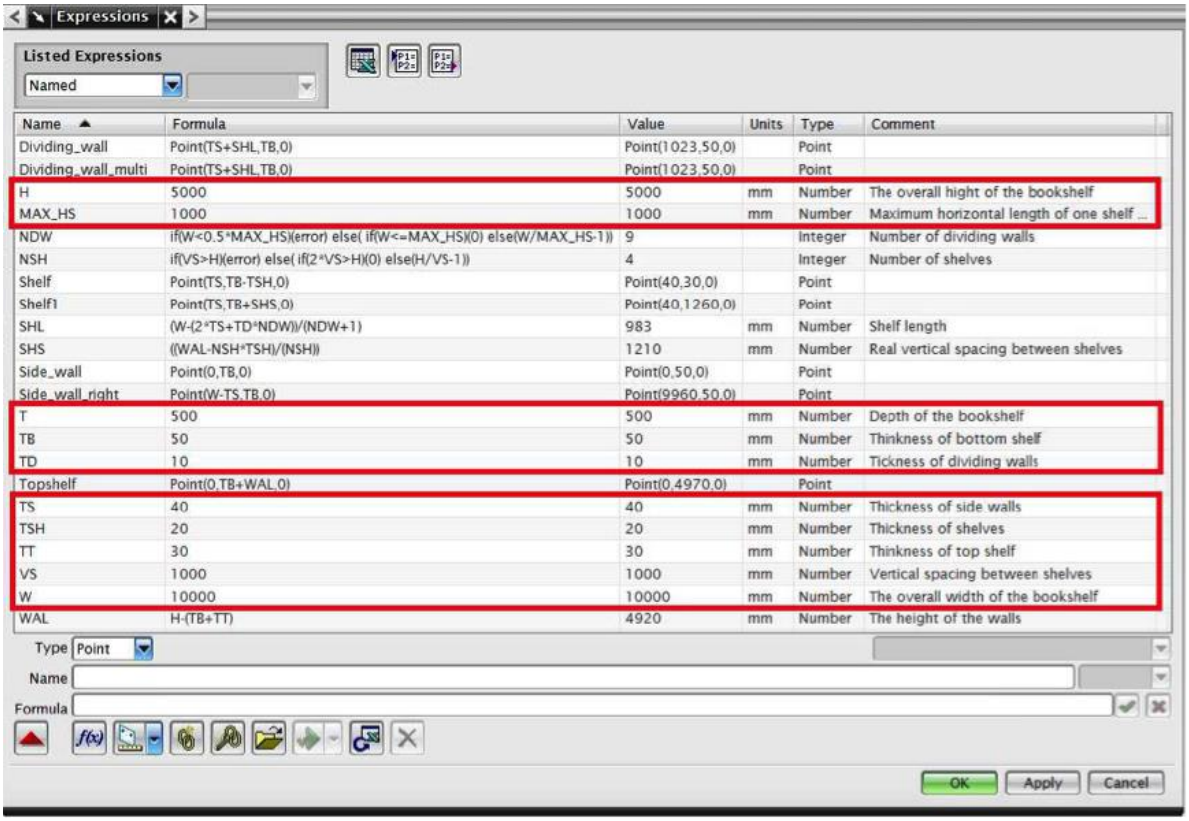
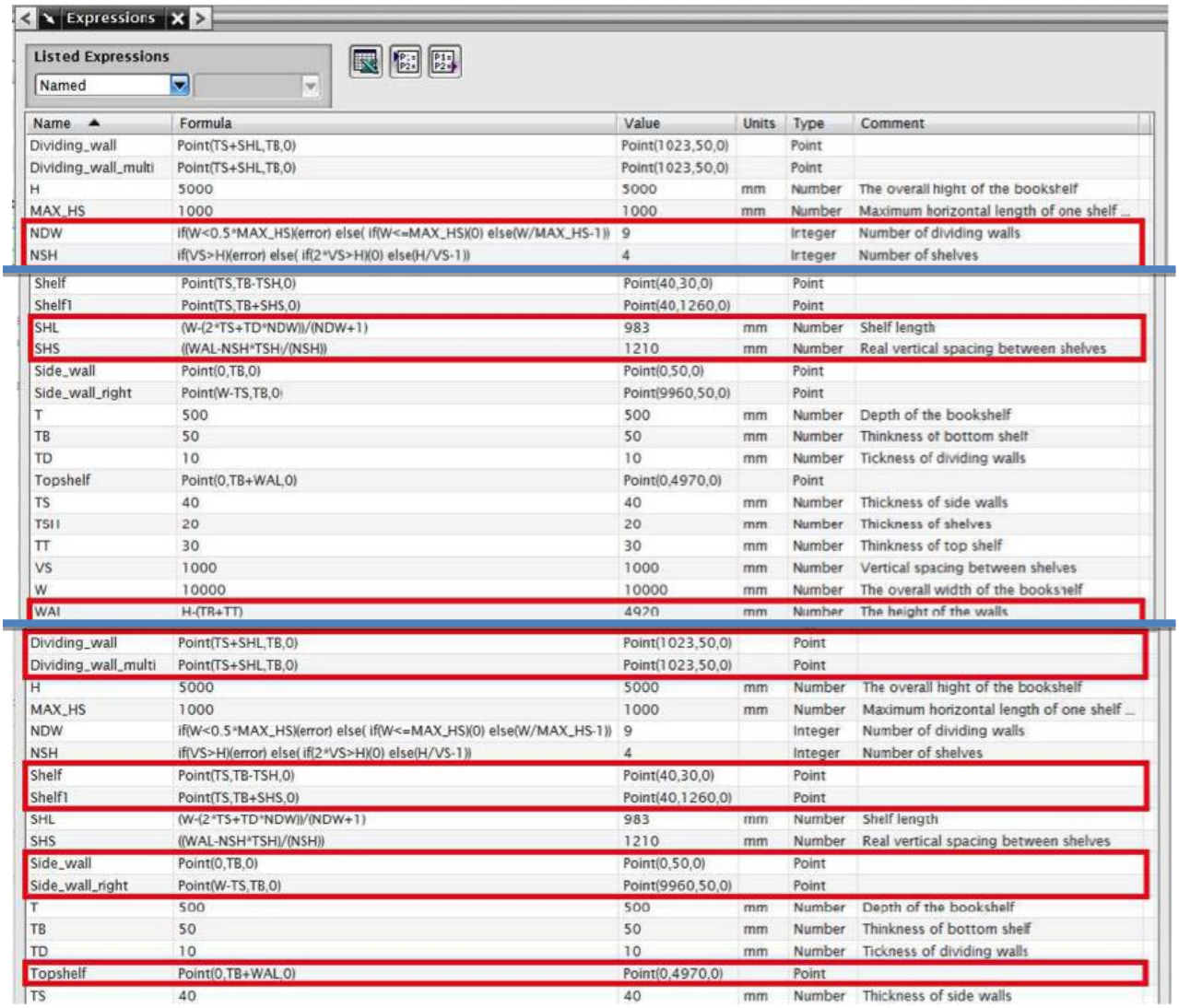


Figure 7-25: Input values to bookshelf attributes – Siemens NX Expression Window (Lützenberger et al., 2012, Pg 39)

For experimental verification of the implementation, the same values are instantiated to bookshelf attributes in the developed ontology as shown in Figure 7-19 and 7-20 as compared to the implementation in Siemens NX expression window in Figure 7-25. The output values of attributes based on rules inside the expression windows are illustrated with Figure 7-26.



Name	Formula	Value	Units	Type	Comment
Dividing_wall	Point(TS+SHL,TB,0)	Point(1023,50,0)		Point	
Dividing_wall_multi	Point(TS+SHL,TB,0)	Point(1023,50,0)		Point	
H	5000	5000	mm	Number	The overall hight of the bookshelf
MAX_HS	1000	1000	mm	Number	Maximum horizontal length of one shelf ...
NDW	if(W<0.5*MAX_HS)(error) else(if(W<=MAX_HS)(0) else(W/MAX_HS-1))	9		Integer	Number of dividing walls
NSH	if(VS>H)(error) else(if(2*VS>H)(0) else(H/VS-1))	4		Integer	Number of shelves
Shelf	Point(TS,TB-TSH,0)	Point(40,30,0)		Point	
Shelf1	Point(TS,TB+SHS,0)	Point(40,1260,0)		Point	
SHL	(W-(2*TS+TD*NDW))/(NDW+1)	983	mm	Number	Shelf length
SHS	((WAL-NSH*TSH)/(NSH))	1210	mm	Number	Real vertical spacing between shelves
Side_wall	Point(0,TB,0)	Point(0,50,0)		Point	
Side_wall_right	Point(W-TS,TB,0)	Point(9960,50,0)		Point	
T	500	500	mm	Number	Depth of the bookshelf
TB	50	50	mm	Number	Thinkness of bottom shelf
TD	10	10	mm	Number	Tickness of dividing walls
Topshelf	Point(0,TB+WAL,0)	Point(0,4970,0)		Point	
TS	40	40	mm	Number	Thickness of side walls
TSH	20	20	mm	Number	Thickness of shelves
TT	30	30	mm	Number	Thinkness of top shelf
VS	1000	1000	mm	Number	Vertical spacing between shelves
W	10000	10000	mm	Number	The overall width of the bookshelf
WAL	H-(TB+TT)	4920	mm	Number	The height of the walls
Dividing_wall	Point(TS+SHL,TB,0)	Point(1023,50,0)		Point	
Dividing_wall_multi	Point(TS+SHL,TB,0)	Point(1023,50,0)		Point	
H	5000	5000	mm	Number	The overall hight of the bookshelf
MAX_HS	1000	1000	mm	Number	Maximum horizontal length of one shelf ...
NDW	if(W<0.5*MAX_HS)(error) else(if(W<=MAX_HS)(0) else(W/MAX_HS-1))	9		Integer	Number of dividing walls
NSH	if(VS>H)(error) else(if(2*VS>H)(0) else(H/VS-1))	4		Integer	Number of shelves
Shelf	Point(TS,TB-TSH,0)	Point(40,30,0)		Point	
Shelf1	Point(TS,TB+SHS,0)	Point(40,1260,0)		Point	
SHL	(W-(2*TS+TD*NDW))/(NDW+1)	983	mm	Number	Shelf length
SHS	((WAL-NSH*TSH)/(NSH))	1210	mm	Number	Real vertical spacing between shelves
Side_wall	Point(0,TB,0)	Point(0,50,0)		Point	
Side_wall_right	Point(W-TS,TB,0)	Point(9960,50,0)		Point	
T	500	500	mm	Number	Depth of the bookshelf
TB	50	50	mm	Number	Thinkness of bottom shelf
TD	10	10	mm	Number	Tickness of dividing walls
Topshelf	Point(0,TB+WAL,0)	Point(0,4970,0)		Point	
TS	40	40	mm	Number	Thickness of side walls

Figure 7-26: Output values to bookshelf attributes – Siemens NX Expression Window (Lützenberger et al., 2012, Pg 40, 41, 43)

On comparison of the inferred values for the bookshelf design in OWL/SWRL in Figure 7-19 and 7-20 to the attributes inside Siemens NX Expression Window in Figure 7-26, it can be observed that the values are exactly the same such as *NDW=9, NSH=4, SHL=983 mm, WAL=4920 mm, Topshelf position coordinates as (0,4970,0) and Dividing Walls position coordinates as (1023, 50, 0).*

An anomaly is also compared in both OWL/SWRL and AML as a violation of assertion. Figure 7-27 illustrates the specified incorrect value to asserted parameters – *Bookshelf height (H) as 2.5 m and Vertical spacing between shelves (VS) as 2.6 m inside AML.*

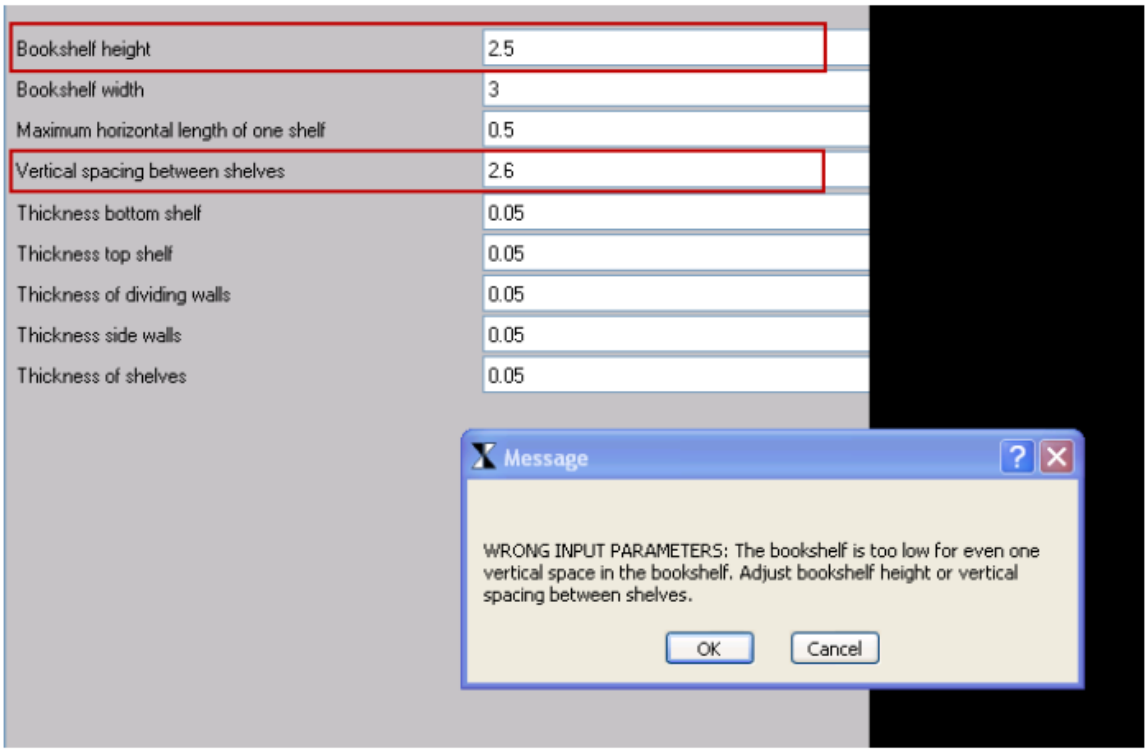


Figure 7-27: Incorrect value to H and VS parameters inside AML – Bookshelf Design Process (Lützenberger et al., 2012, Pg 73)

It is important to note the difference in units in AML, which is in Meters (m) and Siemens NX Expression window and OWL/SWRL (Protégé) in Millimeters (mm). Upon assertion of the same set of values to H and VS and all the other bookshelf attributes in OWL/SWRL model in this research as shown in Figure 7-28, the query result of the shelves rule1 clause shows "*Error – Too low for even one space in the bookshelf*" which offers the same result as the output message inside AML in Figure 7-27.

The AML code for the rules for the bookshelf design is shown in Appendix.

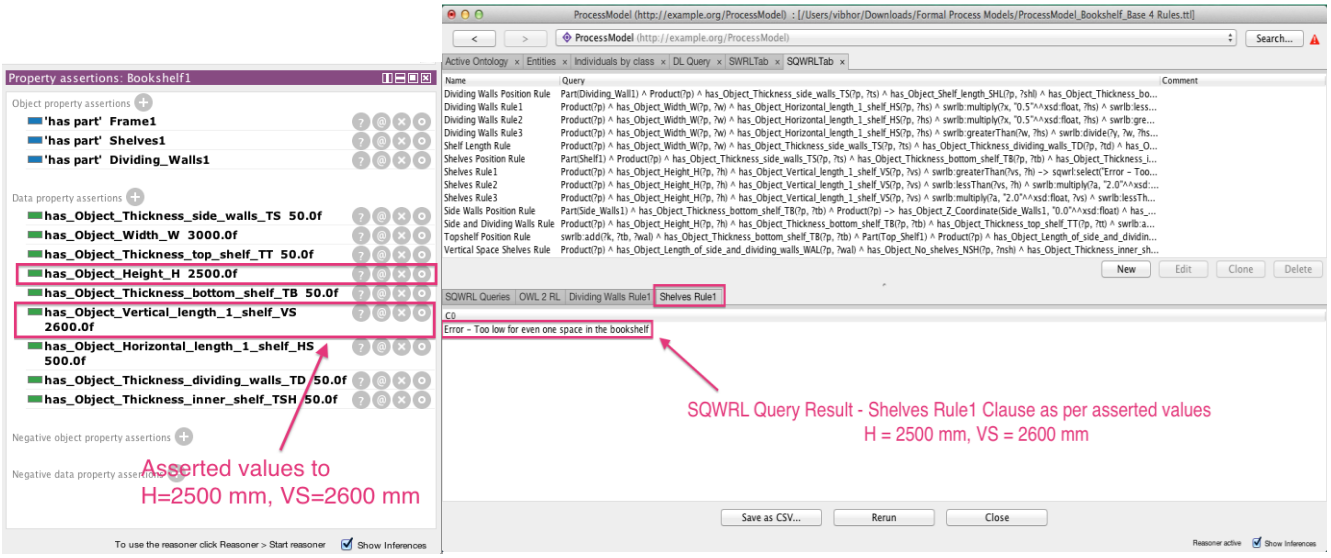


Figure 7-28: Incorrect value to H and VS parameters in OWL/SWRL – Bookshelf Design Process

7.7 Discussion of the experimentation results

The results of the Use Case 3 & 4 experiment prove that the inference and query mechanism in OWL/SWRL for GPM-DEA enables DEA in a virtual environment with both design and manufacturing knowledge by providing accurate results with transparency of knowledge. It is important to state platform specific DEA systems such as ParaPy, Siemens NX and AML have an inbuilt GUI to show the effect of assertions and violations directly on the product's visual form through an inbuilt geometry modeller but without any semantic clarity, which is open to interpretation by engineers. The OWL/SWRL representation formulated in this research doesn't provide a GUI interface through incorporation of an inbuilt product geometry modeller to show the effect of GPM-DEA on the product's visual form in the present stage. However, the inference and query results with variation in assertions and violations are created as text and numerical values to show the effect of the process model on product's attributes and provide much more semantic clarity as compared to ParaPy, AML as a DEA system. Thus the GPM-DEA schema, developed by this research, provides a method to use ontologies with rule representation in context to achieving DEA with a KBE approach

with semantic clarity, transparency, traceability and re-usability of developed Meta model in this research. ***GPM-DEA provides a robust, structured and coherent method to build knowledge model with usage of formal OWL/SWRL ontologies as knowledge representation (KR) in context to achieving KBE based DEA***. The ontology and rule based OWL/SWRL representation adopted by the author successfully represents the equivalent platform independent and neutral formal representation.

7.8 Summary

This chapter has provided ***experimental verification of various research aspects of this thesis with the testing of the functionality of GPM-DEA implemented in OWL/SWRL ontology and rule representation as formal logic based neutral representation***. Thus it has been proven that the GPM-DEA in its informal /semiformal representation, through OWL/SWRL as platform independent and neutral formal representation enables DEA with generative modelling catering to multiple mechanical design with DFM/DFA cases and provides accurate results similar to a platform specific DEA system. The experimentation with both Use Case scenarios provides proof of generic working of GPM-DEA with both re-usable and product specific knowledge. An important point of consideration is the fact that the first step of representing the informal / semiformal knowledge is manually represented in OWL/SWRL as platform independent and neutral formal representation with accurate semantics. The inference (automated reasoning) and the query mechanism on the formally represented OWL/SWRL knowledge returns accurate results with varied generic and product specific concepts and relations of the process model with semantic clarity for DEA with both design and manufacturing viewpoints during the design stage. The inference and query results are shown as text and numerical values to product's attributes as compared to the product's geometric form with GUI inside a proprietary DEA system.

8 Conclusion

8.1 Introduction

The research work discussed in this thesis has developed a Generative Process Model for Design Engineering Automation (GPM-DEA) with neutral formal semantics utilising OWL/SWRL ontology and rule representation formalism for DEA with a KBE approach. GPM-DEA built on the author's Meta model provides a model driven approach utilising strengths of existing modelling standards such as UML/SysML and IDEFO for building structured knowledge models of mechanical design process with DFM knowledge for human access and aid as an informal/semiformal representation. It provides a method to use formal OWL/SWRL ontologies through its schema for the use of DEA with a KBE perspective with generative modelling based on generic SWRL functions developed by the author for queries and reasoning. With experimental system development and verification through 2 test use-cases, it has been demonstrated that the corresponding platform independent and neutral formal representation of GPM-DEA, for machine interpretation, using OWL/SWRL enables DEA for mechanical product design process with DFM/DFA with preserved semantics within a virtual engineering environment and with generative modelling capabilities using the SWRL functions developed in this research as explained in section 5.6.3. This chapter compiles the discussion, provides conclusion from the results and suggests some future work based on the research work completed in this thesis.

8.2 Summary of Thesis and Discussion

The current research has introduced a novel Generative Process Model for Design Engineering Automation (GPM-DEA) with exploration of formal representation with machine interpretation. The schema of the process knowledge model provides a method to use formal logic based ontology representation to achieve DEA with a KBE perspective with

generative modelling. Various concepts and relations of mechanical design process with manufacturing knowledge based on the authors Meta Model have been informally captured in GPM-DEA as a process model and then formally represented in OWL/SWRL as platform independent and neutral formal representation with preserved semantics to address the needs of DEA with a KBE approach. The research work satisfies the aim and objectives stated in section 1.3, which helped raise a few research questions in section 4.3. This research is based on the shortcomings of KBE methodologies such as MOKA being a comprehensive one, others such as KNOMAD and CommonKADS, in order to target the needs of DEA.

'*This aim of this research is to provide a coherent method to develop platform independent and neutral formal representation of an engineering process model, with focus on mechanical product design process with manufacturing knowledge, and semantic clarity for DEA. This coherent method will capture various knowledge entities and relationships such as activity, product attributes, rule, function and behaviour as Meta Model, identified with literature analysis in an informal process model (for human aid and interpretation). The 2nd step will provide a method to represent the schema of the structured process model in neutral formal representation (for machine/system interpretation) with open standards for DEA with KBE as a holistic approach. This will include generative modelling capability by building queries as per a set of generic predefined functions. It will perform DEA with effect of the process model on product attributes with the help of inference (automated reasoning) and querying'*

Post MOKA, Systems engineering approach such as Model Based Systems Engineering (MBSE) based UML/SysML have been used by academics and researchers to capture knowledge with a model driven approach along with formal logic based ontology languages to formally represent engineering design knowledge for machine interpretation with neutral semantics. Chapter 2 described DEA with various perspectives such as CAx

(CAD/CAE/CAM), PDM/PLM and KBE where it was identified that KBE as a design method provides a more holistic automation enabling generative modelling and with a process oriented approach. This provided accomplishment of research objective 1 –

1. To investigate different approaches for Design Engineering Automation (DEA) including CAx, PLM and KBE for product and process based automation

Under the KBE umbrella with a focus on platform independent and neutral knowledge models for design automation, crucial work has been performed by (Sanya and Shehab, 2015, 2014) for usage of OWL/SWRL ontologies, utilisation of OWL/RIF/MathML based ontology representation by (Reijnders, 2012) and RIF for product design engineering rules by (Colombo et al., 2014; Lützenberger et al., 2012). An application was also developed in the form of Design and Engineering Engine (DEE) by (Curran et al., 2010). However, some of the shortcomings that were identified were a structured knowledge modelling method for engineering design with focus on mechanical design and DFM/DFA by developing a process model whose schema can be utilised to effectively use formal ontologies such as OWL based languages to address the needs of DEA in a standardised way. The platform independent and neutral model developed should provide re-usability, transparency, traceability of concepts and relationships based on Meta Model analysis and provide generative modelling. The knowledge should include both geometric and non-geometric knowledge with Function-Behaviour-Structural (F-B-S) aspects such that the developed system can enable rule based modelling and geometry automation (GA) along with wider design space exploration with functional requirements with reasoning and query mechanism on the formal axioms thus targeting DEA for mechanical product design process with DFM/DFA aspects.

The compliance of the outcomes of this research work as per the set objectives, identified research gap along with critical analysis of the developed process model with the ontology system development and its experimental verification is presented in this section.

8.2.1 Development and Formulation of GPM-DEA model

Research Question 1 in section 4.3 is stated as -

I. 'How can the mechanical product design process with inclusion of manufacturing knowledge (DFM/DFA) based on various entities such as *activities, rules, logic, function and behaviour* for product realisation as per author's Meta model, be captured in *a generic and re-usable process model as a model driven approach with structured knowledge model for automation* in a virtual engineering environment?'

The answer to this question caters to research objectives 2 and 4 in section 1.3 which are stated again as –

2. To analyse and compare various informal and semiformal process modelling methods to capture various aspects of an engineering design process with focus on mechanical product design with design for manufacturing knowledge for automation
4. To develop and build a detailed informal/semiformal process model with explicit relationships between identified knowledge entities of a mechanical product design process with design for manufacturing knowledge.

After careful assessment of existing literature for addressing the needs of DEA with KBE as a holistic approach, requirements were formulated for informal/semiformal modelling methods for knowledge modelling of various mechanical design process with manufacturing knowledge concepts such as activities with inputs, outputs, engineering rules, resources, function, behaviour and its effect on the product in section 3.2. Comparative analysis of informal/semiformal modelling methods was performed against the formulated requirements in section 3.5. ***The results in section 3.8 indicated that, individual modelling methods are able to informally capture certain aspects for mechanical design knowledge with manufacturing aspects such as IDEF0 for process knowledge with inputs, outputs, links to rules as controls and resources and UML and SysML for product knowledge with***

attributes and methods. However, none of the modelling methods are able to capture all aspects in a unified process model, with its effect on product attributes. This includes function, behaviour and structure (F-B-S) in context to the process model.

The findings of careful literature analysis in chapter 3 for research question 1, demonstrate that a hybrid approach needs to be adopted for knowledge modelling of a complete mechanical design process knowledge covering manufacturing aspects. ***GPM-DEA is developed by this research which can informally capture all the aspects of mechanical design process with inclusion of manufacturing knowledge as DFM/DFA based on the authors Meta model utilising a hybrid approach of existing modelling standards along with addition of new knowledge objects***. ***It achieves this by integration of existing modelling methods such as IDEF0-based function modelling of activities, UML class diagram, UML condition link, SysML requirement diagram and the addition of constructs on top of this to demonstrate behaviour such as bidirectional arrows as properties between IDEF0 activities, SysML requirement diagram and UML class diagram***. The activities include inputs and outputs in terms of product geometric attributes as parameters with float values, engineering rules based on both text and math along with resources. The engineering rules vary from purely process rules to an integrated product specific and process knowledge. ***Process rules are represented with UML condition links to control the sequence of activities***. ***Engineering rules controlling the topology of the product are represented using UML class diagram methods***.

Thus, GPM-DEA provides a model-driven approach for knowledge modelling of mechanical design processes for DEA. The breakdown of the design process functional requirements into sub-functions for various stages of the design process along with objects has been explained in section 5.3.1 and 5.3.2 along with the integration of the process model with its interface to the detailed product model in UML class diagram. ***The complete functioning of GPM-DEA***

with generative modelling capability for DEA with KBE approach has been explained with Figure 5-4 in section 5.4 and section 5.5, which satisfies research objectives 2 and 4 and provides the answer to research question 1. Along with literature analysis, the development of GPM-DEA has been completed in compliance with the results of the comparative analysis in section 4.8 and 4.9, and in-line with the research methodology in section 1.4.2.

8.2.2 Neutral formal representation of GPM-DEA in OWL/SWRL ontology and rule representation

Research question 2 in section 4.3 is stated as -

II. 'How can the developed process model in line with author's Meta model be then *formally represented for machine interpretation in platform independent and neutral representation standards with semantic clarity* (clear meaning of concepts) for Design Engineering Automation (DEA) for mechanical design with DFM/DFA with a KBE approach through open standards?'

The answer to this question satisfies the needs of research objectives 3 and 5 in section 1.3 which have been stated as

3. To analyse and compare state of the art in existing formal representation (machine readable) techniques and standards.
5. To formalise the process model in platform independent and neutral formal representation standards for DEA with semantic clarity. This will incorporate generative modelling capability by generating the activities, objects of the process and rules based on logic as per set of developed generic functions.

GPM-DEA is built as a process model for knowledge modelling of mechanical design process with DFM knowledge for DEA with MOKA as the basis for knowledge modelling and formalisation. Section 2.5.3 discussed various KBE methodologies such as

KOMPRESSA, KCM, CommonKADS, MOKA and KNOMAD. MOKA methodology is one of the most comprehensive and is focussed on the product design process. Section 2.5.3 showed that the MOKA formal knowledge model in the form of MML wasn't successful in achieving DEA with its formal representation. It was verified that UML/SysML based notation as an MBSE language lacks formal semantics and is suitable as semiformal or lightweight formal representation for visual display (Chungoora et al., 2013a, 2013b; Graves, 2009). Thus in order to represent all formulated concepts and relations of GPM-DEA with neutral formal semantics, knowledge representation (KR) languages such as PSL, OWL, RuleML and RIF were considered. Requirements for a generic and re-usable process model for DEA with neutral formal representation with semantic clarity have been compiled in section 4.7. The comparative analysis of formal representation standards against the compiled requirements has been performed in section 4.8 and 4.9.

The results indicate that all the concepts and one-to-many relations of GPM-DEA as described in section 5.2 in chapter 5, cannot be semantically mapped to a single existing neutral formal representation language such as OWL, PSL, RuleML, RIF and MathML. Thus, as discussed in section 4.9.1, PSL comes across as a very capable ontology for neutral formal process descriptions for manufacturing and production operations. Although OWL is less expressive than PSL, it provides a neutral platform to formally represent concepts and binary relations of GPM-DEA for mechanical design processes with both design and manufacturing knowledge. Rule language is required to formally represent the rules represented in UML class diagram with its interdependency on IDEF0 rules to activities such as RuleML, RIF and MathML. Thus integration of ontology with rule language is mandatory to fully represent the GPM-DEA with its F-B-S on neutral formal representation. The final results concluded that OWL/SWRL as a combination of both ontology and rule language is a suitable candidate for the semantic mapping of GPM-DEA concepts and relations.

According to the research methodology in section 1.4.2, the ontology development methodology (Noy and McGuinness, 2001) for GPM-DEA needs to be experimentally verified to show the effectiveness of its working. In spite of the fact that process model aspects as part of pilot use-cases have been experimented with PSL syntax in section 4.4, due to the lack of availability of tools for experimental verification of formal axioms with PSL along with its limitation to represent knowledge for design systems, OWL with its ease of integration with Datalog dialect of RuleML as OWL/SWRL within Protégé IDE (Horridge et al., 2011) as the editing tool was finalised. The GPM-DEA is saved as an XML file using DrawIo tool before being manually mapped to OWL/SWRL ontology.

8.2.3 Functioning of OWL/SWRL system

Research question 3 in section 4.3 is stated as –

III. 'Can the *formalised process model* enable automation with *generative modelling* from the *functional requirements* generated at the initiation of the design process as the design intent *with queries and reasoning on developed generic functions*?'

The answer to this question satisfies the needs of research objectives 5 and 6 in section 1.3 which are stated again as –

5. To formalise the process model in platform independent and neutral formal representation standards for DEA with semantic clarity. This will incorporate generative modelling capability by generating the activities, objects of the process and rules based on logic as per set of developed generic functions.
6. To perform experiments in order to validate and verify the process based knowledge model with its platform independent and neutral formal representation for re-usability, transparency and accuracy.

The OWL/SWRL representation for GPM-DEA is illustrated in section 5.5. ***The generative modelling capability of GPM-DEA has been added as a very crucial part of this research***

with the help of SWRL functions on top of OWL ontology and has been demonstrated in section 5.6.3 and validated with experimental verification of test use-cases in chapter 7. It is based on function structures of design process with activities and objects functional requirements as illustrated in section 5.5.2. All the knowledge objects such as activity, rules, resources, function have been created as classes within OWL ontology whereas inputs and outputs for activities have been created as datatype properties as binary relations between classes and float values. This is explained in section 5.5.1. The engineering rules as methods in the UML class diagram are also represented using SWRL formalism. All the class types, properties and restrictions for the OWL/SWRL are illustrated in section 5.5.

The application of the complete OWL/SWRL model for GPM-DEA as Knowledge Representation (KR) system development has been elaborated in detail in chapter 6 using test use-cases as Use Case 3 and 4. Use-case 3 is an instance of drilling as a design process in a block as a product. The initial task is to break down the function structures of various activities such as drilling, reaming, boring which all can achieve the desired functional requirement of creating a hole along with the assessment of the initial product as block. This has been discussed in section 6.3.1. An instance has been visually represented using GPM-DEA concepts and relations with the Figure 6-6, 6-7 and 6-8 as informal/semiformal representation in section 6.3.2. The corresponding OWL model with classes, properties, restrictions, SWRL rules and the SWRL generative modelling functions for the instance of the drilling process in the plug-in have been explained in section 6.3.3.

The wider applicability and re-usability of this work is proven with the experimentation with another test use-case (Use Case 4), which includes designing a bookshelf. The application of GPM-DEA and its neutral formal representation in OWL/SWRL follows a similar approach to the instance of drilling and is described in section 6.4. The initial step is breaking down of the function structures for bookshelf design process activities and objects along with

assessment of the initial product as described in section 6.4.1. The informal/semiformal representation is illustrated in section 6.4.2 with the OWL/SWRL as neutral formal equivalent representation in section 6.4.3. This includes all the classes, properties, restrictions, SWRL rules and the SWRL generative modelling functions based on existing classes and properties in the OWL model.

8.2.4 Reasoning and querying on OWL/SWRL model

Research question 2 and 3 also provide answer to the research objective 6 in section 1.3. The OWL/SWRL provides a *platform independent and neutral representation* to the coherent model driven GPM-DEA thus providing DEA for mechanical design process and DFM with generative modelling based on authors set of generic SWRL functions. The OWL/SWRL model has been populated with test use-cases to demonstrate generic working, re-usability and traceability of Meta model concepts along with the effect of the process model from functional requirements analysis to inclusion of product parameters. The rule outputs from both these use-cases have been validated inside proprietary platform specific DEA systems such as KBE based AML, ParaPy and GA based parametric CAD based Siemens NX KF and CATIA Knowledgeware. The reasoning and querying on the OWL/SWRL knowledge model has been performed with the rule outputs being compared with corresponding implementation inside DEA systems to test the accuracy of reasoning and querying with semantic clarity.

Various experiments were designed as described in section 7.3 to experimentally test and verify the reasoning capability of OWL/SWRL for both test use-cases. Section 7.7 discusses the results, which indicate that the ***generative modelling functions generate appropriate results for activities and objects based on functional requirements*** **along with** ***rule generation based on logic*** **and initial assessment of product**. The input of the model is a product in initial state and the output of the model is a product in final state where state is indicated by product attributes. The SWRL rules also provide accuracy in float value with

variation in both datatype and object properties based on engineering rules. The SQWRL query also returns appropriate results with preserved semantics based on variation in values and violation of asserted rule axioms. The SQWRL results are text based, which provide semantic clarity. However, they can incorporate float and integer values as well in the query tab. The comparison of both SWRL reasoning and SQWRL query results as rule outputs match to the values of the rule outputs for product configuration inside ParaPy, AML as a KBE based DEA system and Siemens NX KF and CATIA Knowledgeware as GA for parametric modelling based which proves that the inference with Pellet and Drools reasoner is accurate for the equivalent OWL/SWRL model of GPM-DEA schema.

However, a limitation of the OWL/SWRL model in this research is the generation of text and numerical based results with reasoning and querying which provide semantic clarity but are unable to show the exact effect on product attributes with the help of GUI to show the results on product's visual form as a product model with an inbuilt geometry modeller.

8.3 Applicability and Effectiveness of the Research Outputs

GPM-DEA was developed with generic and re-usable engineering concepts such as activity, product attributes, rule, function-functional requirements, behaviour based on authors Meta model for knowledge modelling with a model driven approach (MDA). The MDA approach led to the development of GPM-DEA with functional modelling as the basis, as the purpose of the engineering mechanical design process is to satisfy a set of functional requirements in context to a product (Chen et al., 2008). After experimental verification of the OWL/SWRL as system development based on GPM-DEA schema, the concepts and relations of GPM-DEA have been proven effective for generic and product specific design processes with concepts and relations such as activity with inputs and outputs, engineering rules comprising of both design and manufacturing constraints, function and product architecture covering a wide array of cases. GPM-DEA contains both declarative and procedural design process

knowledge with more focus on declarative knowledge to satisfy the needs of DEA with a KBE approach as against purely procedural approach for DEA with other virtual engineering approaches such as CAx tools and PDM/PLM systems (Cooper and LaRocca, 2007; Prasad, 2006). The use of OWL ontology and SWRL rules as a platform independent and neutral knowledge model for DEA supports both representation of declarative and procedural knowledge, supports modularity and re-usability (Siricharoen, 2007).

8.3.1 Positioning of the Model in Comparison to Related Work

Work performed by (Usman, 2012; Usman et al., 2013) and (Chungoora, 2010; Chungoora et al., 2013a) has elaborated on the usage of Common Logic based PSL ontology as neutral formalised semantics for equivalent UML based lightweight formal representation for machining processes with knowledge sharing and access across product design. Their work caters to the needs of PLM systems and can also be used for automation purposes specially manufacturing and production automation. However, as discussed, pertaining to the engineering design domain, due to the lack of formal axioms for design systems such as functional requirements analysis and finer product attributes with form, fit and features (Cochrane et al., 2009; Young et al., 2007; Zhan et al., 2010) along with the lack of supporting tools for PSL in accordance with research design, OWL/SWRL based ontology has been adopted in line for the needs of developing neutral knowledge models for DEA.

Work has been performed in developing neutral knowledge model for DEA in context to a KBE approach specifically for the aerospace industry (Sanya and Shehab, 2015, 2014). Following the MOKA methodology and formulation of platform independent models for ensuring high abstraction, modularity and re-usability of represented knowledge, OWL/SWRL as a combination of semantic web representation language was chosen to formalise the design knowledge with Protégé as a tool. Although the knowledge model was based on functional requirements as the basis, more focus was laid on design intent in the

form of design parameters, constraints and rules for specific aerospace components such as compressors and turbines based on feature and shapes such as sleeve, panel and flanges as compared to the more generic and re-usable process oriented approach as part of this research. GPM-DEA developed as part of this research has been validated for wider applicability with use-cases from aerospace components with pilot use-cases, DFM aspect with drilling process and bookshelf design process. It was also recognised that using semantic web based languages such as OWL ontology for DEA with a KBE approach, there was a lack of common model based on a set of activities which would deploy the OWL based model for use in KBE applications with a lack of widely adopted ontology development for engineering design and DEA (Sanya and Shehab, 2014). This research bridges this gap by not only using OWL/SWRL as a platform independent and neutral representation of mechanical design knowledge with DFM for DEA in a KBE environment, but also providing clear and concise method of modelling of the knowledge into ontology development with reusable classes and properties in OWL using concepts and relationships in the structured knowledgebase as formulation of GPM-DEA schema. The population of GPM-DEA with multiple use-cases as instances verifies the effective working of the process model. The work carried out by Sanya and Shehab focussed on the usage of BPMN along with UML for process modelling on context to DEA as informal representation. ***Contrary to this approach, research work in this thesis has elaborated on the usage of IDEF0 and UML/SysML as the basis and then addition of concepts and relationships as illustrated in section 8.2.1 to formulate a more comprehensive informal process model with generative modelling capability with initial assessment of product as GPM-DEA***. It was also recognised that there was lack of research between ontology development and engineering design (Sanya and Shehab, 2015). This research also bridges this gap by merging and mapping engineering design aspects for DEA and ontology development using OWL/SWRL.

Post MOKA, another contribution was made by (Reijnders, 2012) in developing platform independent and formal representation of engineering design knowledge for aerospace industry for DEA with a KBE approach using a combination of OWL, RIF Production Rule Dialect (PRD) and Content MathML using a commercial implementation tool AllegroGraph based on Allegro Common Lisp platform. Although both product and process knowledge was represented, the main focus of the captured and represented knowledge was based on engineering rules for product design as compared to a process based approach performed in this thesis. MOKA ICARE forms were used as informal representation with the corresponding platform independent formal representation of rules in RIF-PRD and Content MathML (Reijnders, 2012). As explained earlier, this research has developed an advanced process model GPM-DEA that is much more comprehensive than MOKA ICARE forms for knowledge modelling or informal representation for mechanical design. In Reijnders work, although the forward reasoning works on the rules leading to the successful implementation of design knowledge, the predicates of the rules such as the antecedent and the consequent couldn't be queried due to integration between RIF-PRD and OWL leading to loss of contextual relevance of rules with co-related knowledge. On the contrary, this research has used SWRL, which offers ease of integration with OWL making the query on the internal predicates of the rules relatively easier thus also preserving the semantic clarity of the represented knowledge of GPM-DEA. Also, it was stated that single rules related to an object or a process were easily modelled, but multiple rules were difficult to implement. However, in this research multiple rules related to an object or a process have been modelled at the same level as a singular rule within the SWRL tab with the same ease of implementation for inference and querying.

Other work that was also similar in developing platform independent and neutral knowledge models for DEA with a KBE perspective was performed by (Lützenberger et al., 2012;

Pardalis and Kadiri, 2014; Pugliese and Colombo, 2014), where the authors recommended the usage of RIF, as the focus was purely on formal representation of engineering rules. The investigation of OWL/SWRL as the potential for representation of neutral knowledge models for DEA with a KBE perspective was recognised which is discussed in Table 2-3 in section 2.8 of Chapter 2. This research has bridged this gap by potential investigation of OWL/SWRL for knowledge representation for DEA based on the developed model GPM-DEA, along with Use Case 4 adopted from this project and verified by experimentation that OWL/SWRL as ontology and rule representation is successful as platform independent and neutral formal representation of mechanical design knowledge for automation.

Also, as compared to AMAAD (Van Der Velden et al., 2012) for DEA with a KBE perspective, this research has successfully provided a structured method to perform detailed activities with product architecture knowledge. This research has also provided the association of the activities of the process model with the working of the developed OWL/SWRL system attributes, which is explained in chapter 5.

Thus, as compared to the previous work by Sanya, Rejinders and LinkedDesign project, GPM-DEA provides a method to describe mechanical design process models with DFM in platform independent and neutral formal representation as OWL/SWRL enabling DEA with generative modelling capabilities and preserved semantics, with a KBE approach. The working of the GPM-DEA model in OWL/SWRL proves that logic based formalisms such as OWL based on DL and SWRL based on Horn Logic do have the potential capability as knowledge representation formalisms for DEA.

8.3.2 Integration and Extension of the Model to other Engineering Applications

The GPM-DEA working has been validated with multiple use cases varying from aerospace components such as compressor and fan blades design and manufacturing processes at a preliminary level to a simpler drilling process and bookshelf design process at the detailed

product attribute level. ***Even though the ontology model contains extensive manufacturing aspects, the verification of the model has not been performed for all complete manufacturing domains with tooling for e.g. additive manufacturing. Although the model provides the sub-functions as functional requirements of various CAE analysis processes, the testing and verification of the model with CAE analysis processes such as stress analysis, structural analysis and thermal analysis has not been performed. Thus the model may need extensions in its classes and relationships along with SWRL rules to fully cover the CAE analysis process lifecycle along with wider manufacturing domain with newer methods.***

In its present stage, the testing and verification of the model has proved that it is comprehensive for mechanical design, manufacturing and design for manufacturing (DFM)/design for assembly (DFA) stages of the product development lifecycle based on the functional requirements. The current model has proven to be generic and high level for a mechanical design process with manufacturing knowledge for DEA. The implementation of the model in OWL/SWRL can be extended for detailed manufacturing and production processes domain along with Design for Manufacturing (DFM) ontologies such as MASON and ONTO-PDM (Chang et al., 2010; Lemaignan et al., 2006; Panetto et al., 2012).

As the main strength and applicability of GPM-DEA is a process modelling approach with its effect on product attributes with an interface to the product model, its corresponding implementation in OWL/SWRL also provides compatibility with detailed product models with geometry kernels for visualisation, for DEA. This research provides scope of integration with previous work in developing semantic product models with geometric kernels using OWL/SWRL ontology across heterogeneous CAD systems with various product attributes as parameters, features and shapes such as surfaces, faces, edges (Lu et al., 2016; Qin et al., 2016; Tessier and Wang, 2013) which have been included in this research to show the effect

of the process model on product geometric attributes with an interface. This also includes features such as holes, extrusion and chamfering with Boolean representations, which have been embedded in this research with SWRL, making this directly compatible as a KR language for integration with GPM-DEA as a process model for DEA.

This work can also be integrated with non-geometric product models. An example as illustrated in section 3.7.5.2, an ontology was developed for UML based CPM/OAM product model with both non-geometric and geometric attributes along with function and behaviour, although it was not fully validated for visual display using geometry kernel as it was targeted for PLM systems (Fiorentini et al., 2007). Other work for integration to the process model in this research are ontology based neutral product models for visual display with geometry kernels across CAx systems, which have been developed. These include mapping of STEP based EXPRESS schemas to OWL/SWRL based ontologies in order to develop neutral product models with geometric knowledge such as Onto-STEP and ONTO-PDM (Barbau et al., 2012; Krima et al., 2009; Zhao and Liu, 2008a, 2008b). The reason for conversion of OWL ontology to STEP schemas for product models for geometric representation is that STEP is the current widely adopted neutral product model representation across various CAx and DEA systems. Thus, the OWL/SWRL process model of GPM-DEA provides a good foundation as KR formalism with automated reasoning to integrate with detailed platform independent and neutral product models with geometric kernels for DEA. As stated in section 3.7.1 and 3.7.5.2, work has been performed for capturing design rationale with the help of DRed (Design Rationale editor) and DRed 2.0 based on both UML/SysML and OWL/SWRL based ontology as formal representation for access in PLM systems and also across CAD applications (Bracewell et al., 2009a, 2009b; Eng et al., 2011). ***Although, GPM-DEA in the present stage doesn't include the Rationale class as rationale in not a necessary requirement for DEA, it can be added both informally based on UML notation and its***

corresponding ontological representation in OWL/SWRL. Thus GPM-DEA as a knowledgebase can be extended with rationale for mechanical design process. Similarly, although presently, GPM-DEA is quite exhaustive for function and behaviour as FBS for a mechanical design process with manufacturing knowledge for DEA, it can be extended and merged with functional and behavioural aspect of other engineering processes and products both informally and with ontologies as formal representation. The working of the OWL/SWRL model with drilling a hole in a block and bookshelf design process has been validated inside platform specific DEA systems such as KBE based AML, ParaPy and GA based parametric CAD applications such as CATIA Knowledgeware and Siemens NX KF at the product geometric attribute level. Thus the OWL/SWRL model of GPM-DEA with its interface to the product model to illustrate the effect of mechanical design process on the geometric attributes of the product, can be used as a basis for integrating with a product model in neutral format using a front-end visual DEA application with product form, shapes and features using X3D (Web3D, 2017) based geometry kernels. Along with extension to wider domain such as design rationale, function and behaviour of engineering design and manufacturing, it can also be used as a back end platform for visualisation of queries and inference results to the design engineer for decision support and DEA with the support of semantic web pages. This visualisation of automation results over the semantic web pages can be achieved with the help of an API written on OWL/SWRL with languages such as Java such as those supported by Apache Jena framework.

8.4 Contributions to Knowledge

a. This research has developed a standardised and coherent method to use ontology based structured knowledge model as formal representation to address Design Engineering Automation (DEA) for mechanical design and DFM process with a KBE

perspective with semantic clarity and generative modelling by building queries and reasoning on author's set of generic SWRL functions.

b. The method to use OWL/SWRL ontology is based on the schema of developed informal/semiformal model GPM-DEA as a structured knowledge modelling method, based on author's Meta model which is built on strengths of IDEF0, UML/SysML and addition of modelling constructs by the author.

The main strengths and contribution of this research work are -

8.4.1 Model Driven Approach for Knowledge Modelling and Automation for Mechanical Design Process with DFM

The knowledge modelling method through GPM-DEA with an MBSE approach provides a generic, re-usable process model with transparency and traceability of concepts and relationships as per author's Meta model based on activity, product attributes, rules and logic, function-functional requirements, behaviour for mechanical design processes with DFM. It is based on F-B-S based modelling and includes functional requirements analysis, activity-object-rule association and an *interface to the product model* with geometric attributes and form-features-fit, thus including both geometric and non-geometric knowledge to cover and address automation for mechanical product design process with DFM/DFA. Thus the knowledgebase acts as superset of platform specific DEA applications.

8.4.2 Utilisation of Formal Logic for Implementation of a Process Model for DEA

The successful implementation of GPM-DEA with OWL/SWRL ontology and rule representation proves that *formal logic is able to capture the semantic meaning of various mechanical design process concepts and properties* with inclusion of manufacturing knowledge. Thus it provides a suitable machine interpretation of mechanical design knowledge for DEA with its automated reasoning on the formal axioms as syntax with depth of meaning of classes and relationships as concepts and bi-directional properties with OWL

(DL) and addition of forward chaining reasoning capability on classes and properties using SWRL (Horn logic) with math, boolean and comparison built-ins. The inclusion of float datatype properties ensures that product parameters as geometric attributes can be included in the model although it depends upon careful execution of the OWL/SWRL model.

8.4.3 Neutral (Open Standard) Usage of the Ontology Knowledge Model across Platform Specific DEA Systems with Semantic Clarity

The developed process model GPM-DEA with its mapping to equivalent OWL/SWRL representation as platform independent formal representation with semantic clarity provides a structured method to use formal ontologies for DEA with a KBE perspective within a virtual engineering environment. Ontology provides open standard usage and provides neutral knowledge model outside of platform specific DEA applications such as KBE based AML, ParaPy and GA based CATIA Knowledgeware, Siemens NX KF.

8.4.4 Extensibility and Scalability of the Knowledge Base

The model offers ease of extensibility with the aid of formal OWL/SWRL representation. Ontology based on formal logic with semantic clarity provides scalability with addition of classes, properties and instances. ***The model can be extended to cover other aspects of engineering knowledge depending upon the end user such as design rationale, function-behaviour and product data models, advanced and detailed manufacturing, maintenance and operations for production including tooling***. The new knowledge objects can be easily integrated or merged in the OWL/SWRL ontology representation to cater to specific engineering requirements.

8.4.5 Web Based Decision Support for Engineering Applications

The knowledge within the platform independent and neutral model can be extracted to platform specific DEA applications or web pages to provide decision support for a wider design space exploration for the designer by developing an API on the OWL/SWRL model.

These can be developed using languages such as Java. Ontology models can be directly exported to Java code within Protégé IDE. Methods have been devised to map OWL/SWRL ontology methods to O-O programming which can pave the way for retrieving the knowledge in neutral file format, developed as part of this research, for direct utilisation inside the proprietary DEA applications.

8.4.6 Integration of Generative Modelling Capability within Process Model

The developed formal model enables *generative modelling capabilities* by building queries and reasoning on author's generic set of SWRL functions by automatically generating the activities and objects based on the *functional requirements as sub-function structures* of the mechanical design process with DFM along with process sequencing. It also provides initial assessment of a product to adapt and provide re-usability of processes and activities for different products. The automatic generation of the activities, objects based on matching the functional requirements as sub-function structures to those of the mechanical design process along with the initial assessment of product is achieved with implementation of developed *SWRL functions* as part of this research. For engineering rules based on logic, the rules are automatically generated based on SWRL functions by matching the engineering logic structures developed as part of this research. All the *SWRL inference and query results* have been validated during experimentation including the execution of generic and product specific engineering rules for block and bookshelf usage as test use-cases.

8.4.7 Ontology Representation of Design and Manufacturing Knowledge within a Unified Process Model

The process model includes manufacturing knowledge and DFM/DFA aspects during the mechanical design stage and represents both physical and virtual representation of the products in context to mechanical product design with DFM processes. Both *design and manufacturing requirements* have been included in the *functional requirements* (equivalent to

function) class and sub-class as function structures with instantiation. The individual activities can be classified as physical, virtual and informatical and the equivalence between physical and virtual representation is achieved with the SWRL functions developed as part of this research.

8.5 Limitations

Although the research contributes to the body of scientific knowledge by satisfying the aim and objectives thus verifying the research hypothesis, there are a few limitations due to the scope and the context in which the results are valid.

Firstly, the focus of both pilot and test use-cases collected from industrial partner and literature is on mechanical design, DFM with manufacturing processes as part of product development cycle. Although the ontology model is quite exhaustive, it has not been verified through use cases for all aspects of manufacturing/production methods with tooling such as additive manufacturing. Although CAE analysis process concepts such as stress analysis, thermal analysis, structural analysis have been included as subclasses in the OWL/SWRL model for GPM-DEA, the model has not been instantiated or populated and verified with analysis process use-cases to validate the implementation results. Also, the complexity of the model based on Meta model concepts such as activity, product attributes, rule and logic, function-functional requirements and behaviour may need extension to cover these other engineering processes not covered in this research. Secondly, the reasoning results of GPM-DEA as a process model with an interface to the product model on OWL/SWRL as formal logic based representation generates both text and numeric values for product parameters as geometric attributes as described within a CAx virtual platform. However it doesn't incorporate the visual representation of product form, shape and features through its geometry kernels. In spite of the limitations, the model is widely applicable to mechanical design and manufacturing along with DEA both within a KBE context and GA based

parametric CAD automation, which proves that GPM-DEA is robust, structured, generic and re-usable as extensions can be applied within a specific domain for highly granular capabilities within the mechanical design space.

8.6 Recommendations for Future Work

Based on the results of this research work, further work can be conducted in the following areas for the applicability of this research to a wider problem domain -

- The integration of geometry kernels for detailed product model visual representation through a GUI in terms of its form, features and shapes using neutral format such as X3D for DEA. This will help the designer visualise the direct impact of the process model on the geometry with open standards. This can cover different kernels such as NURBS, splines and closed profiles for surface along with extrusion, pockets, notch for volume representation as part of neutral product model
- The mapping or equivalent formal representation of GPM-DEA in OWL/SWRL ontology as a proof–of-concept follows a manual approach in accordance with research design to ensure the correctness of formal syntax, preserved semantics and detailed implementation for accurate reasoning results. Although the inference and query results are found accurate for the test use-cases during experimentation, the process of populating the knowledgebase is slightly time consuming. In order to reduce the translation time for high volume use-cases and industrial implementation, automatic mapping can be addressed to a certain extent from GPM-DEA schema as informal/semiformal process model to OWL/SWRL knowledge model
- For platform independent and neutral formal representation of mathematical rules with complex equations currently not supported by built-in SWRL plugins, MathML with various dialects such as Presentation MathML with focus on Content MathML can be investigated for integration on top of OWL/SWRL as an additional layer.

8.7 Closing Summary

GPM-DEA is a process model with F-B-S modelling, based on authors Meta model, as an MBSE approach and its effect on the product parameters as geometric attributes with form-features-fit through an interface. It combines the strengths of UML/SysML and IDEF0 and addition on authors constructs, is a high level, generic, re-usable and extensible process model for knowledge modelling of mechanical design processes with incorporation of DFM/DFA as manufacturing knowledge. The model enables DEA through OWL/SWRL as a platform independent and neutral formal representation with generative modelling based on generic SWRL functions developed by the author. The development of GPM-DEA follows a model driven approach with equivalent ontology and rule representation as neutral standards with open standard usage. OWL/SWRL provides combination of DL and horn logic based formal logic representation with automated reasoning capabilities for the developed process model GPM-DEA to satisfy the needs of DEA. The inference and query results on OWL/SWRL have been experimentally verified at generic as well as product specific level for mechanical design, manufacturing and DFM as part of engineering processes. GPM-DEA can be extended or merged with other function-behaviour, rationale, product data models and integrate with manufacturing and production domain both at the informal level and at the OWL/SWRL as formal model. Thus a contribution to knowledge has been made in terms of fulfilment of aim and objectives, which verifies the research hypothesis and can be stated as-

> "Platform independent and neutral formal representation of an engineering design process model with focus on mechanical product design and manufacturing knowledge built on standardised concepts and relationships, structured and well defined axioms along with semantic clarity can achieve the requirements of design engineering automation (DEA) enabling generative modelling and re-usability of knowledge"

References

Abdullah, M.S., Kimble, C., Paige, R., Benest, I., Evans, A., 2005. Developing a UML profile for modelling knowledge-based systems, in: Model Driven Architecture. Springer, pp. 220–233.

Aguilar-Saven, Ruth, S., 2004. Business process modelling: Review and framework. Int. J. Prod. Econ. 90, 129–149.

Ahmed, S., Kim, S., Wallace, K.M., 2007. A methodology for creating ontologies for engineering design. J. Comput. Inf. Sci. Eng. 7, 132–140.

Al-Ahmari, A.M.A., Ridgway, K., 1999. An integrated modelling method to support manufacturing systems analysis and design. Comput. Ind. 38, 225–238.

Alexandrou, D., Clobes, J., Maza, S. de la, Parrotta, S., Domenica, E.M., Louw, E., Buda, A., Kristensen, K., Pardalis, K., Iversen, G., Milicic, A., Lutzenberger, J., Wartner, C., 2013. D10.6 LinkedDesign Demonstration Plan, Linked Knowledge in Manufacturing, Engineering and Design for Next-Generation Production. LinkedDesign Consortium. LinkedDesign Consortium.

Alvarez Cabrera, A.A., Erden, M.S., Tomiyama, T., 2009. On the Potential of Function-Behavior-State (FBS) Methodology for the Integration of Modeling Tools. Proc. 19th CIRP Des. Conf. Des. 30–31.

Amadori, K., 2012. Geometry Based Design Automation: Applied to Aircraft Modelling and Optimization. Doctoral dissertation, Linköping University Electronic Press. doi:10.1016/S0965-9978(01)00041-2

Amigo, C.R., Iritani, D.R., Rozenfeld, H., Ometto, A., 2013. Product development process modeling: state of the art and classification. Smart Prod. Eng. Springer B, 169–179.

Ammar-Khodja, S., Perry, N., Bernard, a., 2008. Processing Knowledge to Support Knowledge-based Engineering Systems Specification. Concurr. Eng. 16, 89–101. doi:10.1177/1063293X07084642

Amoo, L.M., 2013. On the design and structural analysis of jet engine fan blade structures. Prog. Aerosp. Sci. 60, 1–11. doi:10.1016/j.paerosci.2012.08.002

Andrews, P.T.J., Shahin, T.M.M., Sivaloganathan, S., 1999. Design reuse in a CAD environment — Four case studies. Comput. Ind. Eng. 37, 105–109. doi:10.1016/S0360-8352(99)00033-9

Antoniou, G., Van Harmelen, F., 2004. A Semantic Web Primer. MIT press. doi:10.5860/CHOICE.46-1523

Ausbrooks, R., Buswell, S., Carlisle, D., Chavchanidze, G., Dalmas, S., Devitt, S., Diaz, A., Dooley, S., Hunter, R., Ion, P., Kohlhase, M., Lazrek, A., Libbrecht, P., Miller, B., Miner, R., Rowley, C., Sargent, M., Smith, B., Soiffer, N., Sutor, R., Watt, S., 2014. Mathematical Markup Language (MathML) Version 3.0 2nd Edition [WWW Document]. W3C Recomm. URL http://www.w3.org/TR/MathML3/ (accessed 11.1.15).

Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., 2003. The Description Logic Handbook: Theory, Implementation, and Applications. Descr. Log. Handb. 622. doi:10.2277/0521781760

Badica, A., Badica, C., 2011. Formal Verification of Business Processes as Role Activity

Diagrams, in: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS).

Badica, C., Badica, A., Litoiu, V., 2003. Role activity diagrams as finite state processes, in: Second International Symposium on Parallel and Distributed Computing (ISPDC'03). doi:10.1109/ISPDC.2003.1267638

Badica, C., Teodorescu, M., Spahiu, C., Badica, A., Fox, C., 2005. Integrating Role Activity Diagrams and Hybrid IDEF for Business Process Modeling Using MDA, in: SYNASC. pp. 71–74.

Ball, M., Boley, H., Hirtle, D., Mei, J., Spencer, B., 2005. Implementing RuleML Using Schemas, Translators, and Bidirectional Interpreters [WWW Document]. W3C. URL https://www.w3.org/2004/12/rules-ws/paper/49/ (accessed 12.1.16).

Bancroft, C.N., Crump, S.J., Lovett, P.J., Bone, D., Kightley, N.J., 2000. Taking KBE into the Foundry. Proc. 7th ISPE Int. Conf. Concurr. Eng. 24, 17–20.

Barbau, R., Krima, S., Rachuri, S., Narayanan, A., Fiorentini, X., Foufou, S., Sriram, R.D., 2012. OntoSTEP: Enriching product model data using ontologies. Comput. Des. 44, 575–590. doi:10.1016/j.cad.2012.01.008

Barkmeyer, E.J., Feeney, A.B., Denno, P., Flater, D.W., Libes, D.E., Steves, M.P., Wallace, E.K., 2003. Concepts for automating systems integration, National Institute of Standards and Technology, Technical Report. doi:10.1109/ICASSP.2004.1326632

Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., Su, J., Tabet, S., 2005. Semantic Web Services Language (SWSL) [WWW Document]. W3C Memb. Submiss. URL http://www.w3.org/Submission/SWSF-SWSL/ (accessed 11.1.15).

Baxter, D., Gao, J., Case, K., Harding, J., Young, B., Cochrane, S., Dani, S., 2007. An engineering design knowledge reuse methodology using process modelling. Res. Eng. Des. 18, 37–48.

Bechhofer, S., 2009. OWL: Web Ontology Language. Encycl. Database Syst.

Beckett, D., McBride, B., 2004. RDF/XML Syntax Specification (Revised) [WWW Document]. W3C Recomm. URL http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/ (accessed 6.1.15).

Benjamin, P.C., Menzel, C.P., Mayer, R.J., Fillion, F., Futrell, M.T., Dewitte, P.S., Lingineni, M., 1994. Information Integration for Concurrent Engineering (IICE) IDEF5 Method Report.

Bermell-Garcia, P., 2007. A metamodel to annotate knowledge based engineering codes as enterprise knowledge resources. Cranfield University. doi:10.1017/CBO9781107415324.004

Bermell-García, P., Fan, I.-S., 2002. A kbe system for the design of wind tunnel models using reusable knowledge components. doi:DOI

Bermell-garcia, P., Fan, I., Murton, A., 2007. Towards the Semantic Interoperability Between Kbe and Plm Systems. Artif. Intell. 1–12.

Bermell-Garcìa, P., Fan, I.S., Li, G., Porter, R., Butter, D., 2001. Effective abstraction of engineering knowledge for KBE implementation, in: ICED (Vol. 1., No. 1). pp. 99–106.

Bernard, A., 2005. Virtual engineering: Methods and tools. Proc. Inst. Mech. Eng. Part B J.

Eng. Manuf. 219, 413–421. doi:10.1243/095440505X32238

Beynon-Davies, P., Carne, C., Mackay, H., Tudhope, D., 1999. Rapid Application Development (RAD): an empirical review. Eur. J. Inf. Syst. 8, 211–223.

Bhaskara, S., 2010. DSM Based Approach for Managing Requirements, Rules and Design Parameters in Knowledge Based Design Process, in: DSM 2010: Proceedings of the 12th International DSM Conference, Cambridge, UK, 22.-23.07. 2010.

Blekhman, A., Dori, D., 2013. Tesperanto – A Model-Based System Specification Methodology and Language. INCOSE Int. Symp. 23, 139–153. doi:10.1002/j.2334-5837.2013.tb03009.x

Blekhman, A., Wachs, J.P., Dori, D., 2015. Model-Based System Specification With Tesperanto: Readable Text From Formal Graphics. Syst. Man, Cybern. Syst. IEEE Trans. PP, 1. doi:10.1109/TSMC.2015.2406753

Bluntzer, J.-B., Gomes, S., Sagot, J.-C., 2009. Definition of a Knowledge Representation Based on Functional CAD Models. DS 58-8 Proc. ICED 09, 17th Int. Conf. Eng. Des. Vol. 8, Des. Inf. Knowledge, Palo Alto, CA, USA, 24.-27.08. 2009.

Bock, C., 2006. Interprocess Communication in the Process Specification Language, US Department of Commerce, National Institute of Standards and Technology. Citeseer.

Bock, C., Gruninger, M., 2005. PSL: A semantic domain for flow models. Softw. Syst. Model. 4, 209–231. doi:10.1007/s10270-004-0066-x

Bock, C., Gruninger, M., 2004. Inputs and Outputs in the Process Specification Language, US Department of Commerce, Technology Administration, National Institute of Standards and Technology. Citeseer.

Bodein, Y., Rose, B., Caillaud, E., 2009. Improving CAD Performance: A Decisional Model for Knowledgeware Implementation. Proc. Int. Conf. Eng. Des. 311–322.

Boley, H., Grosof, B., Tabet, S., 2005. RuleML Tutorial [WWW Document]. URL http://ruleml.org/papers/tutorial-ruleml-20050513.html (accessed 4.1.15).

Boley, H., Kifer, M., 2013. RIF Basic Logic Dialect (Second Edition) [WWW Document]. W3C Recomm. URL https://www.w3.org/TR/rif-bld/ (accessed 12.1.15).

Boley, H., Paschke, A., Tabet, S., 2016a. Specification of RuleML [WWW Document]. RuleML. URL http://wiki.ruleml.org/index.php/Specification_of_RuleML (accessed 10.1.15).

Boley, H., Paschke, A., Tabet, S., 2016b. RuleML Home [WWW Document]. RuleML. URL http://wiki.ruleml.org/index.php/RuleML_Home (accessed 7.1.15).

Boley, H., Paschke, A., Tabet, S., 2016c. Introducing RuleML [WWW Document]. RuleML. URL http://wiki.ruleml.org/index.php/Introducing_RuleML (accessed 5.1.16).

Booch, G., Rumbaugh, J., Jacobson, I., 1999. The Unified Modeling Language User Guide. Pearson Education India.

Bos, B., Carlisle, D., Chavchanidze, G., Ion, P.D.F., Miller, B.R., 2011. A MathML for CSS Profile [WWW Document]. W3C Recomm. URL https://www.w3.org/TR/mathml-for-css/ (accessed 2.1.16).

Braaksma, A.J., Klingenberg, W.W., van Exel, P.P., 2011. A review of the use of asset information standards for collaboration in the process industry. Comput. Ind. 62, 337–350. doi:10.1016/j.compind.2010.10.003

Bracewell, R., Gourtovaia, M., Moss, M., Knott, D., Wallace, K., Clarkson, P.J., 2009a. DRed 2.0: a method and tool for capture and communication of design knowledge deliberated in the creation of technical products, in: DS 58-6: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 6, Design Methods and Tools (Pt. 2), Palo Alto, CA, USA, 24.-27.08. 2009.

Bracewell, R., Wallace, K., Moss, M., Knott, D., 2009b. Capturing design rationale. Comput. Des. 41, 173–186. doi:10.1016/j.cad.2008.10.005

Bracewell, R.H., Ahmed, S., Wallace, K.M., 2004. DRed and design folders: a way of capturing, storing and passing on-knowledge generated during design projects, in: Proceedings of the ASME Design Engineering Technical Conference, 1pp. American Society of Mechanical Engineers, pp. 235–246. doi:10.1115/DETC2004-57165

Browning, T.R., 2009. The Many Views of a Process: Toward a Process Architecture Framework for Product Development Processes. Syst. Eng. 12, 69–90.

Browning, T.R., 2002. Process Integration Using the Design Structure Matrix. Syst. Eng. 5, 180–193.

Browning, T.R., Fricke, E., Negele, H., 2006. Key concepts in modeling product development processes. Syst. Eng. 9, 104–128.

Bruijn, J. de, Welty, C., 2013. RIF RDF and OWL Compatibility (Second Edition) [WWW Document]. W3C Recomm. URL https://www.w3.org/TR/rif-rdf-owl/ (accessed 4.1.16).

Brunetti, G., Golob, B., 2000. A feature-based approach towards an integrated product model including conceptual design information. Comput. Des. 32, 877–887.

Bruun, H.P.L., Mortensen, N.H., Harlou, U., Wörösch, M., Proschowsky, M., 2015. PLM system support for modular product development. Comput. Ind. 67, 97–111. doi:10.1016/j.compind.2014.10.010

Bullinaria, J.A., 2005. IAI : Knowledge Representation 1–20.

Burkett, M., O'Marah, K., Carrillo, L., 2003. CAD Versus ERP Versus PDM: How Best To Anchor a PLM Strategy?, AMR Research, Sept.

Calkins, D., Egging, N., Scholz, C., 2000. Knowledge-based engineering (KBE) design methodology at the undergraduate and graduate levels [J]. Int. J. Engng Ed. 16, 21–38.

Ćatić, A., Malmqvist, J., 2007. Towards integration of KBE and PLM, in: Proceedings of the International Conference on Engineering Design (ICED07), Paper.

Cederfeldt, M., Elgh, F., 2005. Design Automation in SMEs-Current State, Potential, Need and Requirements, in: DS 35: Proceedings ICED 05, the 15th International Conference on Engineering Design, Melbourne, Australia, 15.-18.08. 2005.

Chalupnik, M., Eckert, C., Clarkson, P., 2006. Modelling design processes to improve robustness, in: IPD 2006: 6th Workshop on Integrated Product Development, Magdeburg, Germany, 18.-20.09. 2006.

Chan, P.K.M., 2013. A new methodology for the development of simulation workflows: Moving beyond MOKA. Delft University of Technology. doi:10.1007/978-0-387-35350-0_3

Chandrasegaran, S.K., Ramani, K., Sriram, R.D., Horváth, I., Bernard, A., Harik, R.F., Gao, W., 2013. The evolution, challenges, and future of knowledge representation in product design systems. Comput. Des. 45, 204–228. doi:10.1016/j.cad.2012.08.006

Chang, X., Rai, R., Terpenny, J., 2010. Development and Utilization of Ontologies in Design for Manufacturing. J. Mech. Des. 132, 21009. doi:10.1115/1.4000697

Chang, X., Sahin, A., Terpenny, J., 2008. An ontology-based support for product conceptual design. Robot. Comput. Integr. Manuf. 24, 755–762. doi:10.1016/j.rcim.2008.03.004

Chapman, C.., Pinfold, M., 1999. Design engineering—a need to rethink the solution using knowledge based engineering. Knowledge-Based Syst. 12, 257–267.

Chapman, C., Preston, S., Pinfold, M., Smith, G., 2007. Utilising enterprise knowledge with knowledge-based engineering. Int. J. Comput. Appl. Technol. 28, 169–179.

Chapman, C.B., Pinfold, M., 2001. The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure. Adv. Eng. Softw. 32, 903–912.

Chen, A.-P., Chen, M.-Y., 2005. A Unifying Ontology Modeling for Knowledge Management, in: Knowledge-Based Intelligent Information and Engineering Systems. Springer Berlin Heidelberg, pp. 318–324.

Chen, Y.-M.J.M.Y.-J., Chen, Y.-M.J.M.Y.-J., Chu, H.-C.C., Kao, H.-Y.Y., 2008. On technology for functional requirement-based reference design retrieval in engineering knowledge management. Decis. Support Syst. 44, 798–816. doi:10.1016/j.dss.2007.10.003

Christophe, F., 2012. Semantics and Knowledge Engineering for Requirements and Synthesis in Conceptual Design: Towards the Automation of Requirements Clarification and the Synthesis of Conceptual Design Solutions. Dr. Diss. Ec. Cent. Nantes (ECN)(ECN)(ECN)(ECN); Aalto Univ.

Chulvi, V., Sancho, A., Cebrián, D., Jiménez, R., Muñoz, C., Vidal, R., 2007. Knowledge-Based Engineering in Cranioplasty Implant Design, in: Proceedings of the 16th International Conference on Engineering Design (ICED'07), Paris. pp. 365–384.

Chung, J., Lee, K., 2002. A framework of collaborative design environment for injection molding. Comput. Ind. 47, 319–337. doi:10.1016/S0166-3615(02)00004-0

Chungoora, N., 2010. A Framework to Support Semantic Interoperability in Product Design and Manufacture. Glob. Prod. Dev. doi:10.1007/978-3-642-15973-2_44

Chungoora, N., Cutting-Decelle, A.-F., Young, R., Gunendran, G., Usman, Z., Harding, J.A., Case, K., 2013a. Towards the ontology-based consolidation of production-centric standards. Int. J. Prod. Res. 51, 327–345. doi:10.1080/00207543.2011.627885

Chungoora, N., Young, R.I., Gunendran, G., Palmer, C., Usman, Z., Anjum, N.A., Cutting-Decelle, A.F., Harding, J.A., Case, K., 2013b. A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. Comput. Ind. 64, 392–401. doi:10.1016/j.compind.2013.01.003

Chungoora, N., Young, R.I.M., 2011. The configuration of design and manufacture knowledge models from a heavyweight ontological foundation. Int. J. Prod. Res. 49, 4701–4725. doi:10.1080/00207543.2010.504754

Ciocoiu, M., Nau, D.S., Gruninger, M., 2001. Ontologies for Integrating Engineering Applications. J. Comput. Inf. Sci. Eng. 1, 12–22. doi:10.1115/1.1344878

Clark, P., 1996. Requirements For a Knowledge Representation System: Working Note 10.

Clarkson, P.J., Hamilton, J.R., 2000. "Signposting", a parameter-driven task-based model of

the design process. Res. Eng. Des. 12, 18–38.

Cochrane, S., Young, R., Case, K., Harding, J., Gao, J., Dani, S., Baxter, D., 2009. Manufacturing knowledge verification in design support systems. Int. J. Prod. Res. 47, 3179–3204. doi:10.1080/00207540701802452

Colledani, M., Terkaj, W., Tolio, T., Tomasella, M., 2008. Development of a Conceptual Reference Framework to Manage Manufacturing Knowledge Related to Products, Processes and Production Systems, in: Methods and Tools for Effective Knowledge Life-Cycle-Management. Springer Berlin Heidelberg, pp. 259–284. doi:10.1007/978-3-540-78431-9_15

COLOMBO, G., CUGINI, U., PUGLIESE, D., PULLI, M., 2005. Levels of knowledge representation for Product Design, in: PLM'05: International Conference on Product Life Cycle Management. pp. 137–146.

Colombo, G., Pugliese, D., Klein, P., Lutzemnberger, J., 2014. A study for neutral format to exchange and reuse engineering knowledge in KBE applications, in: Engineering, Technology and Innovation (ICE), 2014 International ICE Conference On. IEEE. IEEE, pp. 1–10.

Colquhoun, G.J., Baines, R.W., Crossley, R., 1993. A state of the art review of IDEF0. Int. J. Comput. Integr. Manuf. 6, 252–264.

Composer, T., 2011. TopBraid Composer ™ Getting Started Guide Version 3.0. TopQuadrant, Inc. (2011) July 18th.

Cooper, D., LaRocca, G., 2007. Knowledge-based Techniques for Developing Engineering Applications in the 21st Century, in: 7th AIAA ATIO Conference, Belfast, Northern Ireland. Belfast, Northern Ireland.

Cooper, S., Fan, I., Li, G., 1999. Achieving Competitive Advantage Through Knowledge-Based Engineering: A Best Practice Guide, Prepared by Dept. of Enterprise Integration, Cranfield University.

Corallo, A., Laubacher, R., Margherita, A., Turrisi, G., 2009. Enhancing product development through knowledge-based engineering (KBE): A case study in the aerospace industry. J. Manuf. Technol. Manag. 20, 1070–1083. doi:10.1108/17410380910997218

Creswell, J.W., 2003. Research Design - Qualitative, Quantitative, and Mixed Methods Approaches, Second. ed. SAGE Publications.

Curran, R., Verhagen, W.J., Van Tooren, M.J., 2010. The KNOMAD methodology for integration of multi-disciplinary engineering knowledge within aerospace production. Am. Inst. Aeronaut. Astronaut. doi:10.2514/6.2010-1315

Danjou, S., Lupa, N., Koehler, P., 2008. Approach for Automated Product Modeling Using Knowledge-Based Design Features. Comput. Aided. Des. Appl. 5, 622–629. doi:10.3722/cadaps.2008.xxx-yyy

Dartigues, C., Ghodous, P., Gruninger, M., Pallez, D., Sriram, R., 2007. CAD/CAPP integration using feature ontology. Concurr. Eng. 15, 237–249. doi:10.1177/1063293X07079312

Das, B., Cutting-Decelle, A.-F., Young, R.I., 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. Int. J. Comput. Integr. Manuf. 20, 396–410. doi:10.1080/09511920600873741

Davis, R., Shrobe, H., Szolovits, P., 1993. What is a Knowledge Representation? AI Mag. 14, 17.

Dean, M., Schreiber, G., Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A., 2004. OWL Web Ontology Language Reference, W3C Recommendation February.

Deshayes, L.M.L.M., Beqqali, O.E., Bouras, A., El Beqqali, O., Bouras, A., 2005. The use of Process Specification Language for cutting processes. Int. J. Prod. Dev. 2, 236–253.

Ding, L., Giess, M., Goh, Y.M., McMahon, C.A., Thangarajah, U., 2009. Component-based records: A novel method to record transaction design work. Adv. Eng. Informatics 23, 332–347. doi:10.1016/j.aei.2009.03.001

Dong, Y., Chee, C.F.Y., He, Y., Goh, A., 1997. Active database support for STEP/EXPRESS models. J. Intell. Manuf. 8, 251–261. doi:10.1023/A:1018581426556

Dori, D., 2004. ViSWeb - The Visual Semantic Web: Unifying human and machine knowledge representations with Object-Process Methodology. VLDB J. 13, 120–147. doi:10.1007/s00778-004-0120-x

Dori, D., 2002. Object-Process Methodology: A Holistic Systems Paradigm. Springer Science & Business Media, Berlin. doi:10.1007/978-3-642-56209-9

Dori, D., Howes, D., Blekhman, A., Martin, R., 2010. OPM as a Basis for Model-Based Enterprise Standards. Tokyo.

Eckert, C., Albers, A., Bursac, N., Chen, H.X., Clarkson, P.J., Gericke, K., Gladysz, B., Maier, J.F., Rachenkova, G., Shapiro, D., Wynn, D., 2015. Integrated product and process models: Towards an integrated framework and review, in: Proceedings of 20th International Conference on Engineering Design. pp. 1–10.

El Kadiri, S., Kiritsis, D., 2015. Ontologies in the context of product lifecycle management: state of the art literature review. Int. J. Prod. Res. 53, 5657--5668.

El Kadiri, S., Terkaj, W., Urwin, E.N., Palmer, C., Kiritsis, D., Young, R., 2015. Ontology in Engineering Applications, in: International Workshop Formal Ontologies Meet Industries. Springer, pp. 126–137. doi:10.1007/978-3-319-21545-7

Elgh, F., 2008. Supporting management and maintenance of manufacturing knowledge in design automation systems. Adv. Eng. Informatics 22, 445–456. doi:10.1016/j.aei.2008.05.004

Elgh, F., 2007. Computer-Supported Design for Producibility : Principles and Models for System Realisation and Utilisation. Product and Production Development, Chalmers University Of Technology, Gothenburg, Sweden.

Elgh, F., 2006. Automated cost estimation of product variants-a tool for enhanced producibility. Chalmers University of Technology.

Elgh, F., Johansson, J., 2014. Knowledge Object-a Concept for Task Modelling Supporting Design Automation., in: ISPE CE. pp. 192–203. doi:10.3233/978-1-61499-440-4-192

Eng, N., Aurisicchio, M., Bracewell, R., Armstrong, G., 2011. More space to think: Eight years of visual support for rationale capture, creativity and knowledge management in aerospace engineering, in: DETC/CIE. American Society of Mechanical Engineers. doi:10.1115/DETC2011-47911

Eppinger, S.D., Whitney, D.E., Smith, R.P., Gebala, D.A., 1994. A Model-Based Method for Organizing Tasks in Product Development. Res. Eng. Des. 6, 1–13.

Erden, M.S., Komoto, H., Van Beek, T.J., D'Amelio, V., Echavarria, E., Tomiyama, T., 2008. A Review of Function Modeling: Approaches and Applications. Artif. Intell. Eng. Des. Anal. Manuf. 22, 147–169. doi:10.1017/S0890060408000103

Estefan, J.A., 2007. Survey of Model-Based Systems Engineering (MBSE) Methodologies. Incose MBSE Focus Gr. 25, 1–12. doi:10.1109/35.295942

Evenson, M., Huelsman, E., Mommer, M.S., Yang, C., 2015. Common Lisp [WWW Document]. Common Lisp Found. URL https://common-lisp.net/ (accessed 10.1.16).

Feigenbaum, L., Booth, D., Cyganiak, R., Manola, F., Brickley, D., 2013. RIF FAQ [WWW Document]. W3C Powered by MediaWiki. URL https://www.w3.org/2005/rules/wiki/RIF_FAQ (accessed 10.1.16).

Fellmann, M.., Zarvić, N.., Sudau, a. ., Nobbe, L., 2013. Ontology-Based Assistance for Semi-Formal Process Modeling. Proc. 5th Int. Work. Enterp. Model. Inf. Syst. Archit. (EMISA 2013) 119–132.

Fenves, S.J., 2001. A core product model for representing design information, US Department of Commerce, Technology Administration, National Institute of Standards and Technology.

Fenves, S.J., Foufou, S., Bock, C., Sriram, R.D., 2008. CPM2: a core model for product data. J. Comput. Inf. Sci. Eng. 8, 14501.

Fernández-López, M., Gómez-Pérez, A., Juristo, N., 1997. METHONTOLOGY: From Ontological Art Towards Ontological Engineering, in: Ontological Engg. AAAI-97 Spring Symposium Series, Stanford University, Stanford. pp. 33–40. doi:10.1109/AXMEDIS.2007.19

Finance, G., 2010. SysML Modelling Language explained.

Fiorentini, X., Gambino, I., Liang, V., Foufou, S., Rachuri, S., Bock, C., Mahesh, M., 2007. Towards an ontology for open assembly model, in: International Conference on Product Lifecycle Management. pp. 445–456.

FIPS PUBS, 1993. Announcing the Standard for: Integration Definition for Function Modelling (IDEF0), Draft Federal Information Processing Standards Publication 183.

Foderaro, J., 1991. LISP: introduction. Commun. ACM 34, 27.

Foufou, S., Fenves, S.J., Bock, C., Rachuri, S., Sriram, R.D., 2005. A core product model for plm with an illustrative xml implementation. Int. Conf. Prod. Lifecycle Manag. 21–32.

Främling, K., Terzi, S., Louw, E., Potter, D., Parmar, S., Maharjan, M., Voigt, K., 2012. D4.1 Knowledge exploitation framework – architecture and bundle draft specification, Linked Knowledge in Manufacturing, Engineering and Design for Next-Generation Production. LinkedDesign Consortium. LinkedDesign Consortium.

France, R.B., Ghosh, S., Dinh-Trong, T., Solberg, A., 2006. Model-Driven Development Using UML 2.0: Promises and Pitfalls. Computer (Long. Beach. Calif). 39, 59–66. doi:10.1109/MC.2006.65

Frank, G., Entner, D., Prante, T., Khachatouri, V., Schwarz, M., 2014. Towards a Generic Framework of Engineering Design Automation for Creating Complex CAD Models. Int. J. Adv. Syst. Meas. 7, 179–192.

Franke, M., Klein, P., Schroder, L., Thoben, K.-D., 2011. Ontological semantics of standards and PLM repositories in the product development phase, in: Global Product Development. Springer, pp. 473–482. doi:10.1007/978-3-642-15973-2-48

Frisch, H.P., 2007. Model-Based Systems.

Furini, F., Rossoni, M., Colombo, G., 2016. Knowledge Based Engineering and Ontology Engineering Approaches for Product Development: Methods and Tools for Design Automation in Industrial Engineering, in: ASME 2016 International Mechanical Engineering Congress and Exposition. American Society of Mechanical Engineers, p. V011T15A032--V011T15A032. doi:10.1115/IMECE2016-67292

Genesereth, M.R., Fikes, R.E., Others, 1992. Knowledge Interchange Format - Version 3.0: Reference Manual, Logic Group: Report Logic-92-1. Computer Science Department, Stanford University, Stanford, California.

Gero, J.S., Kannengiesser, U., 2007a. A function-behavior-structure ontology of processes. Ai Edam 21, 379–391. doi:10.1017/S0890060407000340

Gero, J.S., Kannengiesser, U., 2007b. An ontology of situated design teams. Ai Edam 21, 295–308. doi:10.1017/S0890060407000297

Gero, J.S., Kannengiesser, U., 2004. The situated function–behaviour–structure framework. Des. Stud. 25, 373–391. doi:10.1016/j.destud.2003.10.010

Gingele, J., Childe, S.J., Miles, M.E., 2002. A modelling technique for re-engineering business processes controlled by ISO 9001. Comput. Ind. 49, 235–251. doi:10.1016/S0166-3615(02)00113-6

Glimm, B., Horridge, M., Parsia, B., Patel-Schneider, P., 2009. A Syntax for Rules in OWL 2, in: Proceedings of the 6th International Conference on OWL: Experiences and Directions-Volume 529. pp. 29–38.

Golbreich, C., 2004. Combining rule and ontology reasoners for the semantic web. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 3323 LNCS, 6–22. doi:10.1007/978-3-540-30504-0_2

Golbreich, C., Imai, A., 2004. Combining SWRL rules and OWL ontologies with Protégé OWL Plugin, Jess, and Racer, in: 7th International Protégé Conference, Bethesda, MD.

Golbreich, C., Wallace, E.K., Patel-Schneider, P.F., 2012. OWL 2 Web Ontology Language New Features and Rationale (Second Edition) [WWW Document]. W3C Recomm. URL https://www.w3.org/TR/2012/REC-owl2-new-features-20121211/ (accessed 1.1.16).

Goldberg, A., Robson, D., 1983. Smalltalk-80: the language and its implementation. Addison-Wesley Longman Publishing Co., Inc.

Gómez, A., Mas, F., Menéndez, J.L., Ríos, J., 2013. A knowledge based application for industrialization design. Procedia Eng. 63, 318–326. doi:10.1016/j.proeng.2013.08.178

Graves, H., 2009. Integrating SysML and OWL. CEUR Workshop Proc. 529, 117–124.

Graves, H., Horrocks, I., 2008. Application of OWL 1.1 to systems engineering, in: OWL Experiences and Directions April Workshop.

Grobshtein, Y., Dori, D., 2011. Generating SysML views from an OPM model: design and evaluation. Syst. Eng. 14, 327–340. doi:10.1002/sys

Grosof, B., Dean, M., Kifer, M., 2010. Web Rules: Fundamentals, Standards, and Applications., ISWC2010 tutorial.

Gruber, T.R., 1995. Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum. Comput. Stud. 43, 907–928. doi:citeulike-article-id:230211

Gruninger, M., 2013. Common Logic (ISO 24707), SC32 WG2 Meeting, Santa Fe, NM.

Gruninger, M., 2004. Ontology of the process specification language, in: Handbook on Ontologies. Springer Berlin Heidelberg, pp. 575–592.

Grüninger, M., 2009. Using the PSL Ontology. Handb. Ontol. 423–443. doi:10.1007/978-3-540-92673-3

Gruninger, M., Cutting-Decelle, A., 2000. ISO TC184/SC4/WG8 N225.

Gruninger, M., Katsumi, M., Mossakowski, T., 2013. Revision of ISO 24707 (Common Logic).

Grüninger, M., Menzel, C., 2003. The Process Specification Language (PSL) Theory and Applications. AI Mag. 24, 63–74. doi:10.1609/aimag.v24i3.1719

Hart, L.E., 2015. Introduction To Model-Based System Engineering (MBSE) and SysML, Delaware Valley INCOSE Chapter Meeting: Lockheed Martin Corporation.

Hartman, J., Wernecke, J., 1996. The VRML 2.0 handbook: building moving worlds on the web. Addison Wesley Longman Publishing Co., Inc.

Hay, D., 2006. Data Modeling, RDF, & OWL – Part One: An Introduction To Ontologies [WWW Document]. URL http://tdan.com/data-modeling-rdf-owl-part-one-an-introduction-to-ontologies/5025 (accessed 9.1.15).

Hayes, P., Menzel, C., 2001. A semantics for the knowledge interchange format, in: IJCAI 2001 Workshop on the IEEE Standard Upper Ontology. p. 145.

Heidari, F., Loucopoulos, P., Brazier, F., 2013. Business Process Modelling for Measuring Quality. Int. J. Adv. Intell. Syst. 6, 342–355.

Helgoson, M., Kalhori, V., 2012. A conceptual model for knowledge integration in process planning. Procedia CIRP 3, 573–578. doi:10.1016/j.procir.2012.07.098

Hennig, C., Eisenmann, H., Viehl, A., Bringmann, O., 2015. On Languages for Conceptual Data Modeling in Multi-disciplinary Space Systems Engineering, in: Model-Driven Engineering and Software Development (MODELSWARD), 2015 3rd International Conference on. IEEE, pp. 384–393.

Hennig, C., Hoppe, T., Eisenmann, H., Viehl, A., Bringmann, O., 2016. SCDML: A language for Conceptual Data Modeling in Model-based Systems Engineering. Model. Eng. Softw. Dev. (MODELSWARD), 2016 4th Int. Conf. 184–192.

Hew, K.P., Fisher, N., Awbi, H.B., 2001. Towards an integrated set of design tools based on a common data format for building and services design. Autom. Constr. 10, 459–476.

Hirtle, D., Dema, T., Boley, H., 2006. The Modularization of RuleML [WWW Document]. URL http://ruleml.org/modularization/#Model (accessed 8.1.15).

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S., 2012. OWL 2 Web Ontology Language Primer (Second Edition) [WWW Document]. W3C Recomm. URL https://www.w3.org/TR/owl2-primer/ (accessed 1.1.16).

Holt, A.W., Ramsey, H.R., Grimes, J.D., 1983. Coordination system technology as the basis for a programming environment. Electr. Commun. 57, 307–314.

Horridge, M., Knublauch, H., Rector, A., Stevens, R., Wroe, C., Jupp, S., Moulton, G.,

Drummond, N., Brandt, S., 2011. A Practical GuideTo Building OWL Ontologies Using Protege 4 and CO-ODE Tools: Edition 1.3.

Horridge, M., Patel-Schneider, P.F., 2012. OWL 2 Web Ontology Language Manchester Syntax (Second Edition) [WWW Document]. W3C Work. Gr. Note. URL https://www.w3.org/TR/2012/NOTE-owl2-manchester-syntax-20121211/ (accessed 1.1.16).

Horrocks, I., Patel-schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M., 2004. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission.

Hsu, W., Woon, I.M.Y., 1998. Current and Future Research in the Conceptual Design of Mechanical Products. Comput. Des. 30, 377–389.

Hunter, R., Vizán, A., Pérez, J., Ríos, J., 2005. Knowledge model as an integral way to reuse the knowledge for fixture design process. J. Mater. Process. Technol. 164–165, 1510–1518. doi:10.1016/j.jmatprotec.2005.02.181

INCOSE, 2007. Systems Engineering Vision 2020, INCOSE-TP-2004-004-02.

Isaksson, O., 2003. A generative modeling approach to engineering design.

ISO, 2017. Information technology -- Common Logic (CL) -- A framework for a family of logic-based languages [WWW Document]. ISO/IEC PRF 24707. URL https://www.iso.org/standard/66249.html (accessed 9.1.16).

ISO, 2015. Automation systems and integration — Object-Process Methodology, ISO/PAS 19450:2015(en).

ISO, 2012. Information technology -- Object Management Group Unified Modeling Language (OMG UML) -- Part 1: Infrastructure, ISO/IEC 19505-1:2012.

ISO, 2007. Information technology -- Common Logic (CL): a framework for a family of logic-based languages [WWW Document]. ISO/IEC 24707:2007. URL https://www.iso.org/standard/39175.html (accessed 7.1.16).

ISO, 2005. Information technology -- Open Distributed Processing -- Unified Modeling Language (UML) Version 1.4.2, ISO/IEC 19501:2005.

ISO, 2004. ISO 10303-11:2004: Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual.

Ivanov, V., Knorr, M., Leite, J., 2015. Reasoning over Ontologies and Non-monotonic Rules. Port. Conf. Artif. Intell. 388–401. doi:10.1007/978-3-319-23485-4_39

Johansson, J., 2015. Howtomation© Suite: A Novel Tool for Flexible Design Automation, in: ISPE CE. Advances in Transdisciplinary Engineering, pp. 327–336. doi:10.3233/978-1-61499-544-9-327

Johansson, J., 2011. Automated Computer Systems for Manufacturability Analyses and Tooling Design: Applied to the Rotary Draw Bending Process. Chalmers Reproservice.

Johansson, J., 2008. Design Automation Systems for Production Preparation: Applied on the Rotary Draw Bending Process. Chalmers.

Jubierre, J.R., Borrmann, A., 2015. Knowledge-based engineering for infrastructure facilities: assisted design of railway tunnels based on logic models and advanced procedural geometry dependencies. J. Inf. Technol. Constuction 20, 421–441.

Kaufmann, M., Moore, J.S., 1997. An industrial strength theorem prover for a logic based on common lisp. IEEE Trans. Softw. Eng. 23, 203–213. doi:10.1109/32.588534

Kifer, M., Boley, H., 2010. RIF Overview [WWW Document]. W3C Work. Gr. Note. URL https://www.w3.org/TR/2010/NOTE-rif-overview-20100622/ (accessed 6.1.16).

Kim, C.-H., Weston, R.H., Hodgson, a., Lee, K.-H., 2003. The complementary use of IDEF and UML modelling approaches. Comput. Ind. 50, 35–56. doi:10.1016/S0166-3615(02)00145-8

Kitamura, Y., 2006. Roles of ontologies of engineering artifacts for design knowledge modeling. Des. METHODS Pract. 21–23.

Kitamura, Y., Mizoguchi, R., 2013. Ontological characterization of functions: Perspectives for capturing functions and modeling guidelines. AI EDAM 27, 259–269. doi:10.1017/S0890060413000267

Kitamura, Y., Mizoguchi, R., 2004. Ontology-based systematization of functional knowledge. J. Eng. Des. 15, 327–351. doi:10.1080/09544820410001697163

Klein, P., Luetzenberger, J., Thoben, K.-D., 2015. A proposal for knowledge formalization in product development processes, in: DS 80-10 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 10: Design Information and Knowledge Management Milan, Italy, 27-30.07. 15.

Klein, P., Pugliese, D., Lützenberger, J., Colombo, G., Thoben, K.D., 2014. Exchange of knowledge in customized product development processes. Procedia CIRP 21, 99–104. doi:10.1016/j.procir.2014.03.149

Klyne, G., Carroll, J.J., McBride, B., 2004. Resource Description Framework (RDF): Concepts and Abstract Syntax [WWW Document]. W3C Recomm. URL http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ (accessed 5.1.15).

Knutilla, A., Schlenoff, C., Ray, S., Polyak, S.T., Tate, A., Cheah, S.C., Anderson, R.C., 1998. Process Specification Language: An Analysis of Existing Representations. Natl. Inst. Stand. Technol. (NIST), Gaithersbg. (MD), NISTIT 6160.

Kopena, J., Regli, W.C., 2003. Extensible Semantics for Representing Electromechanical Assemblies. ASME 2003 Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf. 581–590. doi:10.1115/DETC2003/CIE-48233

Kossiakoff, A., Sweet, W.N., Seymour, S., Biemer, S.M., 2011. Systems engineering principles and practice: second edition. John Wiley & Sons.

Krasner, J., 2015. How Product Development Organizations can Achieve Long- Term Cost Savings Using Model-Based Systems Engineering (MBSE).

Krima, S., Barbau, R., Fiorentini, X., Sudarsan, R., Sriram, R.D., 2009. OntoSTEP: OWL-DL ontology for STEP. Natl. Institue Stand. Technol. NISTIR 7561.

Krötzsch, M., Simancik, F., Horrocks, I., 2012. A Description Logic Primer. arXiv Prepr. arXiv1201.4089.

Kuba, M., 2012. OWL 2 and SWRL Tutorial [WWW Document]. URL http://dior.ics.muni.cz/~makub/owl/ (accessed 10.1.16).

Kulon, J., Broomhead, P., Mynors, D., 2006. Applying knowledge-based engineering to traditional manufacturing design. Int. J. Adv. Manuf. Technol. 30, 945–951. doi:10.1007/s00170-005-0067-0

Kulon, J., Mynors, D.J., Broomhead, P., 2006. A knowledge-based engineering design tool for metal forging. J. Mater. Process. Technol. 177, 331–335. doi:10.1016/j.jmatprotec.2006.04.062

La Rocca, G., 2011. Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization. TU Delft, Delft University of Technology.

La Rocca, G., Krakers, L., van Tooren, M.J.L., 2002. Development of an ICAD Generative Model for Blended Wing Body Aircraft Design. 9th AIAA/ISSMO Symp. Multidiscip. Anal. Optim. 4-6 Sept. 2002, Atlanta, Georg. 1–10.

La Rocca, G., Tooren, M. Van, 2012. Knowledge based engineering to support complex product design. Adv. Eng. Informatics 26, 157–158. doi:10.1016/j.aei.2012.02.008

La Rocca, G., van Tooren, M., 2007. A Knowledge Based Engineering Approach to Support Automatic Generation of FE Models in Aircraft Design, in: 45th AIAA Aerospace Sciences Meeting and Exhibit. doi:10.2514/6.2007-967

La Rocca, G., Van Tooren, M., 2007. Enabling distributed multi-disciplinary design of complex products: a knowledge based engineering approach. J. Des. Res. 5, 333–352. doi:10.1504/JDR.2007.014880

La Rocca, G., Van Tooren, M.J., 2010. Knowledge-Based Engineering To Support Aircraft Multidisciplinary Design and Optimisation, in: Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering. SAGE Publications, pp. 1041–1055.

Lange, C., 2013. Ontologies and languages for representing mathematical knowledge on the semantic web. Semant. Web 4, 119–158. doi:10.3233/SW-2012-0059

Lassila, O., 1990. Frames or Objects , or Both ? Abstract : 1–8.

Lee, J., Fenves, S., Bock, C., Suh, H., Rachuri, S., Fiorentini, X., Sriram, R., 2010. A semantic product modeling framework and language for behavior evaluation. NIST IR 7681. doi:10.1109/COASE.2010.5584462

Lee, J.H., Fenves, S.J., Bock, C., Suh, H.-W., Rachuri, S., Fiorentini, X., Sriram, R.D., 2010. Product modeling framework and language for behavior evaluation. 2010 IEEE Int. Conf. Autom. Sci. Eng. CASE 2010 136–143. doi:10.1109/COASE.2010.5584462

Lemaignan, S., Siadat, A., Dantan, J.-Y., Semenenko, A., 2006. MASON: A proposal for an ontology of manufacturing domain, in: Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on. IEEE, pp. 195–200. doi:10.1109/DIS.2006.48

Li, L., Qin, F., Gao, S., Qin, X., 2014. Ontology-Based Design Rationale Retrieval Supporting Natural Language Query, in: ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, p. V01BT02A017--V01BT02A017.

Li, Z., Ramani, K., 2007. Ontology-based design information extraction and retrieval. Ai Edam 21, 137–154. doi:10.1017/S0890060407070199

Li, Z., Yang, M.C., Ramani, K., 2009. A methodology for engineering ontology acquisition and validation. AI EDAM 23, 37–51. doi:10.1017/S0890060409000092

Lin, M.-C., Lin, Y.-H., Chen, M.-S., Lin, J.-Y., 2013. Development of a Parametric Form Generation Procedure for Customer-Oriented Product Design, in: 20th ISPE International Conference on Concurrent Engineering. pp. 235–243. doi:10.3233/978-1-61499-302-5-235

Lingzhi, L., Leong, A.C., Gay, R.K.L., 1996. Integration of Information Model (IDEF1) with Function Model (IDEF0) for CIM Information Systems Design, in: Expert Systems with Applications. Elsevier, pp. 373–380.

Liu, T., Xu, W., 2001. A review of web-based product data management systems. Comput. Ind. 44, 251–262. doi:10.1016/S0166-3615(01)00072-0

Liu, Y.-J., Lai, K.-L., Dai, G., Yuen, M.M.-F., 2010. A semantic feature model in concurrent engineering. IEEE Trans. Autom. Sci. Eng. 7, 659–665.

Lohith, M., Prasanna, L., Vaderahobli, D.H., 2013. Translating MOKA based Knowledge models into a Generative CAD model in CATIA V5 using Knowledgeware, in: International Conference on Modeling, Simulation and Visualization Methods(MSV). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Lu, W., Qin, Y., Qi, Q., Zeng, W., Zhong, Y., Liu, X., Jiang, X., 2016. Selecting a semantic similarity measure for concepts in two different CAD model data ontologies. Adv. Eng. Informatics 30, 449–466. doi:10.1016/j.aei.2016.06.001

Lützenberger, J., Marthinusen, I., Kristensen, K., Iversen, G., Klein, P., Sivertsen, O.I., Rutkowska, G., 2012. D6.1 Methods for KBE related knowledge acquisition and codification, Linked Knowledge in Manufacturing, Engineering and Design for Next-Generation Production. LinkedDesign Consortium. LinkedDesign Consortium.

Lützenberger, J., Marthinusen, I., Kristensen, K., Iversen, G., Klein, P., Sivertsen, O.I., Rutkowska, G., 2012. Methods for KBE related knowledge acquisition and codification, Linked Knowledge in Manufacturing, Engineering and Design for Next-Generation Production. LinkedDesign Consortium. LinkedDesign Consortium.

Lyons, K.W., Duffey, M.R., Anderson, R.C., 1995. Product Realization Process Modeling: A study of requirements, methods and research issues. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, Gaithersburg, MD.

Maier, J.F., Eckert, C.M., Clarkson, P.J., 2017. Model granularity in engineering design – concepts and framework. Des. Sci. 3. doi:10.1017/dsj.2016.16

Manola, F., Miller, E., McBride, B., 2004. RDF Primer [WWW Document]. W3C Recomm. URL http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ (accessed 5.1.15).

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., Mcllraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K., 2004. OWL-S: Semantic Markup for Web Services [WWW Document]. W3C Memb. Submiss. URL http://www.w3.org/Submission/OWL-S/ (accessed 6.1.15).

Martínez-Pellitero, S., Barreiro, J., Cuesta, E., Álvarez, B.J., 2011. A new process-based ontology for KBE system implementation: Application to inspection process planning. Int. J. Adv. Manuf. Technol. 57, 325–339. doi:10.1007/s00170-011-3285-7

Mas, F., Rios, J., Menendez, J.L., Gomez, A., Ríos, J., Menéndez, J.L., Gómez, A., 2013. A process-oriented approach to modeling the conceptual design of aircraft assembly lines. Int. J. Adv. Manuf. Technol. 67, 771–784. doi:10.1007/s00170-012-4521-5

Mayer, R.J., 1992. IDEF1 Information Modeling. A Reconstruction of the Original Air Force Wright Aeronautical Laboratory Technical Report AFWAL-TR-81-4023.

Mayer, R.J., Keen, A., Wells, M.S., 1992. Information Integration for Concurrent

Engineering (IICE) IDEF4 object-oriented design method report (No. KBSI-IICE-90-STR-01-0592-01).

Mayer, R.J., Menzel, C.P., Painter, M.K., Dewitte, P.S., Blinn, T., Perakath, B., 1995. Information integration for concurrent engineering (IICE) IDEF3 process description capture method report (No. KBSI-IICE-90-STR-01-0592-02).

Mcguinness, D.L., Van Harmelen, F., 2004. OWL Web Ontology Language Overview, W3C recommendation.

Medeiros, A.P. De, Schwabe, D., Pereira, A., Daniel, D.M., Feijó, B., 2005. Design Rationale for Model-Based Designs in Software Engineering 163–168.

Mehrpoor, M., Gulla, J.A., Ahlers, D., Kristensen, K., Ghodrat, S., Sivertsen, O.I., Marthinusen, I., Kalavrytinos, C., Sivertsen, O.I., Mehrpoor, M., Gjarde, A., Sivertsen, O.I., Shyi-Ming Chen, Ke, J., Jin-Fu Chang, Takeda, H., Takeda, H., Veerkamp, P., Veerkamp, P., Tomiyama, T., Tomiyama, T., Yoshikawa, H., Yoshikawa, H., Mehrpoor, M., Gulla, J.A., Ahlers, D., Kristensen, K., Ghodrat, S., Sivertsen, O.I., Zhao, W., Liu, J.K.K., Marthinusen, I., Kalavrytinos, C., Sivertsen, O.I., Mehrpoor, M., Gjarde, A., Sivertsen, O.I., Shyi-Ming Chen, Ke, J., Jin-Fu Chang, Takeda, H., Takeda, H., Veerkamp, P., Veerkamp, P., Tomiyama, T., Tomiyama, T., Yoshikawa, H., Yoshikawa, H., Mehrpoor, M., Gulla, J.A., Ahlers, D., Kristensen, K., Ghodrat, S., Sivertsen, O.I., Zhao, W., Liu, J.K.K., 2013. Using process ontologies to contextualize recommender systems in engineering projects for knowledge access improvement, in: Computers in Industry. Academic Conferences International Limited, p. 2013. doi:10.1016/j.compind.2008.02.002

Michel, J., 2005. Terminology extracted from some manufacturing and modelling related standards. CEN/TC 310, N1119R2.

Milton, N.R., 2007. Knowledge acquisition in practice: a step-by-step guide. Springer Science & Business Media.

Minsky, M., Horn, B., Shirai, Y., Waltz, D., Winston, P.H., 1975. The psychology of computer vision. McGraw-Hill, New York. doi:10.1016/0010-4485(73)90095-X

Mizoguchi, R., 2003. Tutorial on ontological engineering. New Gener. Comput. 21, 363–364. doi:10.1007/BF03037310

Mocan, A., Iversen, G., Rutkowska, G., Tonne, J., Cartino, S., Kristensen, K., Kadiri, S. El, Främling, K., Lützenberger, J., Klein, P., Buda, A., Peuqert, E., 2015. D10.11 The LinkedDesign Solution, Linked Knowledge in Manufacturing, Engineering and Design for Next-Generation Production. LinkedDesign Consortium. LinkedDesign Consortium.

Monfared, R.P., 2000. A component-based approach to design and construction of change capable manufacturing cell control systems. Dr. Diss. © RP Monfared.

Mordecai, Y., Orhof, O., Dori, D., 2017. Model-Based Interoperability Engineering in Systems-of-Systems and Civil Aviation. IEEE Trans. Syst. Man, Cybern. Syst. doi:10.1109/TSMC.2016.2602543

Morgenstern, L., Welty, C., Boley, H., Hallmark, G., 2012. RIF Primer (Second Edition) [WWW Document]. W3C Powered by MediaWiki. URL https://www.w3.org/2005/rules/wiki/Primer (accessed 9.1.16).

Motik, B., Patel-Schneider, P.F., Grau, B.C., Horrocks, I., Parsia, B., Sattler, U., 2012. OWL 2 Web Ontology Language Direct Semantics (Second Edition) [WWW Document].

W3C Recomm. URL https://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/ (accessed 1.1.16).

Muehlen, M. Zur, Recker, J., 2008. How much language is enough? Theoretical and practical use of the business process modeling notation. Adv. Inf. Syst. Eng. - Proc. 20th Int. Conf. Adv. Inf. Syst. Eng. 465–479. doi:10.1007/978-3-540-69534-9_35

Murata, T., 1989. Petri Nets: Properties, Analysis and Applications, in: IEEE. pp. 541–580.

Nan, J., Li, Q., 2012. Design Automation Systems – Supporting Documentation and Knowledge Management. Dr. Diss. Master Thesis, Jönköping Univ. Jönköping, Sweden. Jönköping University, Jönköping.

Natekar, D., Zhang, X., Subbarayan, G., 2004. Constructive solid analysis: A hierarchical, geometry-based meshless analysis procedure for integrated design and analysis. Comput. Des. 36, 473–486. doi:10.1016/S0010-4485(03)00129-5

Negnevitsky, M., 2005. Artificial intelligence: a guide to intelligent systems. Pearson Education.

Niles, I., Pease, A., 2001. Towards a Standard Upper Ontology, in: Proceedings of the International Conference on Formal Ontology in Information Systems-Volume 2001. ACM, pp. 2–9. doi:10.1145/505168.505170

NIST, 2008. Process Specification Language (PSL) - A few PSL Basics [WWW Document]. NIST Natl. Inst. Stand. Technol. URL http://www.mel.nist.gov/psl/ (accessed 3.1.15).

NIST, 2007. Process Specification Language (PSL) - PSL Ontology -- Current Theories and Extensions (version 2.8) [WWW Document]. NIST Natl. Inst. Stand. Technol. URL http://www.mel.nist.gov/psl/ontology.html (accessed 10.1.15).

NIST, 2004. PSL Core [WWW Document]. NIST Natl. Inst. Stand. Technol. URL http://www.mel.nist.gov/psl/psl-ontology/pslcore_page.html (accessed 1.1.16).

Noel, M.F., 2006. A dynamic multi-view product model to share product behaviours among designers : how process model adds semantic to the behaviour paradigm. Int. J. Prod. lifecycle Manag. 1, 380–390.

Noh, J.-D., Suh, H.-W., 2008. Layered Product Knowledge Representation and Reasoning with OWL & SWRL, in: ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference IDETC/CIE 2008. American Society of Mechanical Engineers, pp. 607–616.

Nomaguchi, Y., Fujita, K., 2013. Knowledge representation framework for interactive capture and management of reflection process in product concepts development. Adv. Eng. Informatics 27, 537–554. doi:10.1016/j.aei.2013.06.004

Nomaguchi, Y., Shimomura, Y., Tomiyama, T., 2004. Management of design knowledge for knowledge-based CAD. TMCE. Lausanne, Switz. 1–9.

Nomaguchi, Y., Yoshioka, M., Tomiyama, T., 2002. Document-based Design Process Knowledge Management for Knowledge Intensive Engineering. From Knowl. Intensive CAD to Knowl. Intensive Eng. 131–144.

Noy, N.F., McGuinness, D.L., 2001. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowl. Syst. Lab. 25. doi:10.1016/j.artmed.2004.01.014

O'Donovan, B., Clarkson, P., Eckert, C., 2003. Signposting: Modelling uncertainty in design processes, in: DS 31: Proceedings of ICED 03, the 14th International Conference on

Engineering Design, Stockholm. Stockholm.

Obitko, M., 2007a. Frame Based Models [WWW Document]. Ontol. Semant. Web. URL http://www.obitko.com/tutorials/ontologies-semantic-web/frame-based-models.html (accessed 9.1.15).

Obitko, M., 2007b. Semantic Networks [WWW Document]. Ontol. Semant. Web. URL http://www.obitko.com/tutorials/ontologies-semantic-web/semantic-networks.html (accessed 8.1.15).

Obitko, M., 2007c. Translation to FOPL [WWW Document]. Ontol. Semant. Web. URL http://www.obitko.com/tutorials/ontologies-semantic-web/translation-to-fopl.html (accessed 10.1.15).

Obitko, M., 2007d. Description Logics [WWW Document]. Ontol. Semant. Web. URL http://www.obitko.com/tutorials/ontologies-semantic-web/description-logics.html (accessed 9.1.15).

Obitko, M., 2007e. Web Ontology Language OWL [WWW Document]. Ontol. Semant. Web. URL http://www.obitko.com/tutorials/ontologies-semantic-web/web-ontology-language-owl.html (accessed 8.1.15).

Obitko, M., 2007f. Conceptual Graphs [WWW Document]. Ontol. Semant. Web. URL http://www.obitko.com/tutorials/ontologies-semantic-web/conceptual-graphs.html (accessed 11.1.15).

Obitko, M., 2007g. Common Logic [WWW Document]. Ontol. Semant. Web. URL http://www.obitko.com/tutorials/ontologies-semantic-web/common-logic.html (accessed 7.1.15).

Obitko, M., 2007h. Knowledge Interchange Format [WWW Document]. Ontol. Semant. Web. URL http://www.obitko.com/tutorials/ontologies-semantic-web/knowledge-interchange-format.html (accessed 8.1.15).

Olivetti, N., 2011. Introduction to Non Monotonic Reasoning.

OMG, 2016. Unified Modeling Language™ (UML®), Object Management Group.

OMG, 2013. Requirements Interchange Format (ReqIF) Version 1.1, Object Management Group (OMG).

Ovtcharova, J.G., 2010. Virtual Engineering: Principles , Methods and Applications, in: DS 60: Proceedings of DESIGN 2010, the 11th International Design Conference, Dubrovnik, Croatia.

Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H.H., 2007. Engineering Design: A Systematic Approach, Third Edition (Translators and Editors: Ken Wallace and Luciënne T. M. Blessing). Springer-Verlag London. doi:10.1007/978-1-84628-319-2

Panetto, H., Dassisti, M., Tursi, A., 2012. ONTO-PDM: Product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment. Adv. Eng. Informatics 26, 334–348. doi:10.1016/j.aei.2011.12.002

Pardalis, K., Kadiri, S. El, 2014. D3.3 The LinkedDesign Ontology, Linked Knowledge in Manufacturing, Engineering and Design for Next-Generation Production. LinkedDesign Consortium. LinkedDesign Consortium.

Patil, L., Dutta, D., Sriram, R., 2005. Ontology-based exchange of product data semantics. IEEE Trans. Autom. Sci. Eng. 2, 213–224. doi:10.1109/TASE.2005.849087

Patil, L.M., 2005. Interoperability of formal semantics of product data across product development systems. University of Michigan.

Peak, R.S., Lubell, J., Srinivasan, V., Waterbury, S.C., 2004. STEP, XML, and UML: Complementary Technologies. J. Comput. Inf. Sci. Eng. 4, 379–390. doi:10.1115/1.1818683

Pease, A., 1998. Core plan representation, Object Model Focus Group.

Peleg, M., Dori, D., 1999. Extending the object-process methodology to handle real-time systems. JOOP 11, 53–58.

Peng, G., Wang, H., Zhang, H., Zhao, Y., Johnson, A.L., 2017. A collaborative system for capturing and reusing in-context design knowledge with an integrated representation model. Adv. Eng. Informatics 33, 314–329. doi:10.1016/j.aei.2016.12.007

Penoyer, J., Burnett, G., Fawcett, D., Liou, S.-Y., 2000. Knowledge based product life cycle systems: principles of integration of KBE and C3P. Comput. Des. 32, 311–320. doi:10.1016/S0010-4485(00)00014-2

Perales, F., de la Maza, S., 2015. D10.12 LinkedDesign Demonstrators Evaluation, Linked Knowledge in Manufacturing, Engineering and Design for Next-Generation Production. LinkedDesign Consortium. LinkedDesign Consortium.

Pereira, C.A., Correa, A., Yogui, R., de Lima, C.M., 2011. Interoperability Among Engineering Systems and Their Relevance to the Effectiveness of the Engineering Project's Life Cycle--Regulation, Well Succeeded Examples and Proposed Actions. 21st Brazilian Congr. Mech. Eng. (COBEM), Natal, Brazil 24–28.

Pinfold, M., Chapman, C., 2001. Application of KBE techniques to the FE model creation of an automotive body structure. Comput. Ind. 44, 1–10. doi:10.1016/S0166-3615(00)00079-8

Pinfold, M., Chapman, C., Preston, S., 2008. Knowledge acquisition and documentation for the development of a KBE system for automated FE analysis. Int. J. Knowl. Manag. Stud. 2, 163–174. doi:10.1504/IJKMS.2008.018319

Pinto, H.S., Martins, J.P., 2004. Ontologies: How can They be Built? Knowl. Inf. Syst. 6, 441–464. doi:10.1007/s10115-003-0138-1

Plaia, A., Carrie, A., 1995. Application and assessment of IDEF3-process flow description capture method. Int. J. Oper. Prod. Manag. 15, 63–73. doi:10.1108/01443579510077214

Plateaux, R., Choley, J.Y., Penas, O., Riviere, A., 2009. Towards an integrated mechatronic design process, in: Mechatronics, 2009. ICM 2009. IEEE International Conference On. IEEE. pp. 1–6.

Polleres, A., Boley, H., Kifer, M., 2013. RIF Datatypes and Built-Ins 1.0 (Second Edition) [WWW Document]. W3C Recomm. URL https://www.w3.org/TR/rif-dtb/ (accessed 9.1.16).

Poole, D., Mackworth, A., 2010. Non-monotonic Reasoning [WWW Document]. Artif. Intell. Found. Comput. Agents. URL http://artint.info/html/ArtInt_129.html (accessed 1.1.16).

Pooley, R., King, P., 1999. The unified modelling language and performance engineering, in: Software, IEE Proceedings-, Vol. 146, No. 1. IET. pp. 2–10.

Poorkiany, M., Johansson, J., Elgh, F., 2016. Capturing, structuring and accessing design rationale in integrated product design and manufacturing processes. Adv. Eng.

Informatics 30, 522–536. doi:10.1016/j.aei.2016.06.004

Pouchard, L., Ivezic, N., Schlenoff, C., 2000. Ontology engineering for distributed collaboration in manufacturing, in: Proceedings of the AIS2000 Conference.

Pouchard, L.C., Cutting-Decelle, A.F., Michel, J.J., Grüninger, M., 2005. ISO 18629 PSL: A Standardised language for specifying and exchanging process information. IFAC Proc. Vol. 38, 37–45.

Prasad, B., 2006. Best Practices in Knowledge-based Engineering - Catia Operators Exchange (COE) Report, ResearchGate. doi:10.13140/2.1.4370.5602

Prasad, B., 2005. What distinguishes KBE from automation, COE NewsNet.

Pratt, M.J., 2001. Introduction to ISO 10303 - the STEP Standard for Product Data Exchange. J. Comput. Inf. Sci. Eng. 1, 102–103.

Preston, S.T., Chapman, C.B., Pinfold, M., 2004. Process Integration and Design Exploration.

Prijic, A., Chapman, C., Burton, P., 2005. Knowledge Based Engineering (KBE) Past, present and Future, in: Beograd 2005 EAEC European Automotive Congress. Coventry, UK.

Pugh, S., 1991. Total design: integrated methods for successful product engineering. 1990.

Pugliese, D., Colombo, G., 2014. D6.3 "Rule Interchange Format " standardisation document, Linked Knowledge in Manufacturing, Engineering and Design for Next-Generation Production. LinkedDesign Consortium. LinkedDesign Consortium.

Qiao, L., Kao, S., Zhang, Y., 2011. Manufacturing process modelling using process specification language. Int. J. Adv. Manuf. Technol. 55, 549–563. doi:10.1007/s00170-010-3115-3

Qin, F., Gao, S., Yang, X., Li, M., Bai, J., 2016. An ontology-based semantic retrieval approach for heterogeneous 3D CAD models. Adv. Eng. Informatics 30, 751–768. doi:10.1016/j.aei.2016.10.001

Qin, S.F., Harrison, R., West, A.A., Jordanov, I.N., Wright, D.K., 2003. A framework of web-based conceptual design. Comput. Ind. 50, 153–164. doi:10.1016/S0166-3615(02)00117-3

Qin, Y., Lu, W., Qi, Q., Liu, X., Zhong, Y., Scott, P.J., Jiang, X., 2017. Status, Comparison, and Issues of Computer-Aided Design Model Data Exchange Methods Based on Standardized Neutral Files and Web Ontology Language File. J. Comput. Inf. Sci. Eng. 17, 10801. doi:10.1115/1.4034325

Rachuri, S., Baysal, M., Roy, U., Foufou, S., Bock, C., Fenves, S., Subrahmanian, E., Lyons, K., Sriram, R., 2005. Information models for product representation: core and assembly models. Int. J. Prod. Dev. 2, 207–235.

Rachuri, S., Han, Y.-H., Foufou, S., Feng, S.C., Roy, U., Wang, F., Sriram, R.D., Lyons, K.W., 2006. A Model for Capturing Product Assembly Information. J. Comput. Inf. Sci. Eng. 6, 11–21. doi:10.1115/1.2164451

Ray, S.R., Jones, A.T., 2006. Manufacturing interoperability. J. Intell. Manuf. 17, 681–688. doi:10.1007/s10845-006-0037-x

Reddy, E.J., Sridhar, C.N. V., Rangadu, V.P., 2015. Knowledge Based Engineering: Notion, Approaches and Future Trends. Am. J. Intell. Syst. 5, 1–17. doi:10.5923/j.ajis.20150501.01

Reeker, L.H., 1994. Tools for Representing and Managing Knowledge : Some Practical Requirements and Suggestions, in: Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on. IEEE, pp. 240–244. doi:10.1109/TAI.1994.346485

Regli, W.C., Hu, X., Atwood, M., Sun, W., 2000. A Survey of Design Rationale Systems: Approaches, Representation, Capture and Retrieval. Eng. Comput. 16, 209–235. doi:10.1007/PL00013715

Reijnders, A.W., 2012. Integrating Knowledge Management and Knowledge-Based Engineering. Delft University of Technology.

Reilly, D., 2006. Inside Java : The Java Programming Language [WWW Document]. Insid. Java. URL http://www.javacoffeebreak.com/articles/inside_java/insidejava-nov99.html (accessed 12.1.16).

Reinhartz-Berger, I., Dori, D., 2004. Object-Process Methodology (OPM) vs. UML-a Code Generation Perspective. CAiSE Work. 275–286.

Rezayat, M., 2000. Knowledge-based product development using XML and KCs. Comput. Des. 32, 299–309. doi:10.1016/S0010-4485(00)00013-0

Rihoux, B., Ragin, C.C., 2009. Configurational comparative methods. Qualitative Comparative Analysis (QCA) and related techniques (Applied Social Research Methods). Thousand Oaks and London: Sage.

Ríos, J., Jiménez, J. V, Pérez, J., Vizán, A., Menéndez, J.L., Más, F., 2005. KBE Application for the Design and Manufacture of HSM Fixtures. Acta Polytech. 45.

Robin, 2013. Category: "Knowledge Representation" [WWW Document]. Artif. Intell. URL http://intelligence.worldofcomputing.net/category/knowledge-representation# (accessed 8.1.15).

Rocca, G. La, 2012. Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. Adv. Eng. Informatics 26, 159–179. doi:10.1016/j.aei.2012.02.002

Roy, U., Pramanik, N., Sudarsan, R., Sriram, R.D., Lyons, K.W., 2001. Function-to-form mapping: Model, representation and applications in design synthesis. CAD Comput. Aided Des. 33, 699–719. doi:10.1016/S0010-4485(00)00100-7

Sainter, P., Oldham, K., Larkin, A., 2000. Achieving benefits from knowledge-based engineering systems in the longer term as well as in the short term., in: Proceedings of: 6th International Conference on Concurrent Enterprising. Citeseer.

Sainter, P., Oldham, K., Larkin, A., Murton, A., Brimble, R., 2000. Product knowledge management within knowledge-based engineering systems, in: Design Engineering Technical Conference, Baltimore, Setembro. Baltimore.

Sandberg, M., 2003. Knowledge based engineering-in product development, Lulea University of Technology, Sweden. Department of Applied Physics and Mechanical Engineering, Division of Computer Aided Design, Lulea University of Technology, Sweden.

Sandberg, M., Tyapin, I., Kokkolaras, M., Lundbladh, A., Isaksson, O., 2017. A knowledge-based master model approach exemplified with jet engine structural design. Comput. Ind. 85, 31–38. doi:10.1016/j.compind.2016.12.003

Sanya, I., Shehab, E., Lowe, D., Maksimovic, M., Al-Ashaab, A., 2011. Towards a Semantic Knowledge Life Cycle Approach for Aerospace Design Engineering. Improv. Complex

Syst. Today 285–292.

Sanya, I.O., Shehab, E.M., 2015. A framework for developing engineering design ontologies within the aerospace industry. Int. J. Prod. Res. 53, 2383–2409. doi:10.1080/00207543.2014.965352

Sanya, I.O., Shehab, E.M., 2014. An ontology framework for developing platform-independent knowledge-based engineering systems in the aerospace industry. Int. J. Prod. Res. 52, 6192–6215. doi:10.1080/00207543.2014.919422

Sarigecili, M.I., Roy, U., Rachuri, S., 2014. Interpreting the semantics of GD&T specifications of a product for tolerance analysis. Comput. Des. 47, 72–84. doi:10.1016/j.cad.2013.09.002

Scheuerlein, H., Rauchfuss, F., Dittmar, Y., Molle, R., Lehmann, T., Pienkos, N., Settmacher, U., 2012. New methods for clinical pathways - Business Process Modeling Notation (BPMN) and Tangible Business Process Modeling (t.BPM). Langenbeck's Arch. Surg. 397, 755–761. doi:10.1007/s00423-012-0914-z

Schlenoff, C., Ciocoiu, M., Libes, D., Gruninger, M., 1999. Process Specification Language (PSL): results of the first pilot implementation, in: Proceedings of IMECE: International Mechanical Engineering Congress and Exposition. pp. 1–10.

Schlenoff, C., Gruninger, M., Ciocoiu, M., Lee, J., 2000a. The Essence of the Process Specification Language. Trans. Soc. Comput. Simul. Int. 16, 1–43.

Schlenoff, C., Gruninger, M., Tissot, F., Valois, J., Lubell, J., Lee, J., 2000b. The Process Specification Language (PSL) Overview and Version 1.0 Specification. NISTIR.

Schreiber, G., Akkermans, H., Anjewierden, A., Hoog, R. De, Shadbolt, N., Velde, W. Van De, Wielinga, B., 2000. Knowledge engineering and management: The CommonKADS Methodology. MIT Press, Cambridge, MA.

Section 6.0 Author's Guide to Creating IDEF1 Diagrams, n.d. . Texas.

Sharma, D.K., Hitesh, Rao, V., 2014. Configurable business process modeling notation. Souvenir 2014 IEEE Int. Adv. Comput. Conf. IACC 2014 1424–1429. doi:10.1109/IAdCC.2014.6779535

Shehab, E., Abdalla, H.S., 2002. An intelligent knowledge-based system for product cost modelling. Int. J. Adv. Manuf. Technol. 19, 49–65. doi:10.1007/PL00003967

Shehab, E.M., Abdalla, H.S., 2006. A Cost-Effective Knowledge-Based Reasoning System for Design for Automation. Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. 220, 729–743. doi:10.1243/095440554JEM298

Shintre, N., Shakir, A., 2011. Knowledge based engineering across Product Realization.

Shukla, N., Keast, J.E., Ceglarek, D., 2014. Improved workflow modelling using role activity diagram-based modelling with application to a radiology service case study. Comput. Methods Programs Biomed. 116, 274–98. doi:10.1016/j.cmpb.2014.05.005

Shyamsundar, N., Gadh, R., 2002. Collaborative virtual prototyping of product assemblies over the Internet. Comput. Aided Des. 34, 755–768. doi:10.1016/S0010-4485(01)00204-4

Shyamsundar, N., Gadh, R., 2001. Internet-based collaborative product design with assembly features and virtual design spaces. CAD Comput. Aided Des. 33, 637–651. doi:10.1016/S0010-4485(01)00069-0

Siricharoen, W. V, 2007. Ontologies and Object models in Object Oriented Software Engineering. IAENG Int. J. Comput. Sci. 33, 19–24.

Skarka, W., 2007. Application of MOKA methodology in generative model creation using CATIA. Eng. Appl. Artif. Intell. 20, 677–690. doi:10.1016/j.engappai.2006.11.019

Smith, R.P., Eppinger, S.D., 1997. Identifying Controlling Features of Engineering Design Iteration. Manage. Sci. 43, 276–293.

Smith, R.P., Morrow, J.A., 1999. Product development process modeling. Des. Stud. 20, 237–261.

Sorli, M., Maksimovic, M., Al-Ashaab, A., Sulowski, R., Shehab, E., Sopelana, A., 2012. Development of KBE system to support LeanPPD application, in: Engineering, Technology and Innovation (ICE), 2012 18th International ICE Conference on. IEEE, pp. 1–8. doi:10.1109/ICE.2012.6297671

Sowa, J.F., 2015. Semantic Networks. doi:10.3115/1118735.1118739

Sowa, J.F., 2011. Introduction to Common Logic [WWW Document]. URL http://www.jfsowa.com/talks/clintro.pdf (accessed 4.1.15).

Sowa, J.F., 2008a. Conceptual Graphs. Found. Artif. Intell. Handb. Knowl. Represent. 3, 213−237. doi:10.1016/0950-7051(92)90028-E

Sowa, J.F., 2008b. Common Logic: A Framework for a Family of Logic-Based Languages.

Sowa, J.F., 2007. Fads and fallacies about logic. IEEE Intell. Syst. 22, 84–87. doi:10.1109/MIS.2007.29

Stacey, M., Clarkson, P.J., Eckert, C., 2000. Signposting: An AI approach to supporting human decision making in design. DETC ’00 ASME 2000 Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf. 1–20.

Stark, J., 2011. Product Lifecycle Management: 21st Century Paradigm for Product Realisation (2nd edition). Springer-Verlag London.

Stokes, M., 2001. Managing Engineering Knowledge: MOKA: Methodology for Knowledge Based Engineering Applications, MOKA Consortium. Professional Engineering Publ.

Subahi, A.F., 2015. A Business User Model-Driven Engineering Method for Developing Information Systems. Doctoral dissertation, University of Sheffield.

Sudarsan, R., Fenves, S.J., Sriram, R.D., Wang, F., 2005. A product information modeling framework for product lifecycle management. Comput. Des. 37, 1399–1411. doi:10.1016/j.cad.2005.02.010

SWRL Section 8. Built-Ins [WWW Document], 2009. URL http://www.daml.org/swrl/proposal/builtins.html (accessed 6.1.16).

Szykman, S., Fenves, S.J., Keirouz, W., Shooter, S.B., 2001. A foundation for interoperability in next-generation product development systems. Comput. Des. 33, 545–559. doi:10.1016/S0010-4485(01)00053-7

Szykman, S., Sriram, R.D., Bochenek, C., Racz, J.W., Senfaute, J., 2000a. Design repositories: engineering design’s new knowledge base. IEEE Intell. Syst. Their Appl. 15, 48–55.

Szykman, S., Sriram, R.D., Bochenek, C., Racz, J.W., Senfaute, J., 2000b. Design repositories: next-generation engineering design databases. IEEE Intell. Syst. 15, 48–55.

Tang, D., Zheng, L., Li, Z., Chin, K.S., 2001. STEP-based product modeling for concurrent stamped part and die development. Comput. Ind. 46, 75–94. doi:10.1016/S0166-3615(01)00116-6

TechnoSoft Inc, 2003. The Adaptive Modeling Language. A Technical Perspective.

Terpenny, J.P., Strong, S., Wang, J., 2000. A methodology for knowledge discovery and classification, in: Tenth FAIM 2000 - Flexible Automation and Intelligent Manufacturing Conference.

Tessier, S., Wang, Y., 2013. Ontology-based feature mapping and verification between CAD systems. Adv. Eng. Informatics 27, 76–92. doi:10.1016/j.aei.2012.11.008

Tomiyama, T., Hew, K.P., 2000. Knowledge Intensive Computer Aided Design: Past, Present and Future, in: Knowledge Intensive Computer Aided Design. Springer, pp. 3–18.

Tomiyama, T., Van Beek, T.J., Cabrera, A.A.A., Komoto, H., D'Amelio, V., 2013. Making function modeling practically usable. Ai Edam 27, 301–309. doi:10.1017/S0890060413000309

Tomiyama, T., Yoshioka, M., Tsumaya, A., 2002. A knowledge operation model of synthesis, in: Engineering Design Synthesis. Springer London, pp. 67–90.

Tor, S., Lee, S., Britton, G., Zhang, W., 2008. Knowledge-based functional design of industrial robots. Int. J. Prod. Res. 46, 4501–4519.

Toussaint, J., Cheng, K., 2002. Design agility and manufacturing responsiveness on the Web. Integr. Manuf. Syst. 13, 328–339. doi:10.1108/09576060210429784

Tyapin, I., Sandberg, M., Kokkolaras, M., Lundbladh, A., Isaksson, O., 2012. Jet Engine Design Optimization Using a Knowledge-Based Master Model, in: ASME Turbo Expo 2012: Turbine Technical Conference and Exposition. American Society of Mechanical Engineers, pp. 41–47.

Ullman, D.G., 2010. The Mechanical Design Process - Fourth Edition. McGraw-Hill Science/Engineering/Math.

Ullman, D.G., 2002. Toward the ideal mechanical engineering design support system. Res. Eng. Des. 13, 55–64. doi:10.1007/S00163-001-0007-4

Ulrich, K.T., Eppinger, S.D., 2012. Product Design and Development, Fifth Edition. McGraw-Hill. doi:10.1016/B978-0-7506-8985-4.00002-4

Umeda, Y., Tomiyama, T., 1997. Functional reasoning in design. IEEE Expert 12, 42–48. doi:10.1109/64.585103

Usman, Z., 2012. A Manufacturing Core Concepts Ontology to Support Knowledge Sharing. (Doctoral Diss. © Zahid Usman).

Usman, Z., Young, R.I.M., Chungoora, N., Palmer, C., Case, K., Harding, J., 2011. A manufacturing core concepts ontology for product lifecycle interoperability, in: International IFIP Working Conference on Enterprise Interoperability. Springer Berlin Heidelberg, pp. 5–18. doi:10.1007/978-1-84996-257-5_14

Usman, Z., Young, R.I.M., Chungoora, N., Palmer, C., Case, K., Harding, J.A.J., 2013. Towards a formal manufacturing reference ontology. Int. J. Prod. Res. 51, 6553–6572. doi:10.1080/00207543.2013.801570

Van der Velden, C.A., 2008. Application of Knowledge Based Engineering Principles to Intelligent Automation Systems.

Van Der Velden, C., Bil, C., Xu, X., 2012. Adaptable methodology for automation application development. Adv. Eng. Informatics 26, 231–250. doi:10.1016/j.aei.2012.02.007

Van Renssen, A., 2003. Gellish: an information representation language, knowledge base and ontology, in: Standardization and Innovation in Information Technology, 2003. The 3rd Conference on. IEEE, pp. 215–228.

Van Renssen, A.S.H.P., 2005. Gellish: a generic extensible ontological language-design and application of a universal data structure. Delft University Press. doi:http://dx.doi.org/10.1108/17506200710779521

Van Tooren, M., La Rocca, G., Krakers, L., Beukers, A., 2003. Design and technology in aerospace. Parametric modeling of complex structure systems including active components, in: 13th International Conference on Composite Materials. S. Diego.

Vanderperren, Y., Mueller, W., Dehaene, W., 2008. UML for electronic systems design: a comprehensive overview. Des. Autom. Embed. Syst. 12, 261–292.

Vaziri, M., Jackson, D., 2000. Some Shortcomings of OCL, the Object Constraint Language of UML, in: TOOLS (34). pp. 555–562.

Verhagen, W.J., Curran, R., 2010. Knowledge-based engineering review: conceptual foundations and research issues, in: New World Situation: New Directions in Concurrent Engineering. Springer London, pp. 267–276.

Verhagen, W.J.C., Bermell-Garcia, P., Van Dijk, R.E.C., Curran, R., 2012. A critical review of Knowledge-Based Engineering: An identification of research challenges. Adv. Eng. Informatics 26, 5–15. doi:10.1016/j.aei.2011.06.004

Vernadat, F., 2002. UEML : towards a unified enterprise modelling language. Int. J. Prod. Res. 40, 4309–4321.

Viola, N., Corpino, S., Fioriti, M., Stesina, F., 2012. Functional analysis in systems engineering: methodology and applications, in: Systems Engineering-Practice and Theory. InTech.

W3C, 2016. What is MathML? [WWW Document]. W3C Math Home. URL https://www.w3.org/Math/ (accessed 10.1.15).

W3C, 2012. OWL 2 Web Ontology Language Document Overview (Second Edition) [WWW Document]. W3C Recomm. URL http://www.w3.org/TR/owl2-overview/ (accessed 6.1.15).

Wagner, W.P., Chung, Q.B., Najdawi, M.K., 2003. The impact of problem domains and knowledge acquisition techniques: a content analysis of P/OM expert system case studies. Expert Syst. Appl. 24, 79–86. doi:10.1016/S0957-4174(02)00085-4

Wagner, W.P., Najdawi, M.K., Chung, Q.Q.., 2001. Selection of knowledge acquisition techniques based upon the problem domain characteristics of production and operations management expert systems. Expert Syst. 18, 76–87.

Wang, H.H., Noy, N., Rector, A., Musen, M., Redmond, T., Rubin, D., Tu, S., Tudorache, T., Drummond, N., Horridge, M., Others, 2006. Frames and OWL side by side, in: Presentation Abstracts. Citeseer, p. 54. doi:10.1080/j.1440-1614.2006.01852.x

Wang, L., Shen, W., Xie, H., Neelamkavil, J., Pardasani, A., 2002. Collaborative conceptual design—state of the art and future trends. Comput. Des. 34, 981–996. doi:10.1016/S0010-4485(01)00157-9

Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K., 2004. Ontology based context modeling and reasoning using OWL, in: Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on. IEEE, pp. 18–22. doi:10.1109/PERCOMW.2004.1276898

Web3D, 2017. X3D & VRML, The Most Widely Used 3D Formats [WWW Document]. Web3D Consort. URL http://www.web3d.org/x3d-vrml-most-widely-used-3d-formats (accessed 10.1.16).

Weilkiens, T., 2007. Systems Engineering with SysML/UML-Modeling, Analysis, Design, Systems Engineering with SysML/UML. Morgan Kaufmann, OMG press, Burlington. doi:10.1016/B978-0-12-374274-2.00001-8

Wenzel, H., Gondhalekar, A., Balachandran, L., Guenov, M., Nunez, M., 2011. Automated generation of Isight-Models through a neutral workflow description, in: 2011 SIMULIA Customer Conference. Barcelona.

Wenzel, H., Nunez, M., Gondhalekar, A.C., Guenov, M.D., Balachandran, L.K., 2011. Neutral Description and Exchange of Design Computational Workflows, in: 18th International Conference on Engineering Design, ICED11, Impacting Society through Engineering Design, Vol 1: Design Processes, 15-19.08.2011. Lyngby/Copenhagen, Denmark.

Witherell, P., Krishnamurty, S., Grosse, I.R., 2007. Ontologies for supporting engineering design optimization. J. Comput. Inf. Sci. Eng. 7, 141–150. doi:10.1115/1.2720882

Woestenenk, K., Bonnema, G.M., Alvarez Cabrera, A.A., Tomiyama, T., 2011. Capturing design process information in complex product development, in: ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2011 August 29-31, 2011, Washington, DC, USA. ASME.

Woestenenk, K., Tragter, H., Bonnema, G.M., Cabrera, A.A.A.A., Tomiyama, T., 2010. Multi domain design: integration and reuse, in: ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, pp. 519–528.

Wuest, T., Liu, A., Wei, W., Lu, S.C.Y.Y., Thoben, K.D., 2015. Utilization of state drivers to support design for manufacturing. Procedia CIRP 36, 72–77. doi:10.1016/j.procir.2015.01.081

Wynn, D.C., Clarkson, P.J., 2017. Process models in design and development. Res. Eng. Des. 1–42. doi:10.1007/s00163-017-0262-7

Wynn, D.C., Wyatt, D.F., Nair, S.M.T., Clarkson, P.J., 2010. An Introduction to the Cambridge Advanced Modeller 19–20.

Yahia, N., Mokhtar, S.A., Ahmed, A., 2012. Automatic Generation of OWL Ontology from XML Data Source. arXiv Prepr. arXiv1206.0570.

Young, R., Gunendran, A., Cutting-Decelle, A., Gruninger, M., 2007. Manufacturing knowledge sharing in PLM: a progression towards the use of heavy weight ontologies. Int. J. Prod. Res. 45, 1505–1519. doi:10.1080/00207540600942268

Zeng, J., Chen, W., Ding, Q., 2003. A Web-based CAD system. J. Mater. Process. Technol. 139, 229–232. doi:10.1016/S0924-0136(03)00225-5

Zeng, Y., Gu, P., 1999. A science-based approach to product design theory Part I:

Formulation and formalization of design process. Robot. Comput. Integr. Manuf. 15, 331–339. doi:10.1016/S0736-5845(99)00029-0

Zha, X.., Du, H., 2002. A PDES/STEP-based model and system for concurrent integrated design and assembly planning. Comput. Des. 34, 1087–1110.

Zhan, P., Jayaram, U., Kim, O., Zhu, L., 2010. Knowledge Representation and Ontology Mapping Methods for Product Data in Engineering Applications. J. Comput. Inf. Sci. Eng. 10, 21004. doi:10.1115/1.3330432

Zhang, H., Wang, H., Chen, D., Zacharewicz, G., 2010. A model-driven approach to multidisciplinary collaborative simulation for virtual product development. Adv. Eng. Informatics 24, 167–179. doi:10.1016/j.aei.2009.07.005

Zhang, Q., Deniaud, I., Baron, C., Caillaud, E., 2013. Proposal of an Activity-Based Adaptive Process Model for Innovative Design, in: ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, p. V005T06A032--V005T06A032.

Zhang, S., Wang, G., Zhang, L., Fang, X., 2009. CNC programming system for complex components based on KBE within integrated environment of CAD/CAPP/CAM. Front. Mech. Eng. China 4, 97–102. doi:10.1007/s11465-009-0007-z

Zhang, Y., Luo, X., Zhao, Y., Zhang, H.C., 2015. An ontology-based knowledge framework for engineering material selection. Adv. Eng. Informatics 29, 985–1000. doi:10.1016/j.aei.2015.09.002

Zhao, W., Liu, J., 2008a. OWL/SWRL representation methodology for EXPRESS-driven product information model: Part I. Implementation methodology. Comput. Ind. 59, 580–589. doi:10.1016/j.compind.2008.02.002

Zhao, W., Liu, J., 2008b. OWL/SWRL representation methodology for EXPRESS-driven product information model: Part II: Practice. Comput. Ind. 59, 590–600. doi:10.1016/j.compind.2008.02.002

Zhu, L., Jayaram, U., Jayaram, S., Kim, O., 2009. Ontology-driven integration of CAD/CAE Applications: Strategies and comparisons, in: 2009 ASME IDETC/CIE Conference. American Society of Mechanical Engineers.

Appendix 1: Ontology Development Methodology

A. Introduction

Ontology is a formal explicitdescription of concepts in a domain of discourse (classes (referred as concepts)),properties of each concept describing various features and attributes of the concept (slots(referred as roles or properties)), and restrictions on slots (facets (referred as role restrictions)). Ontology together with a set of individual instances of classesconstitutes a knowledge base(Noy and McGuinness, 2001). Ontologies have been used in engineering applications as part of artificial intelligence and can be used for various purposes such as those of CAD systems, PLM systems and KBE applications along with adopted as part of model driven approach for interoperability. They have been used for product and process model and structure, design automation, requirements engineering, manufacturing and production processes for exchange of knowledge and automation (El Kadiri et al., 2015; El Kadiri and Kiritsis, 2015).

B. Steps adopted to create an Ontology for Design Engineering Automation

In order to create an ontology to address Design Engineering Automation (DEA) with inclusion of manufacturing knowledge, high-level ontology development methodology has been adopted from (Noy and McGuinness, 2001) as shown in Figure A 1. Specifically catering to engineering design domain with manufacturing knowledge for optimisation, ontology development methodology has been also adopted from (Ahmed et al., 2007; Witherell et al., 2007) as shown in Figure A 2.

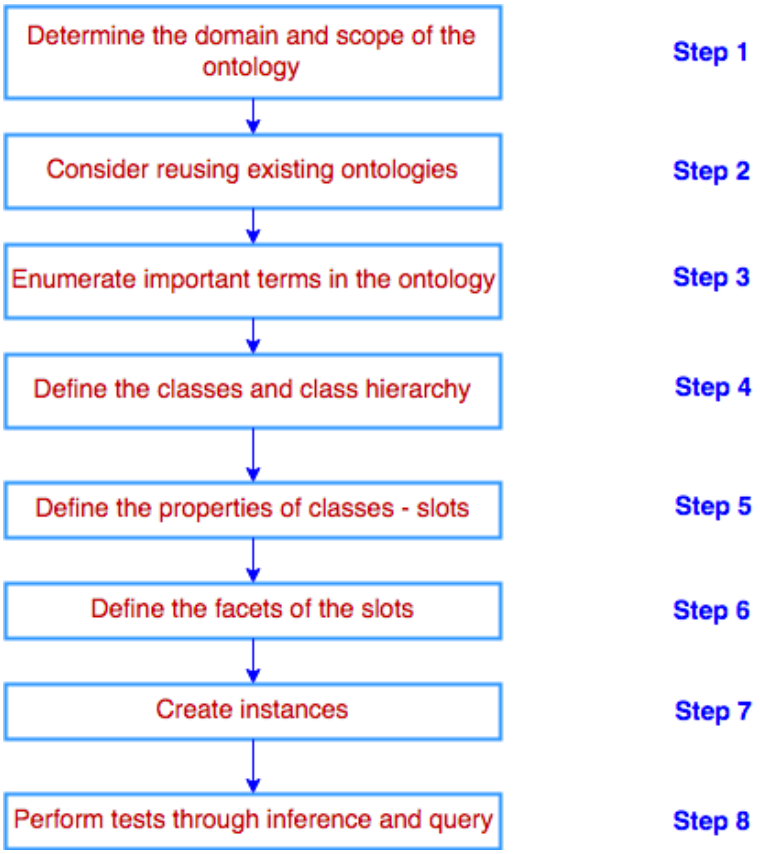


Figure A 1: Ontology Development Methodology [Adopted from (Noy and McGuinness, 2001)]

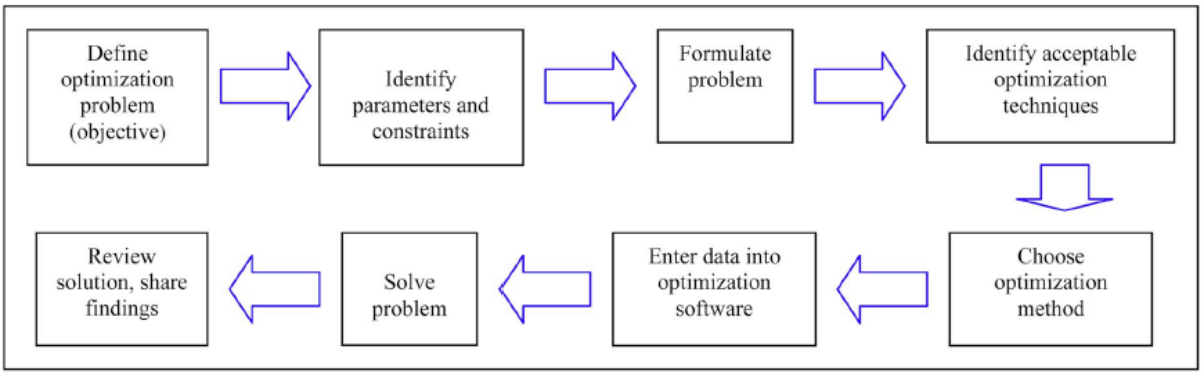


Figure A 2: Ontology Development Approach for Engineering Design Optimisation with DFM [Adopted from (Ahmed et al., 2007; Witherell et al., 2007)]

As observed from Figure A 1 and A 2, the various steps include –

- Define the scope of the problem domain – Engineering knowledge capture based on a model driven approach with focus on re-usable and generic processes wit their effect on product attributes

- Formulating the problem domain with knowledge entities such as activities, object, rule, logic, function and behaviour as high-level concepts. The complete 3 level description of concepts has been illustrated in Figure 4-3 in Chapter 4
- The optimisation method should involve inference and query supporting OWL/SWRL as an integrated layer based on description logic and fragment of horn logic. The optimisation should generate both text based description as string type and product attributes as float type
- Define the class hierarchy, properties with data typing as string, float and population with instances based on pilot and validation use-cases
- Input all the specified data using Protégé IDE as the supporting tool
- Run the Pellet reasoner along with Drools and SQWRL query language on the knowledgebase to generate results
- Verify the inference and query results with specific rule outputs to the rule outputs controlling product configuration and topology inside platform specific DEA systems.

C. OWL Ontology Model – Platform Independent and Neutral Formal Representation System

This document contains the classes, properties and restrictions of the GPM-DEA model mapped to its OWL2 based ontology model, developed in this research.

i. Class Hierarchy

owl:Thing
ProcessModel:Activity (http://example.org/ProcessModel#Activity)
    ProcessModel:Informatical-Activity
    ProcessModel:Physical-Activity
    ProcessModel:Virtual-Activity
ProcessModel:Engineering_Design_Process(http://example.org/ProcessModel#Engineering_Design_Process)
    ProcessModel:Computational_Fluid_Dynamics_CFD
        ProcessModel:Fluid_Flow_Analysis
        ProcessModel:Thermal_Analysis

ProcessModel:Design_for_Cost
ProcessModel:Design_for_Ergonomics
ProcessModel:Design_for_Manufacturing_Assembly
    ProcessModel:Additive_Manufacturing
    ProcessModel:Casting
        ProcessModel:Centrifugal_Casting
        ProcessModel:Die_Casting
        ProcessModel:Permanent_Mould_Casting
    ProcessModel:Forming
        ProcessModel:Blanking
        ProcessModel:Extrusion
            ProcessModel:Cold_Extrusion
            ProcessModel:Hot_Extrusion
ProcessModel:Forging
    ProcessModel:Cold_Forging
    ProcessModel:Drop_Forging
    ProcessModel:Hot_Forging
    ProcessModel:Precision_Forging
    ProcessModel:Press_Forging
ProcessModel:Heading
        ProcessModel:Punching_Piercing
        ProcessModel:Rolling
            ProcessModel:Cold_Rolling
            ProcessModel:Hot_Rolling
        ProcessModel:Stamping_or_Pressing
            ProcessModel:Cold_Pressing
            ProcessModel:Hot_Pressing
        ProcessModel:Thermo_Forming
        ProcessModel:Vacuum_Forming
    ProcessModel:Joining
        ProcessModel:Brazing
        ProcessModel:Riveting
        ProcessModel:Welding
    ProcessModel:Machining
        ProcessModel:Boring
        ProcessModel:CNC_Machining
        ProcessModel:Drilling
        ProcessModel:Electrical_Discharge_Machining
        ProcessModel:Electro_Chemical_Machining
        ProcessModel:Milling
        ProcessModel:Reaming

ProcessModel:Turning
ProcessModel:Moulding
ProcessModel:Blow_Moulding
ProcessModel:Compression_Moulding
ProcessModel:Injection_Moulding
ProcessModel:Design_for_Recycling
ProcessModel:Finite_Element_Analysis_FEA
ProcessModel:Stress_Analysis
ProcessModel:Structural_Analysis
ProcessModel:Mechanical_Design
ProcessModel:Feature
ProcessModel:Attach_Connect_Parts
ProcessModel:Depression_Extrusion
ProcessModel:Hole
ProcessModel:Notch
ProcessModel:Pocket
ProcessModel:Slot
ProcessModel:Protrusion
ProcessModel:Block
ProcessModel:Shaft
ProcessModel:Fit
ProcessModel:Assembly
ProcessModel:Part
ProcessModel:Form
ProcessModel:Edge
ProcessModel:Chamfer
ProcessModel:Fillet
ProcessModel:Line
ProcessModel:Face
ProcessModel:Circle
ProcessModel:Ellipse
ProcessModel:Hyperbola
ProcessModel:Parabola
ProcessModel:Polygon
ProcessModel:Surface
ProcessModel:Bézier_Surface
ProcessModel:NURBS_Surface
ProcessModel:Volume
ProcessModel:Box
ProcessModel:Cone
ProcessModel:Cylinder

ProcessModel:Ellipsoid
ProcessModel:Hyperboloid
ProcessModel:Paraboloid
ProcessModel:Polygon_Volume
ProcessModel:Sphere
ProcessModel:Material_Selection
ProcessModel:Alloys
ProcessModel:Brass
ProcessModel:Bronze
ProcessModel:Duralumin
ProcessModel:Inconel
ProcessModel:Manganin
ProcessModel:Nimonic
ProcessModel:Ceramics
ProcessModel:Boron_Carbide
ProcessModel:Boron_Oxide
ProcessModel:Silicon_Carbide
ProcessModel:Silicon_Nitride
ProcessModel:Composites
ProcessModel:Carbon_Fiber
ProcessModel:Glass_Fiber
ProcessModel:Kevlar
ProcessModel:Reinforced_Plastic
ProcessModel:Ferrous_Metal
ProcessModel:Carbon_Steel
ProcessModel:Cast_Iron
ProcessModel:Mild_Steel
ProcessModel:Stainless_Steel
ProcessModel:Wrought_Iron
ProcessModel:Non_Ferrous_Metal
ProcessModel:Aluminium
ProcessModel:Copper
ProcessModel:Lead
ProcessModel:Nickel
ProcessModel:Tin
ProcessModel:Titanium
ProcessModel:Zinc
ProcessModel:Polymer
ProcessModel:Neoprene
ProcessModel:Plastic
ProcessModel:Polyethylene

ProcessModel:Polypropylene
ProcessModel:Polystyrene
ProcessModel:Polyvinyl_Chloride
ProcessModel:Wood
ProcessModel:Multi_Body_Dynamics_MBD
ProcessModel:Electromagnetic_Analysis
ProcessModel:Kinematic_Analysis
ProcessModel:Stages
ProcessModel:Conceptual_Design
ProcessModel:Detailed_Design
ProcessModel:Computer_Aided_Design_CAD
ProcessModel:Computer_Aided_Engineering_CAE_Analysis
ProcessModel:Computer_Aided_Manufacturing_CAM
ProcessModel:Embodiment_Design
ProcessModel:Function—FunctionalRequirement(http://example.org/ProcessModel#Function--FunctionalRequirement)
ProcessModel:Assess_Product_Initial
ProcessModel:Geometric_3D_Analysis
ProcessModel:Analysis_Stage
ProcessModel:Analysis_Solving
ProcessModel:Post_Processing
ProcessModel:Pre_Processing
ProcessModel:Apply_Boundary_Conditions
ProcessModel:Dirichlet_Boundary_Conditions
ProcessModel:Neumann_Boundary_Conditions
ProcessModel:Robin_Boundary_Conditions
ProcessModel:Meshing
ProcessModel:Hexahedron
ProcessModel:Pyramid
ProcessModel:Quadrilateral
ProcessModel:TetraHedron
ProcessModel:Triangle_
ProcessModel:Triangular_Prism
ProcessModel:Geometric_3D_Modelling
ProcessModel:Create_Point_Cloud
ProcessModel:Create_Solid_as_Added_Volume_Boolean
ProcessModel:Add_Box_Volume
ProcessModel:Add_Cone_Volume
ProcessModel:Add_Cylinder_Volume
ProcessModel:Add_Ellipsoid_Volume
ProcessModel:Add_Polygon_Volume

ProcessModel:Add_Sphere_Volume
ProcessModel:Create_Surface_Volume_Boolean
ProcessModel:Create_Surface_Volume_Bézier
ProcessModel:Create_Surface_Volume_NURBS
ProcessModel:Remove_Solid_as_Subtracted_Volume_Boolean
ProcessModel:Subtract_Box_Volume
ProcessModel:Subtract_Cone_Volume
ProcessModel:Subtract_Cylinder_Volume
ProcessModel:Subtract_Ellipsoid_Volume
ProcessModel:Subtract_Polygon_Volume
ProcessModel:Subtract_Sphere_Volume
ProcessModel:Manufacturing_Feasibility
ProcessModel:Attach_Connect
ProcessModel:Assemble_Parts
ProcessModel:Attach_Connect_Fixture
ProcessModel:Attach_Connect_Jig
ProcessModel:CNC_Path_Instructions
ProcessModel:Costing
ProcessModel:Manufacturing_Method
ProcessModel:Material_Allocation
ProcessModel:Positioning
ProcessModel:Axial
ProcessModel:Circumferential
ProcessModel:Concentric
ProcessModel:Radial
ProcessModel:Tangential
ProcessModel:Quality_Control
ProcessModel:Measurement_Capability
ProcessModel:Precision_Accuracy
ProcessModel:Tool_Selection
ProcessModel:Output_Performance_Evaluation
ProcessModel:Electrical_Magnetic_Performance
ProcessModel:Capacitance
ProcessModel:Current
ProcessModel:Electric_Field
ProcessModel:Electro_Magnetic_Energy
ProcessModel:Electric_Energy
ProcessModel:Magnetic_Energy
ProcessModel:Electro_Magnetic_Power
ProcessModel:Electro_Magnetic_Work
ProcessModel:Induction

ProcessModel:Magnetic_Field
ProcessModel:Voltage
ProcessModel:Mechanical_Performance
ProcessModel:Acceleration
ProcessModel:Angular_Momentum
ProcessModel:Fatigue
ProcessModel:Force
ProcessModel:Foreign_Object_Damage
ProcessModel:Hardness
ProcessModel:Linear_Momentum
ProcessModel:Mechanical_Energy
ProcessModel:Elastic_Energy
ProcessModel:Gravitational_Energy
ProcessModel:Kinetic_Energy
ProcessModel:Potential_Energy
ProcessModel:Mechanical_Power
ProcessModel:Mechanical_Work
ProcessModel:Pressure
ProcessModel:Speed
ProcessModel:Stiffness
ProcessModel:Strain
ProcessModel:Strength
ProcessModel:Stress
ProcessModel:Torque
ProcessModel:Velocity
ProcessModel:Vibration
ProcessModel:Thermodynamic_Performance
ProcessModel:Compression
ProcessModel:Expansion
ProcessModel:Flow
ProcessModel:Foreign_Object_Damage
ProcessModel:Heat
ProcessModel:Pressure
ProcessModel:Thermodynamic_Energy
ProcessModel:Kinetic_Energy
ProcessModel:Potential_Energy
ProcessModel:Thermal_Energy
ProcessModel:Thermodynamic_Power
ProcessModel:Thermodynamic_Work
ProcessModel:Velocity
ProcessModel:Vibration

ProcessModel:Logic

ProcessModel:Object(http://example.org/ProcessModel#Object)

(The object model has the same classes as Feature, Form, Fit and Material Selection. All these 4 classes with their class hierarchy have been assigned subclasses of both Object class and Mechanical Design Class by the author. The object model has 1 additional sub-class, which is shown below)

ProcessModel:Product

ProcessModel:Product_Final

ProcessModel:Product_Initial

ProcessModel:Resources (http://example.org/ProcessModel#Resources)

ProcessModel:Rule (http://example.org/ProcessModel#Rule)

ProcessModel:Configuration_Rule

ProcessModel:Geometry_Rule

ProcessModel:Heuristic_Rule

ProcessModel:Logic_Rule

ProcessModel:Math_Rule

ProcessModel:Process_Rule

ProcessModel:Production_Rule

ProcessModel:Sub-Activity(http://example.org/ProcessModel#Sub-Activity)

It can be observed from the class hierarchy that a few classes such as Velocity, Vibration, Kinetic energy, and Potential energy occur under more than 1 class. In the ontology editor, these classes only exist as 1 class and have been marked as subclasses of multiple classes such as Thermodynamic performance and Mechanical performance in this work, similar to the object model class hierarchy.

ii. Properties

1. Object Properties with Domain and Range

ProcessModel:affectedbyLogic

Domain - ProcessModel:Activity

Range - ProcessModel:Logic

ProcessModel:Assesses

Domain - ProcessModel:Assess_Product_Initial

Range - ProcessModel:Product_Initial

ProcessModel:consists_of_Activity
Domain - ProcessModel:Engineering_Design_Process
Range - ProcessModel:Activity

ProcessModel:consists_of_Object
Domain - ProcessModel:Engineering_Design_Process
Range - ProcessModel:Object, ProcessModel:Product

ProcessModel:consumes_Product_Initial
Domain - ProcessModel:Engineering_Design_Process
Range - ProcessModel:Product_Initial

ProcessModel:controlled_by_Rule
Domain - ProcessModel:Activity
Range - ProcessModel:Rule

ProcessModel:fulfills_Function
Domain - ProcessModel:Object, ProcessModel:Product
Range - ProcessModel:Function—FunctionalRequirement

ProcessModel:governedbyLogic
Domain - ProcessModel:Rule
Range - ProcessModel:Logic

ProcessModel:has_Edge
Domain - ProcessModel:Object, ProcessModel:Product
Range - ProcessModel:Edge

ProcessModel:has_Face
Domain - ProcessModel:Object, ProcessModel:Product
Range - ProcessModel:Face

ProcessModel:has_Feature
Domain - ProcessModel:Assembly, ProcessModel:Object, ProcessModel:Part
Range - ProcessModel:Feature

ProcessModel:has_Form
Domain - ProcessModel:Object, ProcessModel:Product

Range - ProcessModel:Form

ProcessModel:has_Function
Domain - ProcessModel:Activity
Range - ProcessModel:Function—FunctionalRequirement

ProcessModel:has_Object_Material
Domain - ProcessModel:Object, ProcessModel:Product
Range - ProcessModel:Material_Selection

ProcessModel:has_Part
Domain - ProcessModel:Assembly
Range - ProcessModel:Part

ProcessModel:has_Successors
Domain - ProcessModel:Activity
Range - ProcessModel:Activity

ProcessModel:has_Surface
Domain - ProcessModel:Object, ProcessModel:Product
Range - ProcessModel:Surface

ProcessModel:hasSub-Activity
Domain - ProcessModel:Activity
Range – ProcessModel:Sub-Activity

ProcessModel:produces_Product_Final
Domain - ProcessModel:Engineering_Design_Process
Range - ProcessModel:Product_Final

ProcessModel:requires_Resources
Domain - ProcessModel:Activity
Range - ProcessModel:Resources

ProcessModel:satisfies_Functional_Requirement
Domain - ProcessModel:Engineering_Design_Process
Range - ProcessModel:Function—FunctionalRequirement

ProcessModel:Starts_with_Activity
Domain - ProcessModel:Engineering_Design_Process

Range - ProcessModel:Activity

2. Datatype Properties with Domain and Range

ProcessModel:has_Attributes
Domain - ProcessModel:Object, ProcessModel:Product
Range - xsd:float

Following have been created as the sub-properties of the datatype property in this work - ProcessModel:has_Attributes -
ProcessModel:has_Object_Orientation_Angle,
ProcessModel:has_Object_Position_Coordinates, ProcessModel:has_Object_Size
Domain - ProcessModel:Object, ProcessModel:Product
Range - xsd:float

Following have been created as the sub-properties of ProcessModel:has_Object_Orientation_Angle -
ProcessModel:has_Object_Orientation_X_Axis,
ProcessModel:has_Object_Orientation_Y_Axis,
ProcessModel:has_Object_Orientation_Z_Axis
Domain - ProcessModel:Object, ProcessModel:Product
Range - xsd:float

Following have been created as the sub-properties of ProcessModel:has_Object_Position_Coordinates –
ProcessModel:has_Object_X_Coordinate, ProcessModel:has_Object_Y_Coordinate,
ProcessModel:has_Object_Z_Coordinate
Domain - ProcessModel:Object, ProcessModel:Product
Range - xsd:float

Following have been created as the basic sub-properties of ProcessModel:has_Object_Size –
ProcessModel:has_Object_Depth, ProcessModel:has_Object_Height,
ProcessModel:has_Object_Width
Domain - ProcessModel:Object, ProcessModel:Product
Range - xsd:float

However, it is very crucial to note that other properties can be created by the user as additional sub-properties of ProcessModel:has_Object_Size, as it has been illustrated with both test case 4 and 5 in this thesis. For example, test case 4 has an additional sub-property

named as ProcessModel:has_Object_Diameter as a sub-property of ProcessModel:has_Object_Size.

Other datatype properties have been created such as -

ProcessModel:has_Surface_Area, ProcessModel:has_Surface_Finish, ProcessModel:has_Tolerance, ProcessModel:has_Volume
Domain - ProcessModel:Object, ProcessModel:Product
Range - xsd:float

ProcessModel:has_Temperature_Limit, ProcessModel:has_Youngs_Mod
Domain - ProcessModel:Material_Selection
Range – xsd:float

ProcessModel:has_ID
Domain - ProcessModel:Activity
Range - xsd:integer

ProcessModel:has_Inputs, ProcessModel:has_Outputs
Domain - ProcessModel:Activity
Range – xsd:float

Pertaining to a specific use-case, all the object properties as described above can be classified as sub-properties of ProcessModel:has_Inputs, ProcessModel:has_Outputs to indicate inputs and outputs of activity completion and execution in terms of its product attributes as developed in this research. Both test use-cases 4 and 5 have adopted the same approach to create properties with various object attributes as activity inputs and outputs to reflect the working of the model GPM-DEA as developed by the author.

Existential restrictions have been created on the activity class in order to describe it for a DEA system in this research, as explained in chapter 5. These are illustrated here as follows -

Class Name - - ProcessModel:Activity
Existential Restrictions –
ProcessModel:has_ID **some** xsd:integer
ProcessModel:has_Inputs **some** xsd:float
ProcessModel:has_Successors **some** ProcessModel:Activity

D. SWRL in built operators for utilisation and representation of Generative Modelling Functions and Engineering Rules

SWRL offers comparison, math and boolean built-ins on top of OWL classes, properties and restrictions and thus enhances the expressiveness of OWL ("SWRL Section 8. Built-Ins," 2009). These have been adopted by the author to represent generative modelling functions as described in chapter 5 and specific engineering rules for test use cases as described in chapter 6 as part of system development. These have been further experimentally verified in this research using the Pellet and Drools reasoner along with SQWRL query language using the test use cases in chapter 7 using Protégé IDE. Some of the in built operators by the author have been adopted from the following set as described in this appendix.

i. Comparison Operators

1. swrlb:equal(op:numeric-equal, op:compare, op:boolean-equalop:yearMonthDuration-equal, op:dayTimeDuration-equal, op:dateTime-equal, op:date-equal, op:time-equal, op:gYearMonth-equal, op:gYear-equal, op:gMonthDay-equal, op:gMonth-equal, op:gDay-equal, op:anyURI-equal)
   Satisfied if the first argument and the second argument are the same.

2. swrlb:notEqual(from swrlb:equal)
   The negation of swrlb:equal.

3. swrlb:lessThan (from XQuery op:numeric-less-than, op:compare, op:yearMonthDuration-less-than, op:dayTimeDuration-less-than, op:dateTime-less-than, op:date-less-than, op:time-less-than)
   Satisfied if the first argument and the second argument are both in some implemented type and the first argument is less than the second argument according to a type-specific ordering (partial or total), if there is one defined for the type. The ordering function for the type of untyped literals is the partial order defined as string ordering when the language tags are the same (or both missing) and incomparable otherwise.

4. swrlb:lessThanOrEqual (from swrlb:lessThan, swrlb:equal)
   Either less than, as above, or equal, as above.

5. swrlb:greaterThan(from XQuery op:numeric-greater-than, op:compare, op:yearMonthDuration-greater-than, op:dayTimeDuration-greater-than, op:dateTime-greater-than, op:date-greater-than, op:time-greater-than)
   Similarly to swrlb:lessThan.

6. swrlb:greaterThanOrEqual (from swrlb:greaterThan, swrlb:equal)
   Similarly to swrlb:lessThanOrEqual.

ii. Math Operators

1. swrlb:add(from XQuery op:numeric-add)
   Satisfied if the first argument is equal to the arithmetic sum of the second argument through the last argument.

2. swrlb:subtract (from XQuery op:numeric-subtract)
   Satisfied iff the first argument is equal to the arithmetic difference of the second argument minus the third argument.

3. swrlb:multiply (from XQuery op:numeric-multiply)
   Satisfied if the first argument is equal to the arithmetic product of the second argument through the last argument.

4. swrlb:divide (from XQuery op:numeric-divide)
   Satisfied iff the first argument is equal to the arithmetic quotient of the second argument divided by the third argument.

5. swrlb:integerDivide (from XQuery op:numeric-integer-divide)
   Satisfied if the first argument is the arithmetic quotient of the second argument idiv the third argument. If the numerator is not evenly divided by the divisor, then the quotient is the xsd:integer value obtained, ignoring any remainder that results from the division (that is, no rounding is performed).

6. swrlb:mod (from XQuery op:numeric-mod)
   Satisfied if the first argument represents the remainder resulting from dividing the second argument, the dividend, by the third argument, the divisor. The operation a mod b for operands that are xsd:integer or xsd:decimal, or types derived from them, produces a result such that (a idiv b)*b+(a mod b) is equal to a and the magnitude of the result is always less than the magnitude of b. This identity holds even in the special case that the dividend is the negative integer of largest possible magnitude for its type and the divisor is -1 (the remainder is 0). It follows from this rule that the sign of the result is the sign of the dividend

7. swrlb:pow
   Satisfied if the first argument is equal to the result of the second argument raised to the third argument power.

8. swrlb:abs (from XQuery fn:abs)
   Satisfied if the first argument is the absolute value of the second argument.

9. swrlb:round (from XQuery fn:round)
   Satisfied if the first argument is equal to the nearest number to the second argument with no fractional part.

10. swrlb:sin
    Satisfied if the first argument is equal to the sine of the radian value the second argument.

11. swrlb:cos
    Satisfied if the first argument is equal to the cosine of the radian value the second argument.

iii. Strings

1. swrlb:stringConcat (from XQuery fn:concat)
   Satisfied if the first argument is equal to the string resulting from the concatenation of the strings the second argument through the last argument.

2. swrlb:substring (from XQuery fn:substring)
   Satisfied if the first argument is equal to the substring of optional length the fourth argument starting at character offset the third argument in the string the second argument.

3. swrlb:contains (from XQuery fn:contains)
   Satisfied if the first argument contains the second argument (case sensitive).

4. swrlb:containsIgnoreCase
   Satisfied if the first argument contains the second argument (case ignored).

5. swrlb:startsWith (from XQuery fn:starts-with)
   Satisfied if the first argument starts with the second argument.

6. swrlb:endsWith (from XQuery fn:ends-with)
   Satisfied if the first argument ends with the second argument.

7. swrlb:matches (from XQuery fn:matches)
   Satisfied if the first argument matches the regular expression the second argument.

8. swrlb:replace (from XQuery fn:replace)
   Satisfied if the first argument is equal to the value of the second argument with every substring matched by the regular expression the third argument replaced by the replacement string the fourth argument.

Appendix 2: – Use Case 4 and 5 Axioms – Test Cases

E. Use Case 4

i. ParaPy Source Code – Created by Author

As shown in experimental verification with test use cases in chapter 7 in this work, one of the targets for the 4th experiment as designed by the author is to compare the rule output in ontology model as platform independent and neutral formal representation standards to the specific rule outputs inside platform specific and proprietary DEA systems such as ParaPy. The following is the source code created by the author in ParaPy as a platform specific DEA system to represent some of the specific engineering rules controlling the topology and configuration of the block as a product.

```
from __future__ import division
from parapy.core import *
from parapy.geom import *
from math import pi, degrees, radians

class Block(GeomBase):

    #: Block Dimensions - Width, Length(Height), Height(Depth)
    #: :type: float

    block_width = Input(50)          # Block Width(W)
    block_length = Input(60)         # Block Height(H)        #User Inputs

    @Attribute
    def block_height(self):          # Block Depth(D)    #Depth Rule
        return self.block_width*1.5

    @Part
    def block1(self):
        return Box(self.block_width if self.block_width>=50 else "ERROR",
#Dimension Rule
```

```
            self.block_length if self.block_length>=50 else "ERROR",  # Block Height(H) #Dimension Rule

            self.block_height if self.block_height>=50 else "ERROR",  # Block Depth(D) #Dimension Rule

            color="red")

    #: Hole Dimensions - Radius(Diameter/2), Length(Depth)
    #: :type: float

    hole_diameter = Input(30)          # Hole Diameter(HD1)          #User Input

    @Attribute
    def hole_radius(self):
        return self.hole_diameter/2     # Hole Radius = HD1/2

    hole_height = Input(40)    # Hole Depth(HD2)              #User Input

    @Part
    def hole1(self):
        return Cylinder(self.hole_radius if self.hole_radius*2.5<self.block_width
                                and
                                self.hole_radius*2.5<self.block_length else "ERROR",
 #HoleDiameter Rule
                        self.hole_height if self.hole_height<=self.block_height else "ERROR", #Hole Depth Rule

                        color="blue", position=self.position.translate('x', 30, 'y', 20, 'z', 35))

    @Part
    def blockwithhole1(self):
        return SubtractedSolid(shape_in=self.block1,
                               tool=self.hole1)                    #Subtraction of Volume for Drilling

if __name__ == '__main__':
    from parapy.gui import display
    obj = Block()
    display(obj)
```

ii. Variation of SWRL Rule Outputs for Block and Hole Attributes in Ontology and Comparison with ParaPy

The source code created by the author has resulted in variations in output with block and hole attributes in ParaPy as a platform specific DEA application in this research as shown below –

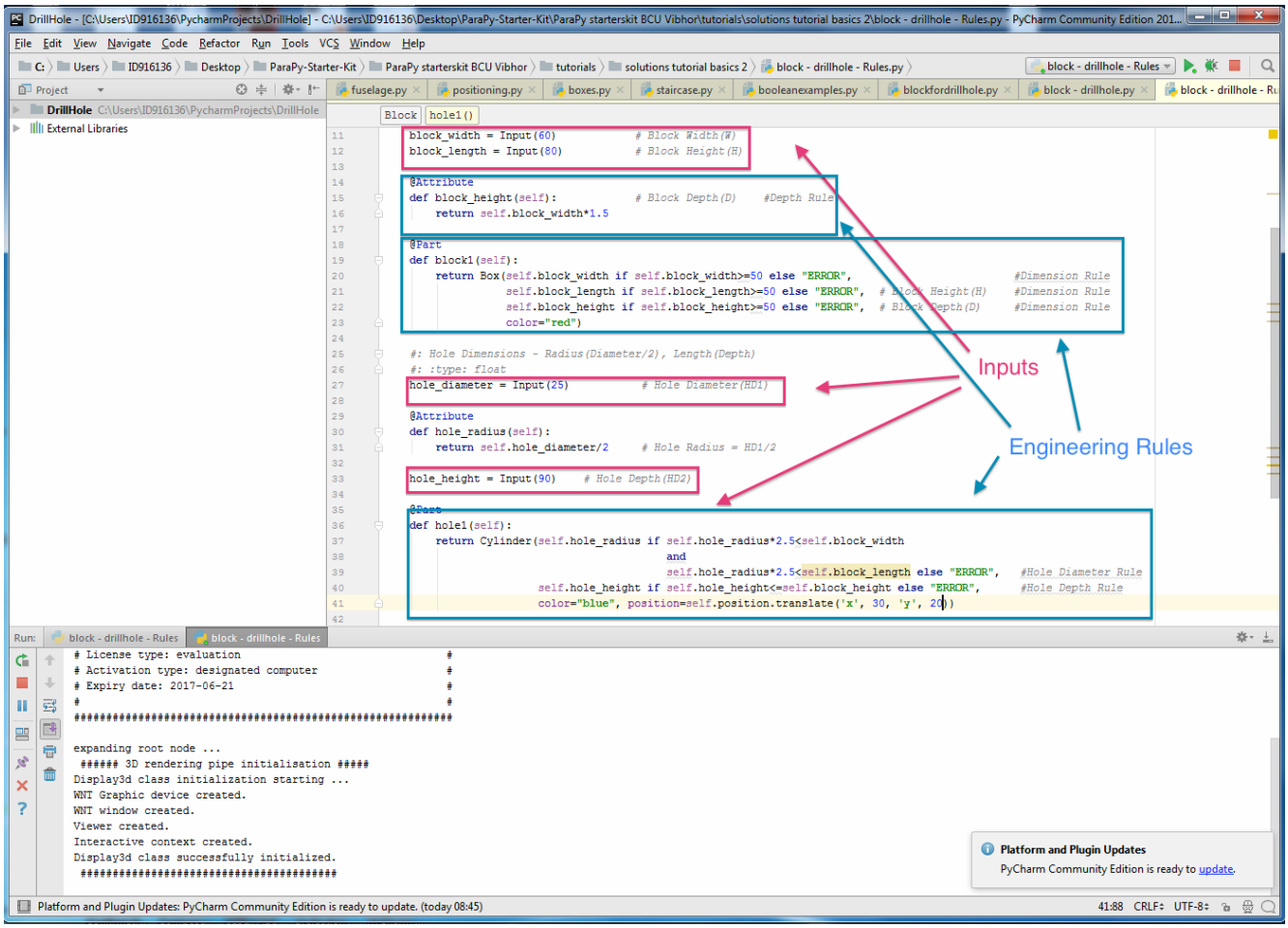


Figure B 1: ParaPy Source Code – Inputs and Rules for Block and Hole Attributes

The SWRL rule representation of the specified engineering rules in this research as part of the developed ontology model corresponding to GPM-DEA schema are explained as follows

Dimension Rule - Minimum dimensions of the block is 50 mm, W>=50mm, H>=50 mm, D>=50mm)

SWRL Representation - Product(?p) ^ hasWidth(?p, ?w) ^ swrlb:greaterThanOrEqual(?w, "50.0"^^xsd:float) ^ hasHeight(?p, ?h) ^ swrlb:greaterThanOrEqual(?h, "50.0"^^xsd:float) ^ hasDepth(?p, ?d) ^ swrlb:greaterThanOrEqual(?d, "50.0"^^xsd:float) -> sqwrl:select("Block adheres to dimensions")

Depth Rule - D=W*1.5

SWRL Representation - Product(?p) ^ hasWidth(?p, ?w) ^ swrlb:multiply(?x, ?w, "1.5"^^xsd:float) -> hasDepth(?p, ?x)

Hole Depth Rule - Hole depth should be less than or equal to depth of block, HD2<=D

SWRL Representation - Product(?p) ^ hasDepth(?p, ?d) ^ Hole(?h) ^ hasDepth(?h, ?d2) ^ swrlb:lessThanOrEqual(?d2, ?d) -> sqwrl:select(("Hole adheres to dimensions")

Else

Product(?p) ^ hasDepth(?p, ?y) ^ Hole(?h) ^ hasDepth(?h, ?z) ^swrlb:greaterThan(?z, ?y) -> sqwrl:select("Hole can't be created")

Hole Diameter Rule - HD1*1.25<W, HD1*1.25<H

SWRL Representation - Product(?p) ^ hasWidth(?p, ?a) ^ hasHeight(?p, ?b) ^ Hole(?h) ^ hasDiameter(?h, ?c) ^swrlb:multiply(?d, ?c, "1.25"^^xsd:float) ^ swrlb:lessThan(?d, ?a) ^ swrlb:lessThan(?d, ?b) ->sqwrl:select("Hole adheres to dimensions")

Else

Product(?p) ^ hasWidth(?p, ?e) ^ Hole(?h) ^ hasDiameter(?h, ?g) ^ swrlb:multiply(?i, ?g, "1.25"^^xsd:float) ^ swrlb:greaterThanOrEqual(?i, ?e) -> sqwrl:select("Hole can't be created")

Else

Product(?p) ^ hasHeight(?p, ?f) ^ Hole(?h) ^ hasDiameter(?h, ?g) ^ swrlb:multiply(?i, ?g, "1.25"^^xsd:float) ^ swrlb:greaterThanOrEqual(?i, ?f) -> sqwrl:select("Hole can't be created")

The output in product form through the Graphical User Interface (GUI) for visual display is illustrated with Figure B 3 and B4.

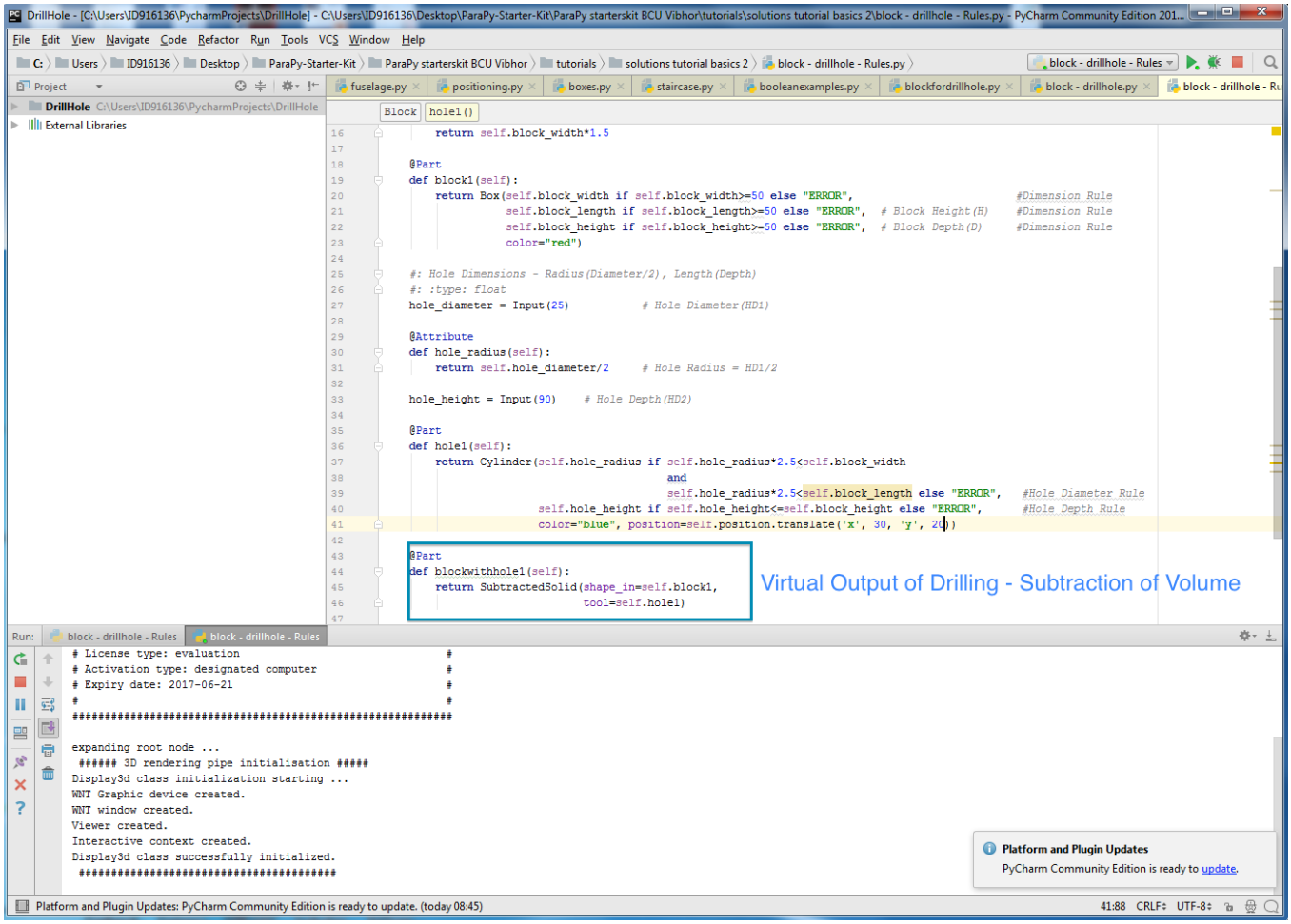


Figure B 2: ParaPy Source Code – Virtual Subtraction of Hole Volume

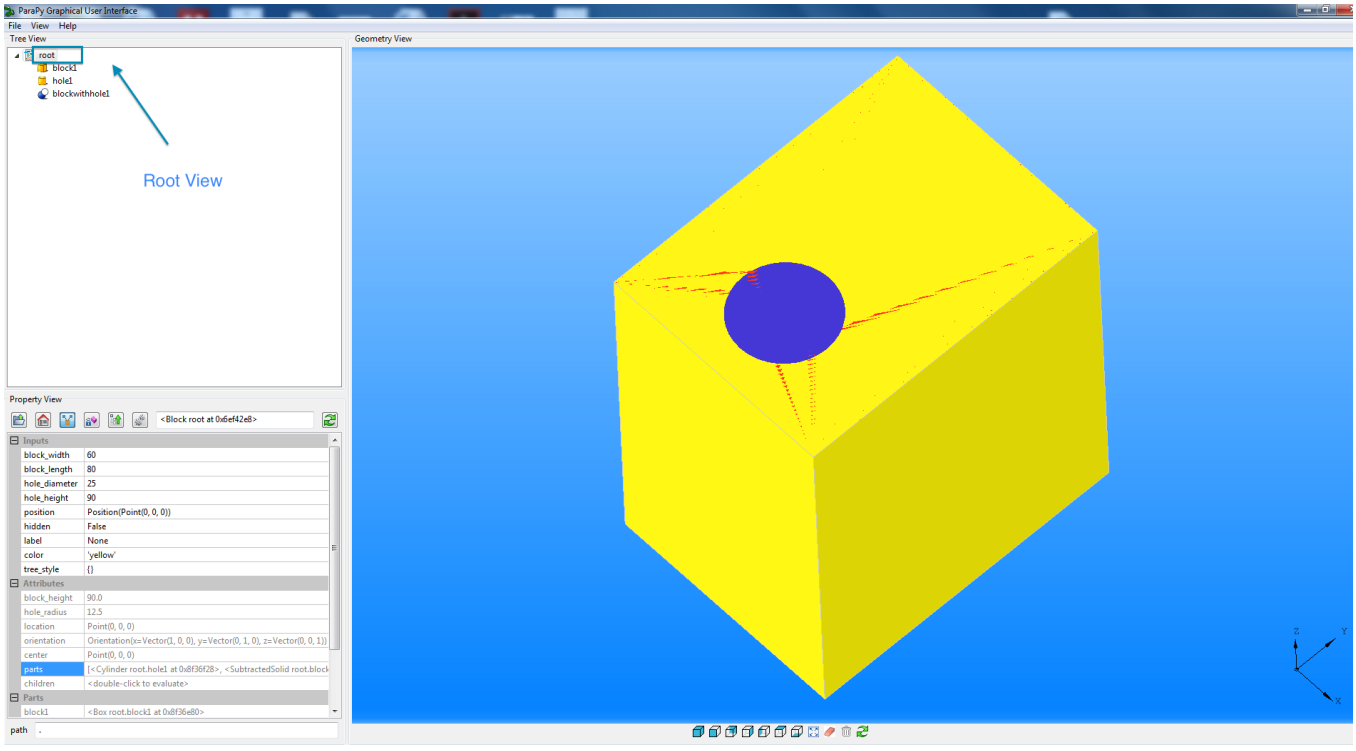


Figure B 3: ParaPy GUI – Output with Root View

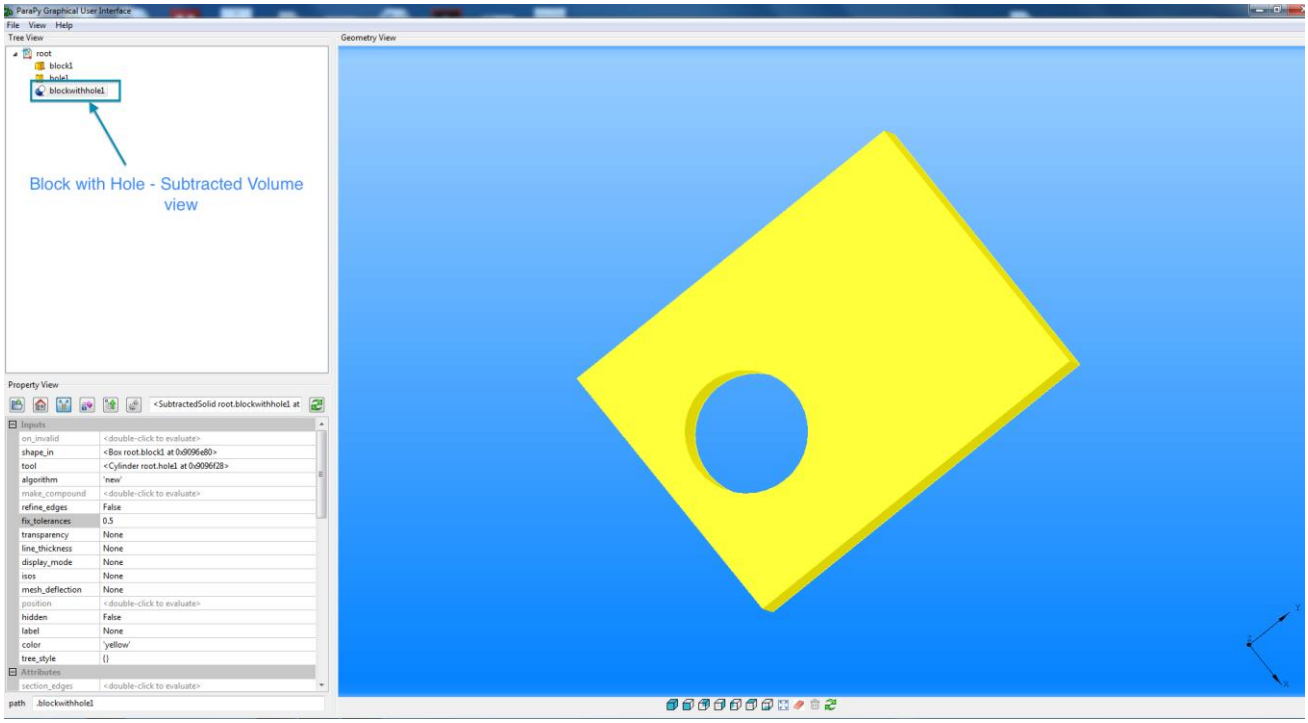


Figure B 4: ParaPy GUI – Output with Subtracted Volume View

The comparison of these specific values for block attributes and representation of SWRL engineering rules on top of OWL as platform independent and neutral representation with those against ParaPy as platform specific DEAS is shown below with Figure B 5 –

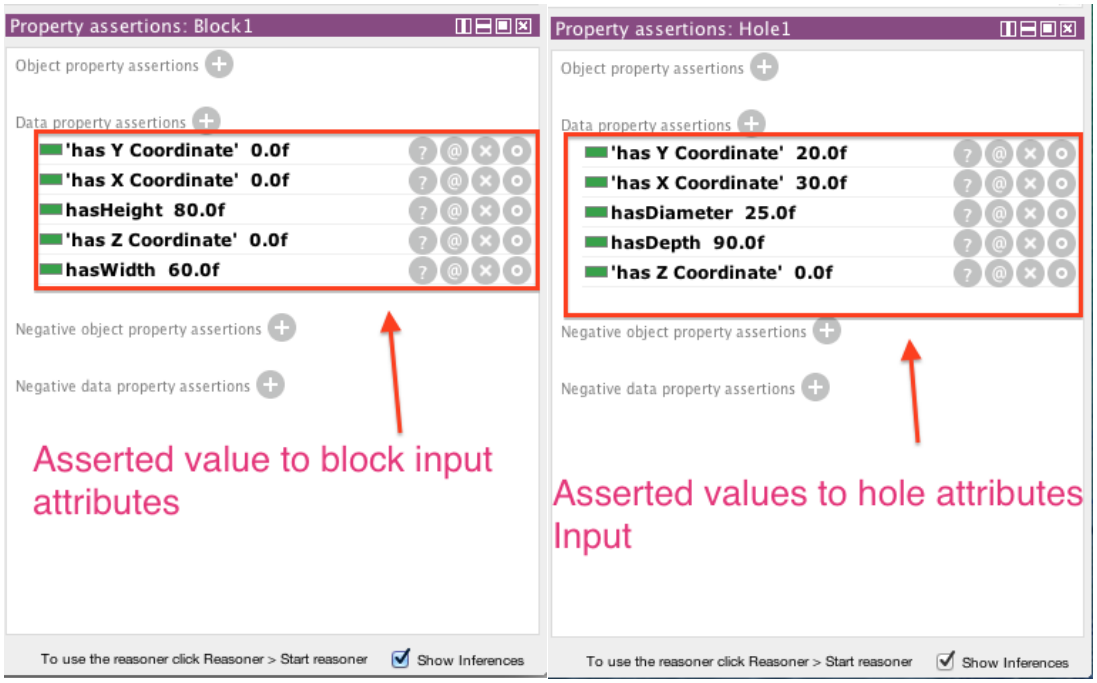


Figure B 5: Asserted Input Values – Block and Hole in OWL/SWRL Ontology

The output of SQWRL query results for the dimension, hole depth ad hole diameter rule is shown with Figure B 6 and B7.

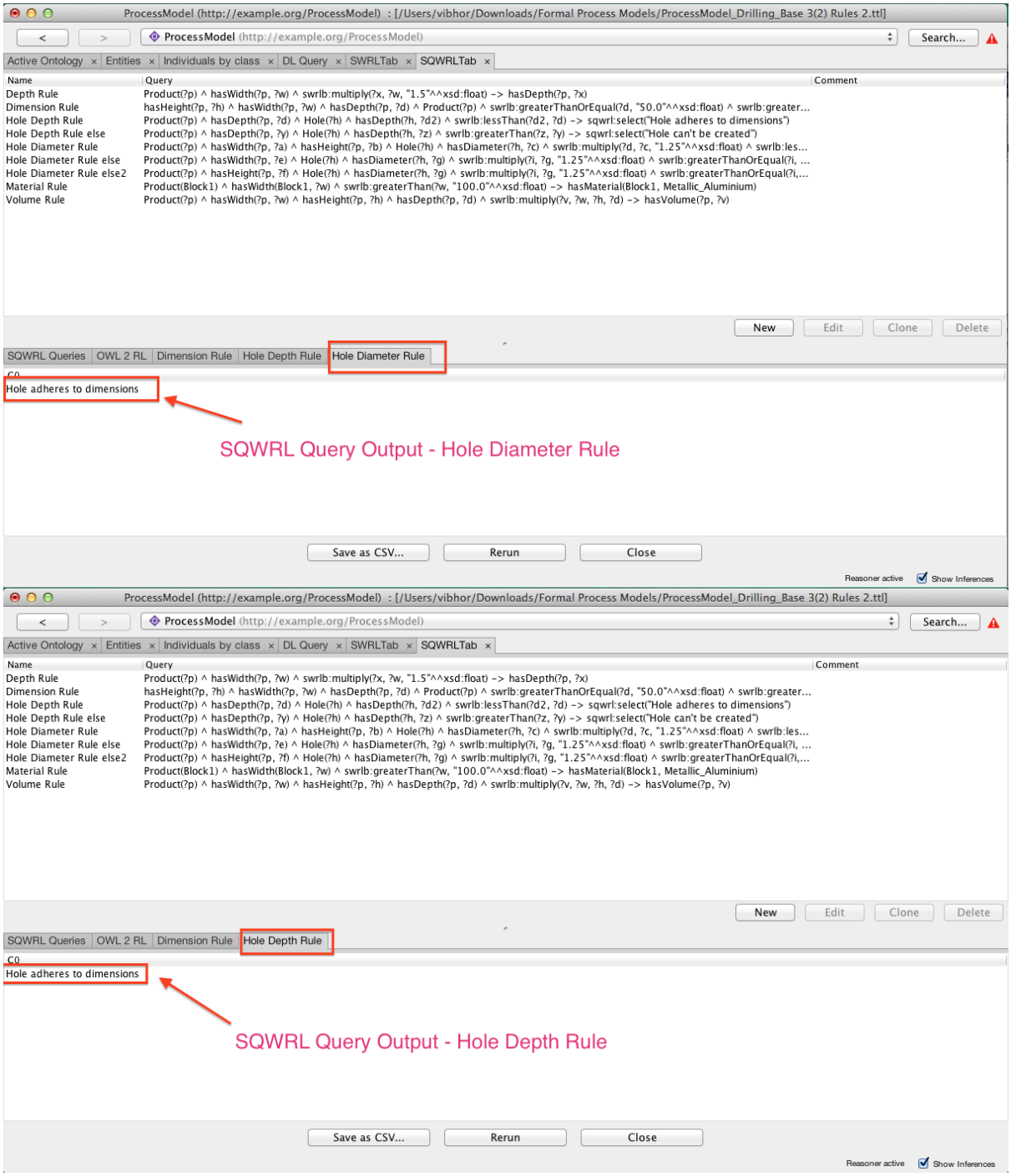


Figure B 6: SQWRL Query Results – Hole Diameter and Hole Depth Rule

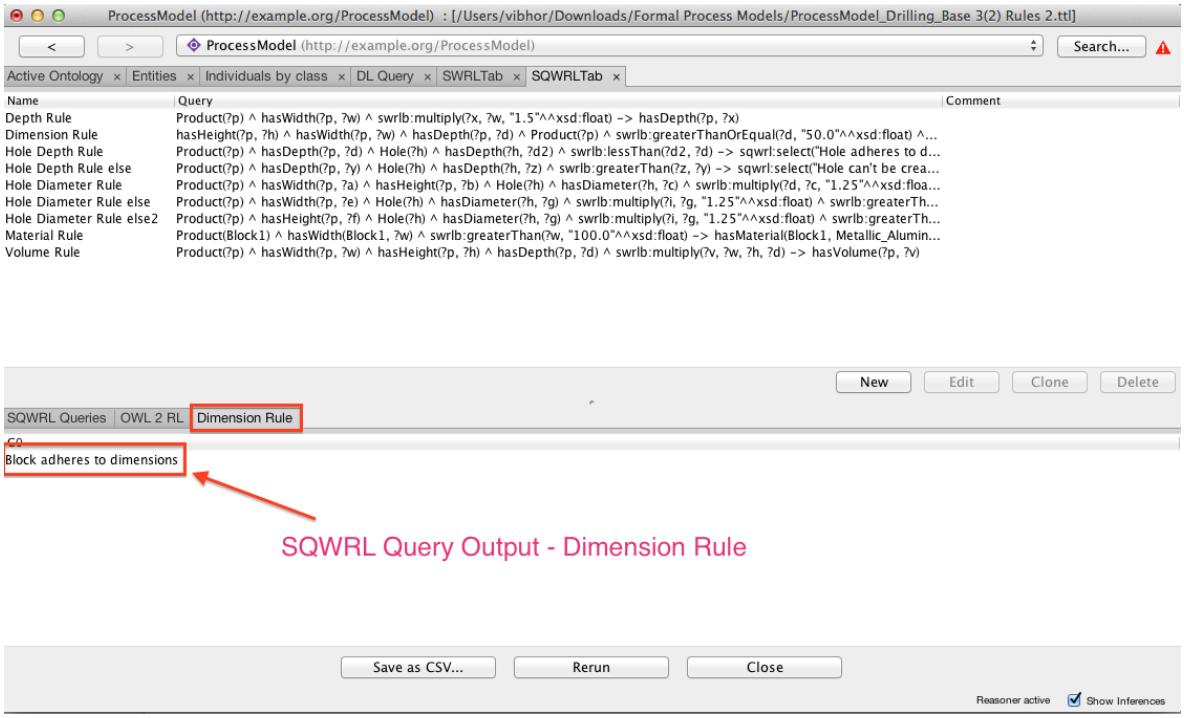


Figure B 7: SQWRL Query Result – Dimension Rule

It can be observed that the query results are inline with the specific output attributes of block and hole inside ParaPy although the output is supported by visual display through an inbuilt GUI. However, the ontology model although doesn't currently support and inbuilt GUI, the query results are accurate and provide semantic clarity. Similarly, the SWRL rule output for hole volume is supported with Figure B 8 and B 9.

It can be observed that the hole volume is similar inside both ontology model and ParaPy. However, there is a slight difference from 44178.64 mm^3 to 44156.25 mm^3due to the value of pi as π =3.141592653589793238 inside ParaPy and 3.14 used inside SWRL rule.

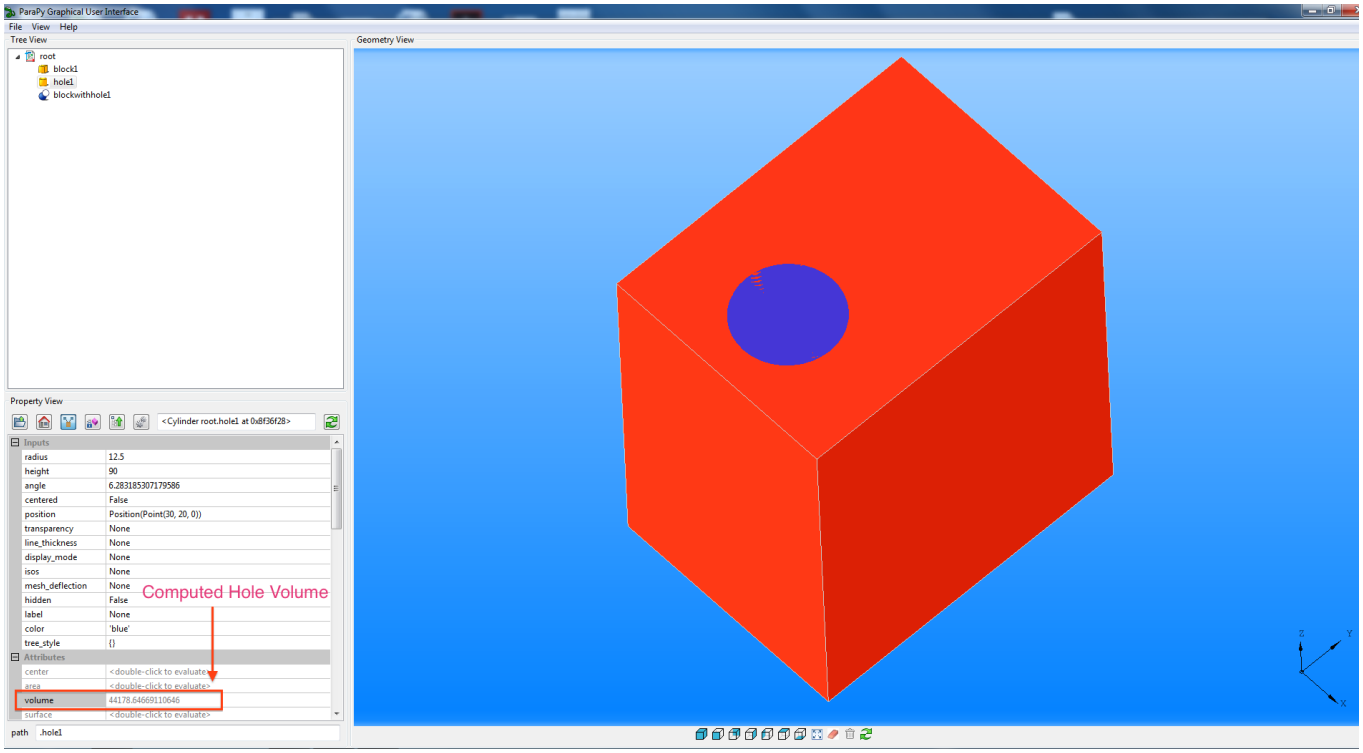


Figure B 8: Computed Hole Volume – ParaPy

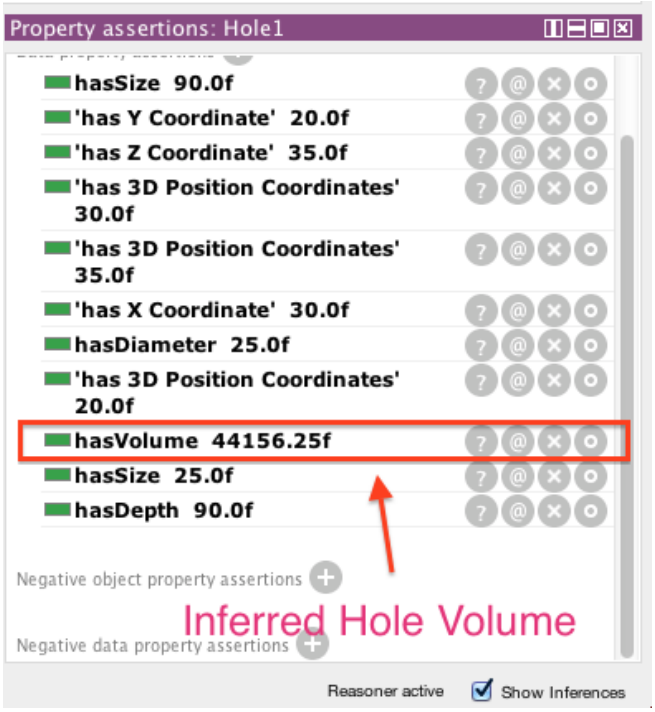


Figure B 9: Inferred Hole Volume – Hole Volume Rule – OWL/SWRL Ontology Model

A violation is also introduced by increasing the hole diameter from 25 to 50 mm which violates the Hole Diameter Rule keeping the block attributes same as above. The output inside ParaPy is shown with the help of Figure B 10.

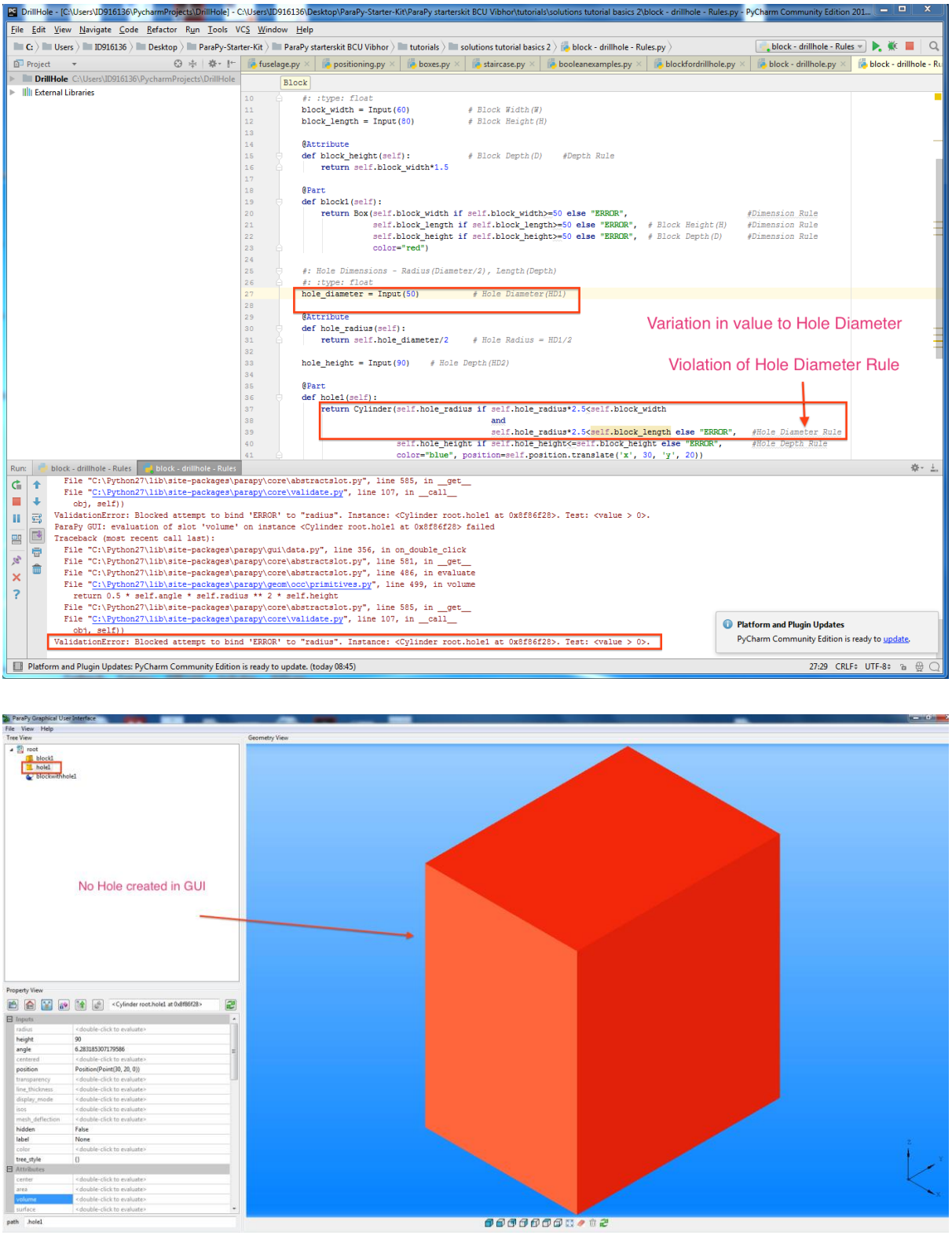


Figure B 10: Violation of Hole Diameter Rule – ParaPy

The output of violation of hole diameter as per Hole Diameter Rule in ontology model is explained with Figure B 11

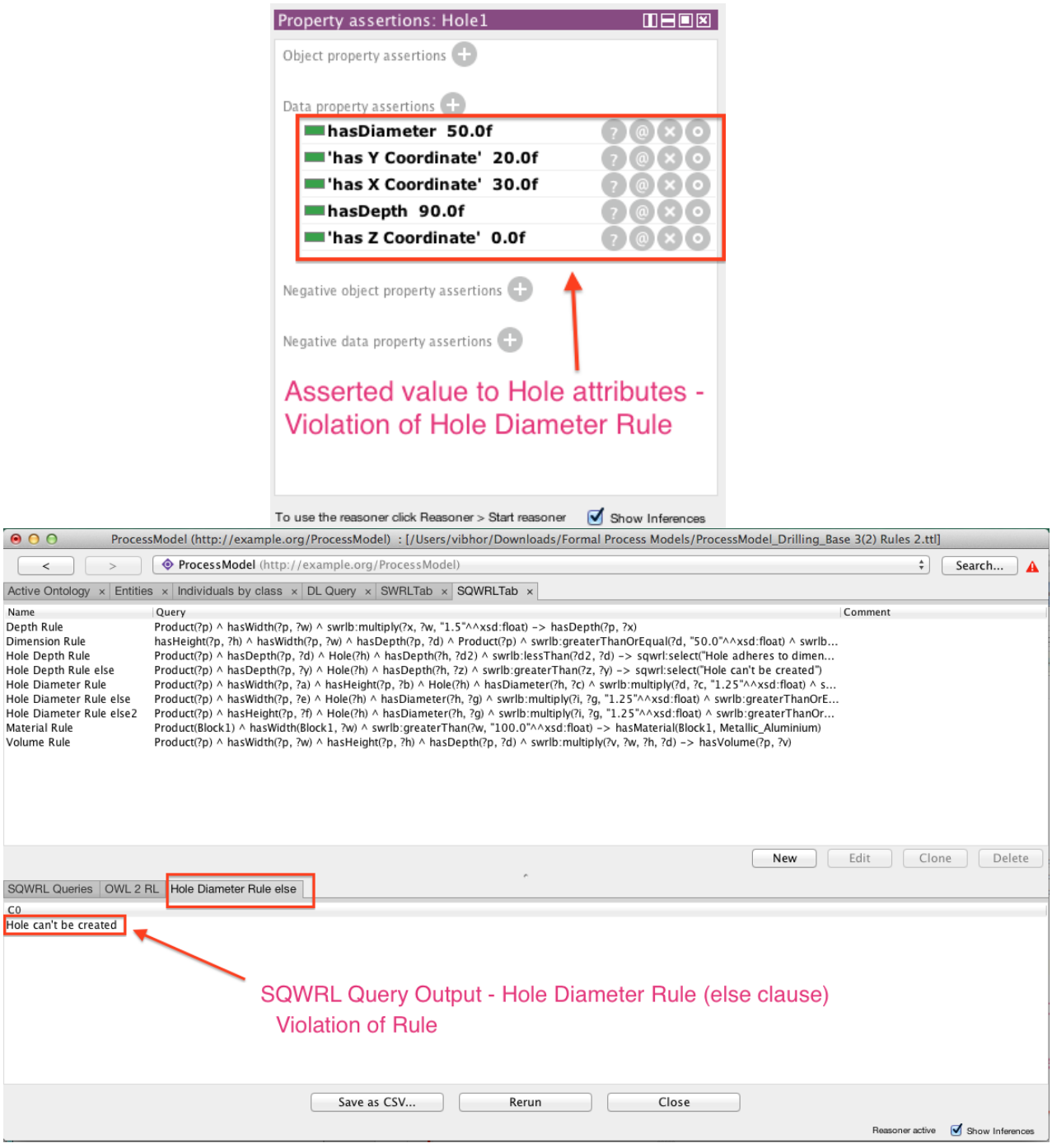


Figure B 11: Violation of Hole Diameter Rule – OWL/SWRL ontology model

Thus, all the results are in line with the results from the experimental verification of the knowledge representation system for Use Case 4 as Test Case performed in Chapter 7.

F. Use Case 5

iii. AML Source Code snippets

As illustrated in Chapter 6 and 7, Use Case 5 as Test Case is adopted from (Lützenberger et al., 2012) with addition of knowledge such as function, activity and object and rule association. The following section shows small snippets of AML source code as a platform specific DEA system for the engineering rules controlling the bookshelf topology and configuration.

;;;Filename: kbe-bookshelf-input-mixin.aml

```
(in-package :AML)
(define-class kbe-bookshelf-input-mixin
:inherit-from (object)
:properties (
;;; parameters set in GUI
height-input 5
width-input 3
max-hs-input 0.5
vertical-spacing-shelves-input 0.5
shelf-depth 0.7
thickness-bottom-shelf-input 0.05
thickness-top-shelf-input 0.05
thickness-dividing-walls-input 0.05
thickness-of-shelves-input 0.05
thickness-side-walls-input 0.05                    #Input Parameters
)
:subobjects (
)
)                                                  (Lützenberger et al., 2012, Pg 58, 59)

;;;-----------------------------------------------------
;;;Method for verification of width-input and max-hs-input
;;;-----------------------------------------------------
(define-method kbe-validate-bookshelf-width kbe-bookshelf-data-model-class ()
(if (< !width-input (* 0.5 (!max-hs-input)))
(pop-up-message "WRONG INPUT PARAMETERS: The bookshelf is too narrow. Adjust
bookshelf width or maximum horizontal length of one shelf. ")
nil                                    #Dividing Walls Rule
```

```
)
)
;;;-----------------------------------------------------
;;;Method for verification of height-input and vs-input
;;;-----------------------------------------------------
(define-method kbe-validate-bookshelf-height kbe-bookshelf-data-model-class ()
(if (> !vertical-spacing-shelves-input !height-input)
(pop-up-message "WRONG INPUT PARAMETERS: The bookshelf is too low
for even one vertical space in the bookshelf. Adjust bookshelf height or vertical
spacing between shelves. ")
nil                                    #Shelves Rule
)
)
```

(Lützenberger et al., 2012, Pg 61, 62)

iv. Variation in SWRL Rule Outputs for Bookshelf Attributes in Ontology

A few variations are produced in the bookshelf design ontology model by the author as per the modelled semantics of Dividing Walls Rule. These are shown with the support of Figure B 12, B 13 and B 14. Similarly, asserted values to bookshelf attributes as violation of Shelves Rule is shown with Figure B 15. The rule and their SWRL representations developed by this research are illustrated as follows -

Dividing Walls Rule – NDW is based on HS and W, If (W<0.5*HS, "ERROR") elseif (W<=HS, NDW=0) else (NDW=Int(W/HS)-1)

SWRL Representation - Product(?p) ^ has_Object_Width_W(?p, ?w) ^ has_Object_Horizontal_length_1_shelf_HS(?p, ?hs) ^ swrlb:multiply(?x, "0.5"^^xsd:float, ?hs) ^ swrlb:lessThan(?w, ?x) -> sqwrl:select("Error - Too narrow for a bookshelf")

And

Product(?p) ^ has_Object_Width_W(?p, ?w) ^ has_Object_Horizontal_length_1_shelf_HS(?p, ?hs) ^ swrlb:multiply(?x, "0.5"^^xsd:float, ?hs) ^ swrlb:greaterThan(?w, ?x) ^ swrlb:lessThanOrEqual(?w, ?hs) -> has_Object_No_dividing_walls_NDW(?p, "0.0"^^xsd:float)

And

Product(?p) ^ has_Object_Width_W(?p, ?w) ^ has_Object_Horizontal_length_1_shelf_HS(?p, ?hs) ^ swrlb:greaterThan(?w, ?hs) ^ swrlb:divide(?y, ?w, ?hs) ^ swrlb:subtract(?z, ?y, "1.0"^^xsd:float) -> has_Object_No_dividing_walls_NDW(?p, ?z)

Shelves Rule - (NSH is based on H and VS, If (VS>H, "ERROR") elseif (2*VS>H, NSH=0) else (NSH=Int((H/VS)-1))

SWRL Representation - Product(?p) ^ has_Object_Height_H(?p, ?h) ^ has_Object_Vertical_length_1_shelf_VS(?p, ?vs) ^ swrlb:greaterThan(?vs, ?h) -> sqwrl:select("Error - Too low for even one space in the bookshelf")

And

Product(?p) ^ has_Object_Height_H(?p, ?h) ^ has_Object_Vertical_length_1_shelf_VS(?p, ?vs) ^ swrlb:lessThan(?vs, ?h) ^ swrlb:multiply(?a, "2.0"^^xsd:float, ?vs) ^ swrlb:greaterThan(?a, ?h) -> has_Object_No_shelves_NSH(?p, "0.0"^^xsd:float)

And

Product(?p) ^ has_Object_Height_H(?p, ?h) ^ has_Object_Vertical_length_1_shelf_VS(?p, ?vs) ^ swrlb:multiply(?a, "2.0"^^xsd:float, ?vs) ^ swrlb:lessThan(?a, ?h) ^ swrlb:divide(?b, ?h, ?vs) ^ swrlb:subtract(?c, ?b, "1.0"^^xsd:float) -> has_Object_No_shelves_NSH(?p, ?c)

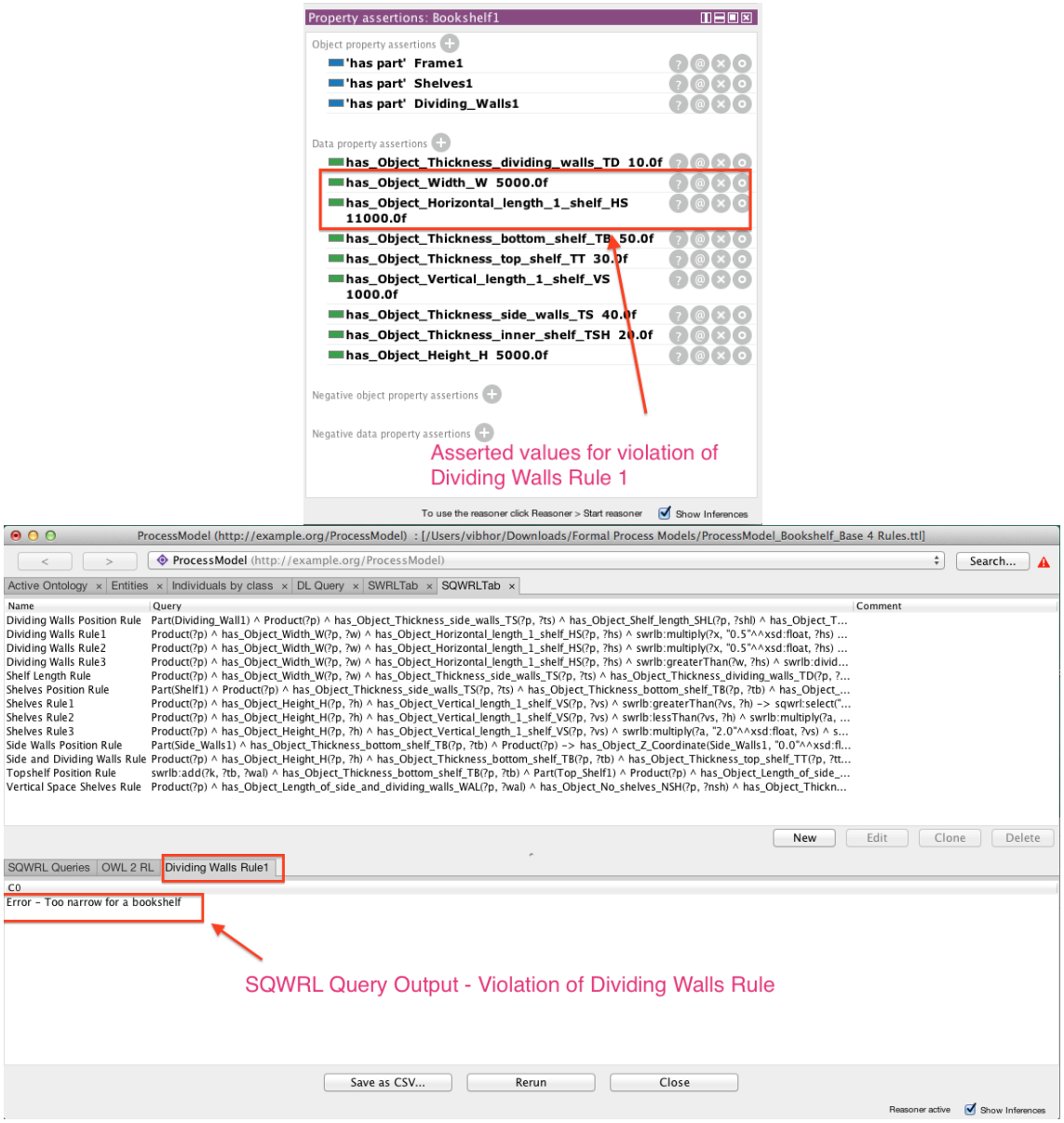


Figure B 12: Violation of Dividing Walls Rule1: Bookshelf Design Process Ontology Model

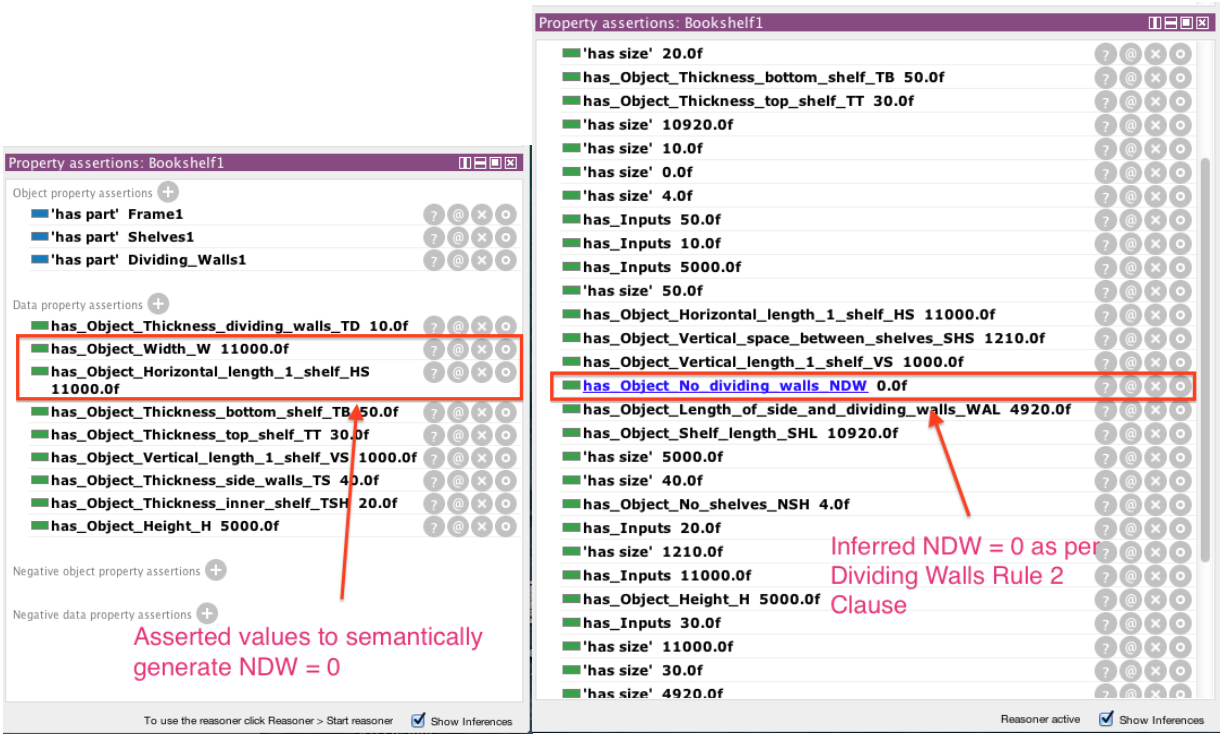


Figure B 13: Dividing Walls Rule Clause 2

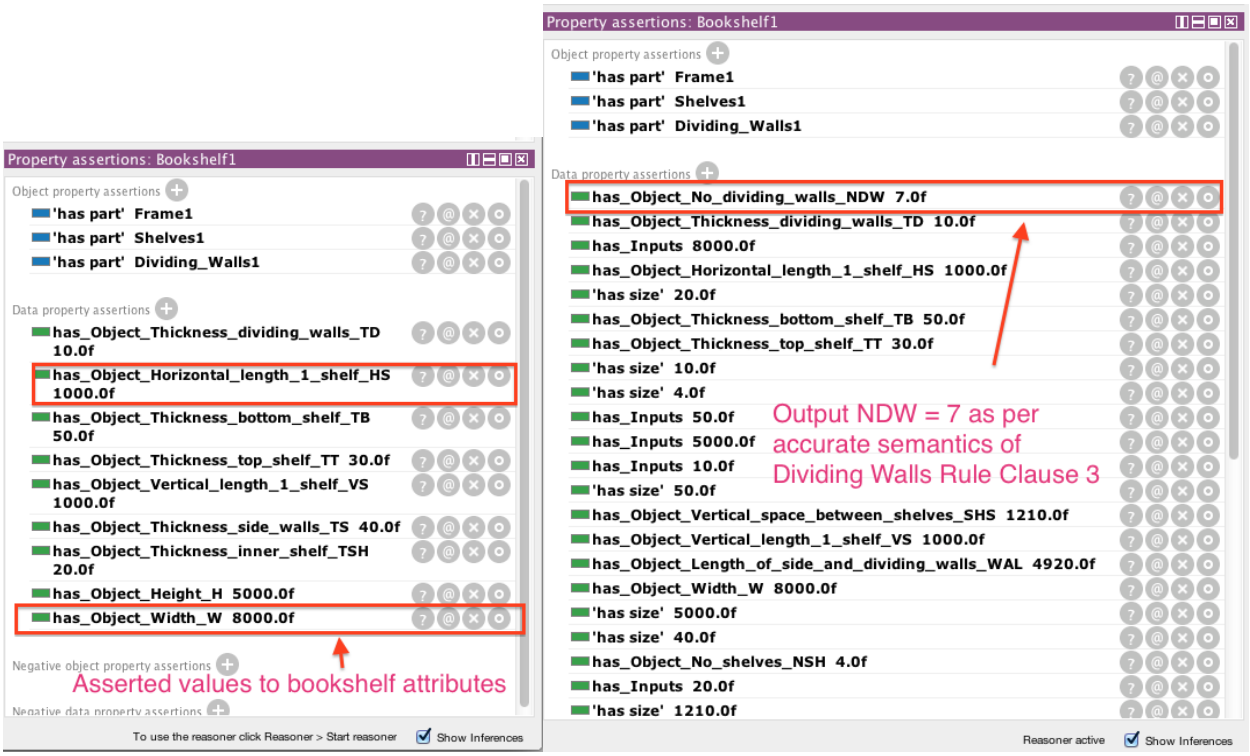


Figure B 14: Dividing Walls Rule Clause 3 – Variation in asserted Values

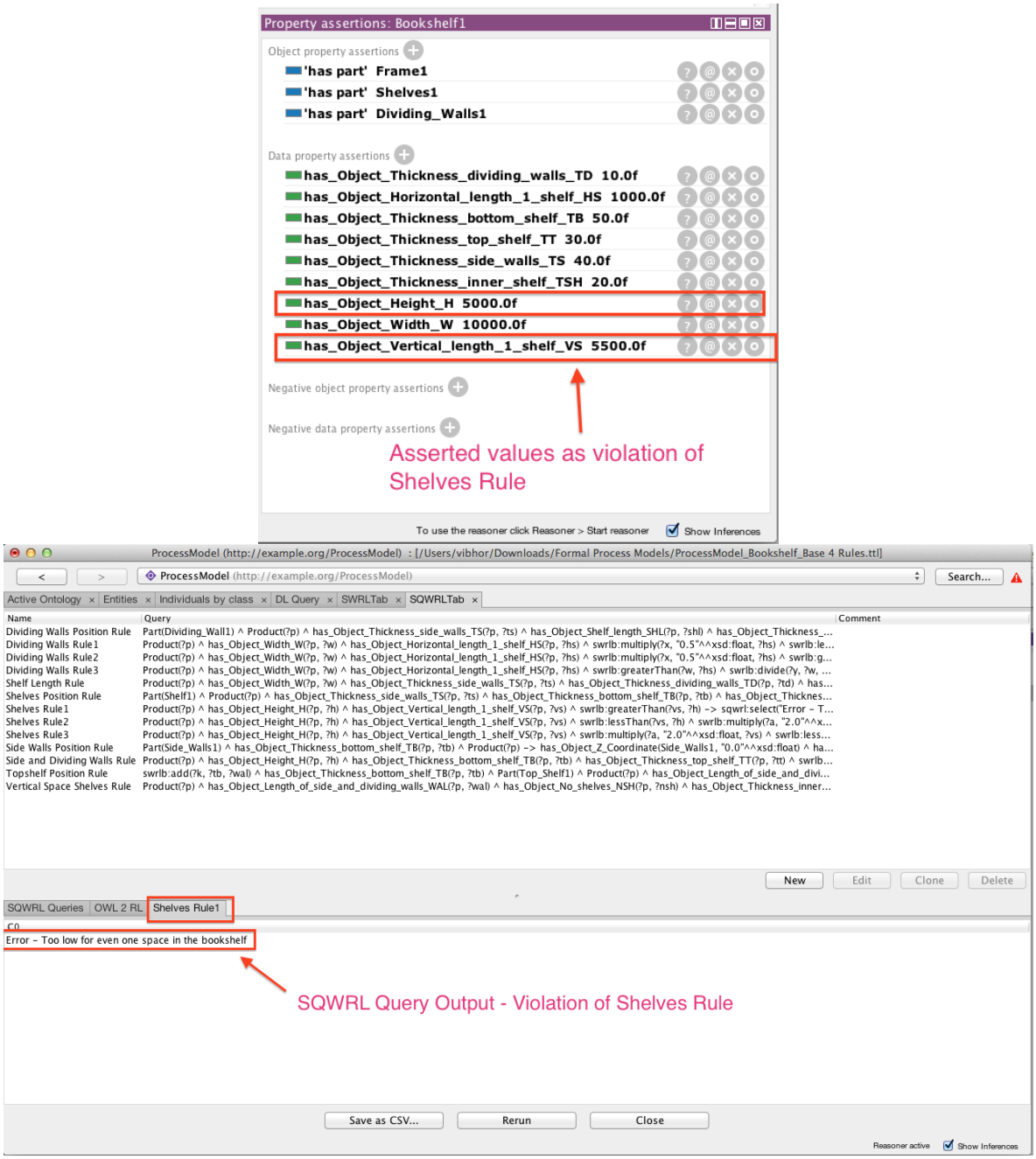


Figure B 15: SQWRL Query Output - Violation of Shelves Rule

Dividing Wall Position Rule - (X1=TS+SHL,Y1=TB, Z1=0)

SWRL Representation - Part(Dividing_Walls1) ^ Product(?p) ^ has_Object_Thickness_side_walls_TS(?p, ?ts) ^ has_Object_Shelf_length_SHL(?p, ?shl) ^ has_Object_Thickness_bottom_shelf_TB(?p, ?tb) ^ swrlb:add(?i, ?ts, ?shl) -> has_Object_X_Coordinate(Dividing_Walls1, ?i) ^ has_Object_Y_Coordinate(Dividing_Walls1, ?tb) ^ has_Object_Z_Coordinate(Dividing_Walls1, "0.0"^^xsd:float)

The dividing walls position as per the semantics of Dividing Walls Position Rule is indicated with Figure B 16. A variation in values is illustrated with Figure B 17.

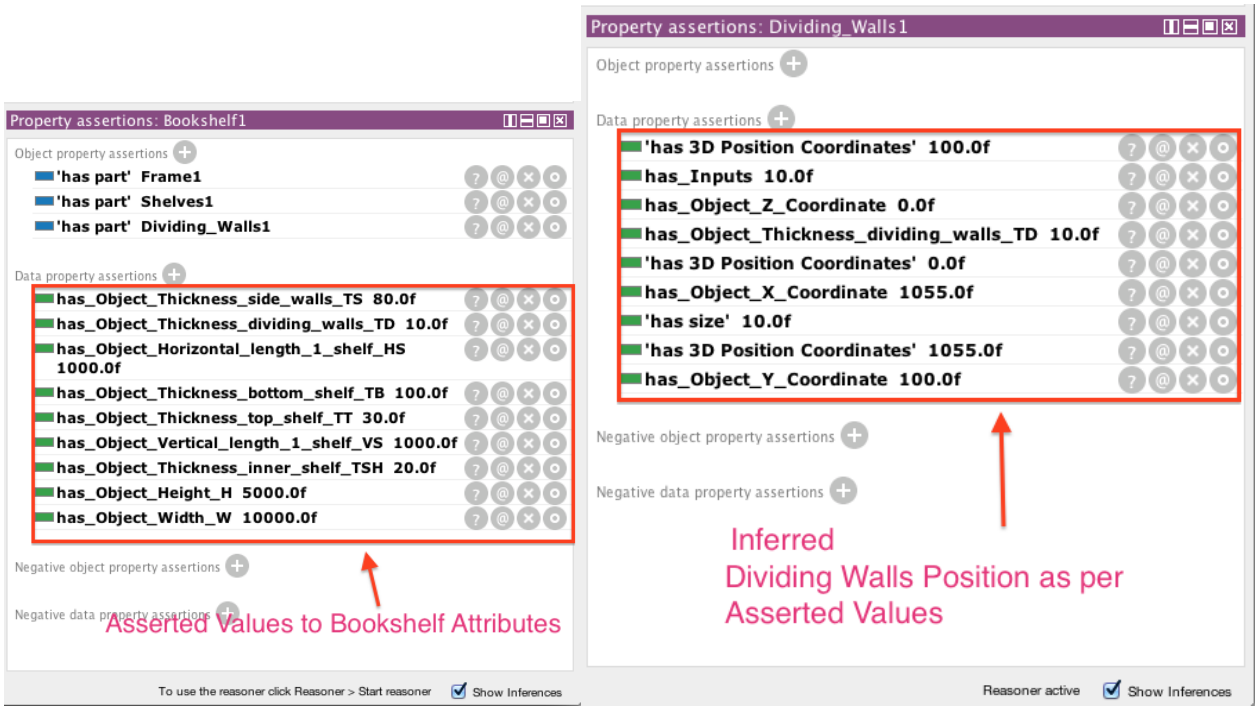


Figure B 16:Inferred Dividing Walls Position Coordinates as per Asserted Values

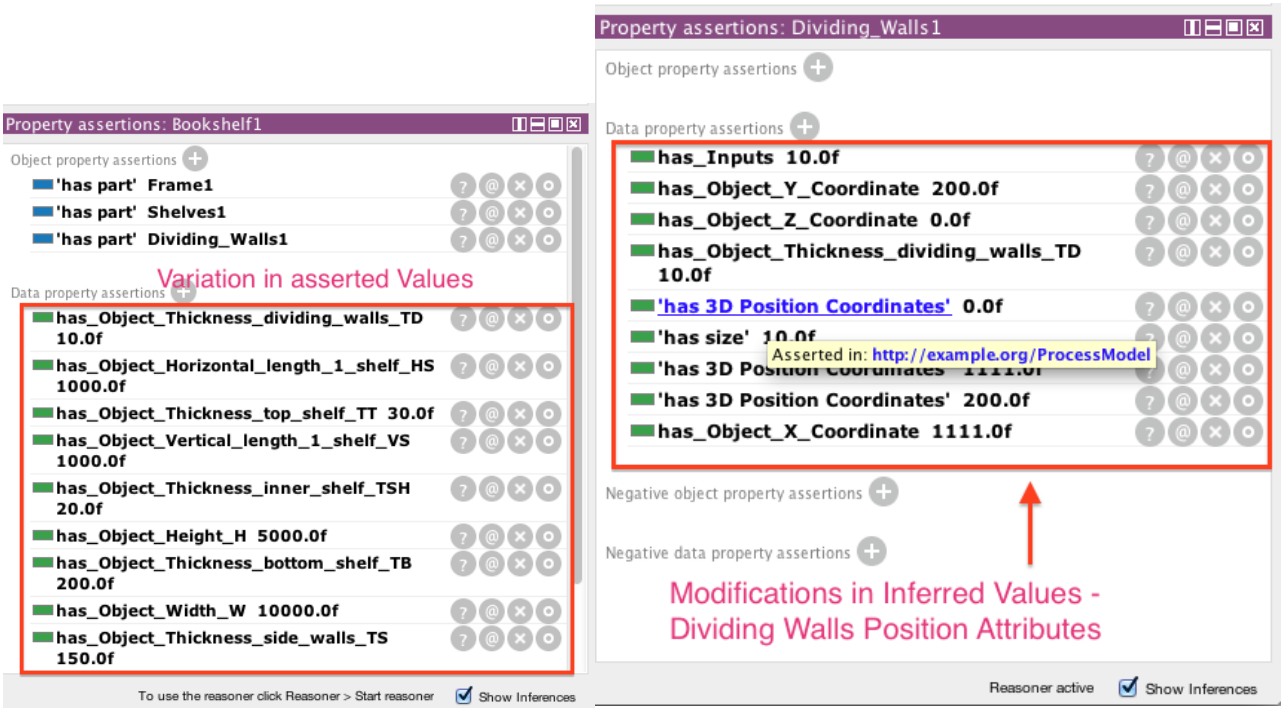


Figure B 17:Modifications in Dividing Walls Position – Variation in Asserted Bookshelf Attributes

Appendix 3: Experimentation of Pilot Use-Cases with Neutral Formal Semantics

G. Process Specification Language

Process Specification Language has been investigated as potential knowledge representation formalism for activity description with focus on manufacturing and production processes based on pilot use-cases. PSL is based on Common Logic Interchange Format (CLIF) and is regarded as ISO 18629. It is based on first order calculus or first order predicate logic (FOPL). The syntax for activity and object description for engineering processes is illustrated as follows –

PSL activity role declaration (ARD) and object declaration syntax:

(define-activity-role
:id <*number*>*
:name <*string*>
:successors <*number*>*
:preconditions <PSL sentence>*
:postconditions <PSL sentence>*)
(define-object
:name <*KIF constant*>
:constraints <*PSL sentence*>*)
(define-parameter
:variable <*KIF variable*>
:constraints <*PSL sentence*>*) (Grüninger and Menzel, 2003)

For representation of inputs and outputs axioms for pilot use-cases, the syntax has been adopted from (Bock and Gruninger, 2004) as explained with the help of a milling process in Figure B 1.

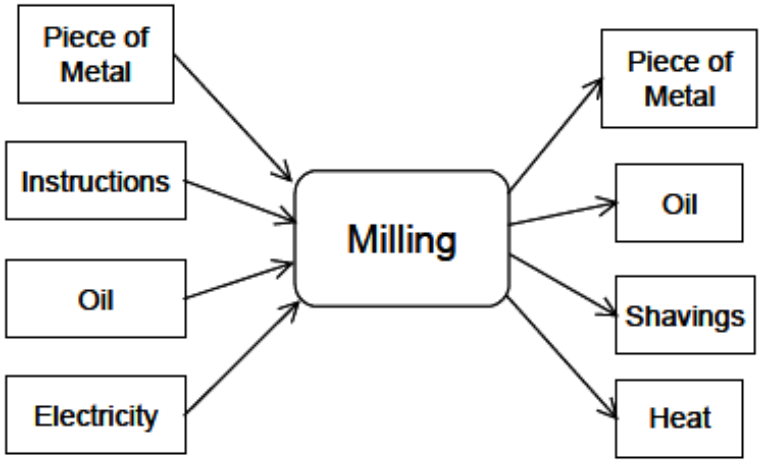


Figure C 1: Inputs and Outputs for a Milling Process (Bock and Gruninger, 2004, Pg 3)

Inputs and Outputs in PSL syntax:

Parameterised term for activities -

```
(forall (?a ?m ?i ?o)
        (implies (= ?a milling(?m ?i ?o))
                 (and (activity ?a)
                      (metal ?m)
                      (instructions ?i)
                      (oil ?o))))
```

Inputs and outputs at activity occurrence –

```
(forall (?x ?s)
    (implies (or (occurrence-input ?x ?s)
                 (occurrence-output ?x ?s))
             (and (object ?x)
             (not (state ?x))
             (activity_occurrence ?s))))

(forall (?x ?s)
    (iff (participant ?x ?s)
             (exists (?t)
                 (participates_in ?x ?s ?t))))

(forall (?x ?s)
    (implies (or (occurrence-input ?x ?s)
             (occurrence-output ?x ?s))
             (participant ?x ?s)))
```

```
(forall (?x ?s2)
        (implies (and (occurrence-input ?x ?s2)
                      (legal ?s2))
                      (exists (?s1)
                              (and (occurrence-output ?x ?s1)
(earlier ?s2 ?s1)))))

(exists (?sDrill ?sMill ?m ?i ?o)
        (and (occurrence_of ?sDrill drilling(?m ?i ?o)
        (occurrence_of ?sMill milling(?m ?i ?o)
        (occurrence-input ?m? sDrill)
        (occurrence-output ?m ?sDrill)
        (occurrence-input ?m ?sMill)
        (occurrence-output ?m ?sMill)
        (earlier ?sDrill ?sMill)
(legal ?sMill)))))

(forall (?x ?s ?f)
        (implies (or (input-state ?x ?s ?f)
                      (output-state ?x ?s ?f))
                      (and (object ?x)
                      (not (state ?x))
                      (activity_occurrence ?s)
(state ?f))))

(forall (?x ?s ?f)
        (implies (input-state ?x ?s ?f)
                      (and (occurrence-input ?x ?s)
                      (prior ?f ?s)
(exists_at ?x (begin_of ?s)))))

(forall (?x ?s ?f)
        (implies (output-state ?x ?s ?f)
                      (and (occurrence-output ?x ?s)
                      (achieved ?f ?s)
(exists_at ?x (end_of ?s)))))

subactivity(subactivity1, activity)
subactivity(subactivity2, activity)
```

H. RuleML

RuleML is a markup language for representing rules using semantic standards and based on horn logic. For experimentation of engineering rules of pilot use-cases with Datalog version of RuleML (Boley et al., 2005), based on XML, RDF, XSLT and OWL, the following syntax was adopted for engineering rule axioms –

Natural Language Sentence - *"Peter Miller's spending has been min 5000 euro in the previous year."*

Datalog RuleML syntax –

```
<Atom>
<Rel>spending</Rel>
<Ind>Peter Miller</Ind>
<Ind>min 5000 euro</Ind>
<Ind>previous year</Ind>
</Atom>
```

Natural Language Sentence - *"A customer is premium if their spending has been min 5000 euro in the previous year."*

Datalog RuleML syntax –

```
<Implies>
<head>
<Atom>
<Rel>premium</Rel>
<Var>customer</Var>
</Atom>
</head>
<body>
<Atom>
<Rel>spending</Rel>
<Var>customer</Var>
<Ind>min 5000 euro</Ind>
<Ind>previous year</Ind>
</Atom>
</body>
</Implies>
```

Natural Language Sentence - *The discount for a customer buying a product is 7.5 percent if the customer is premium and the product is luxury."*

Datalog RuleML syntax –

```
<Implies>
<head>
<Atom>
<Rel>discount</Rel>
<Var>customer</Var>
<Var>product</Var>
<Ind>7.5 percent</Ind>
</Atom>
</head>
<body>
```

<And>
<Atom>
<Rel>premium</Rel>
<Var>customer</Var>
</Atom>
<Atom>
<Rel>luxury</Rel>
<Var>product</Var>
</Atom>
</And>
</body>
</Implies>

Natural Language Sentence - *"The discount for Peter Miller buying a Porsche is 7.5 percent"*

Datalog RuleML syntax –

<Atom>
<Rel>discount</Rel>
<Ind>Peter Miller</Ind>
<Ind>Porsche</Ind>
<Ind>7.5 percent</Ind>
</Atom>